**DOCUMENTACION LIMO KNIGHT**

Contenido

[PROGRAMACIÓN 2](#_Toc509488825)

[MANAGERS 2](#_Toc509488826)

[ENUMERADOS 3](#_Toc509488827)

[MECHANICS 3](#_Toc509488828)

[CABALLEROS Y ATRIBUTOS 3](#_Toc509488829)

[FORMULAS 4](#_Toc509488830)

# **PROGRAMACIÓN**

Este apartado hace referencia a las siguientes preguntas: ¿Cómo se organiza la programación? ¿Qué pautas hay definidas dentro de un script? ¿Qué idioma se utiliza en los scripts?...

1. La programación **está organizada dentro de un namespace común llamado LK**, todo script debe de estar contenido en este y dentro pueden estar enumerados públicos o la propia clase.
2. Dentro de un script se empiezan definiendo las variables privadas que son visibles desde el inspector seguidas de las que son puramente privadas. Los **públicos deberán de ir en último lugar y ser *accesores* de otros privados salvo un caso excepcional**. Dichas **variables siempre empezaran con ‘\_’ y alternando mayúsculas** para el inicio de cada palabra, salvo la primera palabra de todas que será todo minúsculas. Obviando esto las prácticas deben de ser las de sentido común en programación OO.

**Las funciones y todas las variables deberán de ser inglés**, sin embargo, **los comentarios y los sumarios autogenerados /// deberán de ir en español** para una mayor claridad de quien va a programarlo.

## **MANAGERS**

Los managers son clases que heredan el script *Singleton<T>* si viéramos la declaración nos daríamos cuenta de que es una clase que se autodefine como un singleton de sí mismo, esto significa que un objeto que siempre deba de existir sin importar si se está jugando o no seguirá este convenio.

Los managers realizan operaciones de un tipo particular, es decir son entidades que hacen un tipo de función, solo pueden tener un Script asociado para diferenciar aún más sus tareas. Solo pueden comunicarse entre sí a través del GameManager que es la interfaz que todos usaran.

Actualmente están los siguientes managers:

* **GameManager:** Manager filtrador que permite comunicar los diferentes managers entre sí.
* **UIManager:** Encargado de deshabilitar y habilitar los diferentes paneles (no obstante, en la versión final no habría tantos como hay en la versión *alpha*) esto puede ir desde previo al juego a la propia UI.
* **LevelManager:** Encargado de recrear los mapas, una vez comienza el juego, en base a unos parámetros, lanza sus subrutinas para crear el mapa y posicionar al caballero en el terreno de juego.
* **CharacterManager:** Es el encargado de establecer que caballero se selecciona en que momento, seleccionandole su sprite adecuado y quizás manteniendo sus atributos base -llegado el momento-.
* **XMLManager**: La base de dato se basa en XML, de forma predefinida hay unos ficheros creados (esto se hace en la fase de pruebas), cada tipo de fichero xml ‘InfoPersonaje.xml’ tiene un método para crearlo (según un formato especifico), cargar los datos (en dicho formato) y otro para resetear el fichero a un estado inicial (por ejemplo, poner el nivel del usuario y la experiencia a 0).
* **PoolManager**: Encargado de instanciar cualquier cosa en la escena que pueda variar, enemigos, personajes, proyectiles. Para cada tipo nuevo solo se hará un resource.load y para cada objeto simultaneo se crea una instanciación, no obstante a medida que se vayan haciendo se vuelven a meter en el diccionario para recargarlos sin tener que instanciar. Quedaría hacer una corrutina que cada x tiempo liberara memoria de aquellos diccionarios que lleven más de X minutos sin usarse.

## **ENUMERADOS**

Los enumerados nos permiten ahórranos comparaciones con strings al mismo tiempo que son una fácil forma de dividir diferentes categorías. Todos los enumerados empiezan con ‘TypeXXX’. Los enumerados siempre tendrán alguna relación con los XML, ya sea por el tipo de caballero que es, el tipo de enemigo que es o cualquier otro parámetro, pero sirven de identificador dentro del XML.

Todos los enumerados tendrán el nombre en inglés y un número si es relevante o no.

* **Character:** Agrupa los diferentes tipos de caballeros que hay, todos siguen el mismo formato Proposito+Knight. Entre ellos tenemos ‘BaseKnight’ que sería por ejemplo el caballero básico. En base a su nombre, se debería de cargar un Sprite que coincida con este.
* **Level:** El juego tiene 5 niveles, cada uno equivalente a los pisos de un edifico o un castillo (floor, roof…) cada uno de ellos hace que los enemigos crezcan sus estadísticas en un porcentaje y ofrecen un mismo bonus de experiencia.
* **Floor:** Aunque en la fase alpha no hay diferentes tipos de sprites para suelos o paredes, pero permite hacer diferenciaciones entre ‘muros, suelos, puertas…’.
* **Enemy:** Enumerado para diferentes tipos de enemigos, no obstante, los jefes empezarán por el número 100 en adelante. O desde el X00 más próximo al número actual de enemigos que haya.

## **SCRIPTS - PLAYER**

El jugador, no es más que un prefab llamado ‘Player’ sin embargo para cada caballero se carga en el SpriteRenderer una imagen u otra y una serie de scripts particulares. Nota: Si el script no lleva algo como ‘BaseKnightMovement’ quiere decir que se usará el script normal que no tiene como prefijo el nombre de un caballero.

* **CharacterAtribute**: Script que obtiene los atributos bases (que no cambian) del CharacterManager, calcula -siguiendo el apartado de fórmulas- el valor final de cada atributo y los cálculos para cada propiedad. Posee métodos para que, cuando el script que coge un objeto -lo coge- le llame para que actualice sus valores, así otro que necesite saber el daño que debe de hacer (los proy1ectiles del jugador, por ejemplo) lo tengan siempre actualizado.
* **PlayerControl**: MiniScript para avisarle a la cámara de que debe de centrarse en el personaje porque acaba de ser instanciado en la escena.
* **PlayerMovement**: Script que controla el movimiento básico de los caballeros, al ser genérico no hace falta cambiarlo, en caso de que alguno se mueva de forma especial deberá de sustituirse este script por otro.
* **Crosshair**: Internamente todos los Caballeros tienen un sistema para mostrar una esfera a modo de crosshair (o mirilla) hacia donde iran los proyectiles, no obstante, si alguna clase de caballero no dispara entonces se puede eliminar el componente que contiene esto (ProjectileSpawn).

## **SCRIPTS - PANELES**

Previo al inicio del jugador la pantalla de selección de jugador, lanzada de dados, todos tienen scripts que realizan una función determinada. Aunque en la versión final el menú será toda una hoja de personaje los scripts permanecerán iguales, en teoría.

* **AtributeLoader**: Script que solicita al GameManager los AtributosPersonaje del caballero que se ha seleccionado y modifica los valores de los textos a la información que contenga dicha información.
* **ThrowDices**: Script que dado un array de textos de atributos a mostrar y un texto de aviso lanza los dados en base al nivel actual del USUARIO de forma global.
* **DiceModeSelection**: Obtiene los valores de los textos del anterior para el valor de los dados y le solicita al GameManager los AtributosPersonaje del caballero que se seleccionara. A continuación, en base a si se quiere repartir de forma aleatoria o normal se asignan los valores con las formulas, siempre siguiendo (atributo\_base + atributo\_base \* valor dado).
* **DifficultModeSelection**: Permite colocar dos desventajas y ventajas como máximo, para obtenerlas lo que hará será preguntar al XMLManager que le de la lista de ventajas y desventajas y se asignarán de algún modo al jugador y habrá un script que en base a lo que haya salido haga su efecto.

# **MECHANICS**

Este apartado compete a las mecánicas del juego desde cómo deben de funcionar ciertos personajes a apartados más genéricos como el sistema de dados o atributos. Este apartado puede variar y debe de tomarse como una referencia más que como una guía invariable.

## **CABALLEROS Y ATRIBUTOS**

Los atributos son un apartado común para cada caballero. Cada uno codifica cierta parte de su comportamiento, desde que vida máxima tiene a cuenta energía puede almacenar o cuanta regenera por minuto de juego. Además, ciertos atributos pueden permitir interacciones en el juego, p.e*: romper una roca si se tiene más de 100 constitución*.

En total, para un caballero cualquiera existen los siguientes atributos:

* **Constitución:** Determina la cantidad de vida que posee el caballero y la cantidad de energía total que dispone para usar habilidades o consumibles.
* **Destreza:** Determina la capacidad de critico que tiene el personaje por golpe (hasta un límite máximo, a partir de ese punto el daño crítico es el que se incrementa).
* **Agilidad:** Afecta a cuan ágil es el caballero para disparar.
* **Suerte:** La suerte es un atributo que afecta a los objetos que aparecen en un nivel para ese caballero, puede ser que surjan objetos de pools mejores o que los enemigos suelten mejores recompensas. Además, también incrementa el bonus de experiencia total, al ser un atributo pasivo que hace aún más difícil el juego.
* **Poder:** El poder determina el daño que se hace de forma base a un enemigo, esto sería suponiendo dos cosas: que el enemigo tiene 0 de defensa y que el daño que se inflige es neutral, dado que posibles elementos modifiquen el daño real.

A la hora de asignar los incrementos -información del caballero- nunca superaran más de 10 en total y ningún atributo será más de 3 en ningún atributo.

## **CABALLEROS - FORMULAS**

Las fórmulas determinarán las propiedades de los caballeros, para toda formula suelen haber tres partes:

* **Atributos totales:** Resultado aritmético del atributo total con sus incrementos o multiplicadores naturales, este se calcula antes de empezar el juego.
* **Atributos por objetos:** Los objetos pueden sumar o restar a este atributo total.
* **Multiplicador por dado:** Este porcentaje se calcula al lanzar los dados y se mantiene constante, determina del total que daño se hace. El valor del dado será el mínimo entre (nivel+1, 100+nivel-1) y el multiplicador será de ese valor / 10.
* **Multiplicadores finales:** Los multiplicadores afectan a los atributos totales y a los atributos por objetos, es decir es un factor global. Ya sea por dificultad, por elemento, o por algún criterio del nivel. Así pues, para el cálculo de los atributos totales del caballero se hará como:

|  |
| --- |
| **Atributo X** (**AX**)= Valor X Base + (Nivel Caballero \* Incremento de X) |

Con el valor base se procede a calcular el valor del atributo que puede ir variando en tiempo de juego:

|  |
| --- |
| **Atributo Final (AFX):** (AX + Atributo Objetos) \* (Multiplicador Dado + Multiplicador Finales) |

Con dichos atributos finales ya se puede jugar para calcular las propiedades de los caballeros, que evidentemente se calculan constantemente dado que dependen de los atributos finales, a su vez puede ser que cada propiedad dependa de alguna particularidad del caballero, en cuyo caso se añade al total de las propiedades a las que afectara:

* **Daño:** Valor de daño máximo que haría el caballero sobre un enemigo. Este valor puede variar sobre la defensa del enemigo no obstante esto se conoce como ‘daño\_real’ y su cálculo no debe de almacenarse pues variara para cada enemigo.

|  |
| --- |
| **Daño:** AF\_Poder  **Daño Real** = Max(Daño\*0.2, Daño - Defensa de Enemigo + [Daño \* Crítico](*if critico*) ) |

* **Vida:** Determina la vida total que dispone un jugador, esta podría recuperarse con objetos…

|  |
| --- |
| **Vida:** 10 + AF\_Const |

* **Energía:** Determina el límite máximo de energía que se dispone para el uso de habilidades u objetos activables. El ratio de refresco (matar monstruos) lo indica cada objeto propio, asi como el coste de usarlo.

|  |
| --- |
| **Energía:** AF\_Const / 2.5 |

* **Critico:** El crítico determina con qué probabilidad el crítico golpeará al enemigo, por defecto el crítico se salta la posible armadura que tenga el enemigo y de base hace un 1.5 de daño al enemigo.

|  |
| --- |
| **Critico:** 1.5 + AF\_Destreza / 250 |

* **Probabilidad crítica:** La probabilidad critica implica con que probabilidad este critico, esto es un numero aleatorio si al lanzar un dado sale menor al número que toca entonces se pone.

|  |
| --- |
| **Probabilidad Crítica:** Min(1f, 0.05f + AF\_Destreza/300) |

**Multiplicador experiencia Extra:** Esta experiencia solo se suma al caballero cuando termine (no al usuario).

|  |
| --- |
| **Multiplicador Experiencia extra:** 1 + AF\_Suerte / 250 |

* **Mejora Pool:** La mejora del pool se utiliza para saber que contenidos habrá en un cofre o en una sala del tesoro.

|  |
| --- |
| **Mejora Pool**: Min(0.8, AF\_Suerte / 420);  Si las probabilidades del pool son: Normales: 45%, Raros: 35% y Legendarios: 20% y obtenemos un 0.35:   * Normales: 0.45\* (1-0.35) = 0.2925 * Raros: 0.35 \* (1+0.35) = 0.47 * Legendarios 0.20 \* (1+0.35/1.7) = 0.2411   Y al normalizar las probabilidades serían realmente:   * Total = 1.0003 * Normales = 0.2925 /1.003 = 0.2924 -> 29.24% * Raros= 0.47 / 1.003 = 0.4698 -> 46.98% * Legendarios = 0.2411 / 1.066 = 0.2410-> 24.10% |

* **Velocidad de Proyectil:** Los proyectiles tienen rango limitado (en base a su velocidad) este valor implica cuan rápido puede viajar la bala.

|  |
| --- |
| **Velocidad Proyectil:** 1/3 + AF\_Agilidad/400 |

* **Retardo de Disparo:** Indica el tiempo que debe de esperarse entre disparos, lo que incrementa el dps general.

|  |
| --- |
| **Critico:** 1/(4+AF\_Agilidad/200) |