# Resumen

El proyecto realizado es un point and click con vista top down en el que el jugador tendrá que ir combatiendo contra oleadas de diferentes enemigos.

**Acciones**:

* Moverse: Al hacer clic izquierdo sobre un lugar al que el jugador se pueda mover.
* Atacar: Al hacer clic izquierdo sobre un enemigo.
* Coger objeto: Al hacer clic izquierdo sobre un objeto (bola azul)
* Usar objeto (Inventario): Al hacer clic izquierdo en un objecto del inventario. Si es un objeto equipable se equipará, y si es uno consumible se consumirá
  + Atajo para consumir poción: barra espaciadora
* Cámara: Zoom con la rueda del ratón y movimiento lateral moviendo el ratón horizontalmentecon el botón 3 (rueda del ratón) pulsado.

**Estadísticas**:

* Los personajes tienen una serie de estadísticas que guardan relación con el combate:
  + Damage: daño base de los ataques
  + Armor: tiene un valor de 0 a 100 y disminuye el daño recibido. Si el jugador tiene 20 de armadura, los golpes recibidos disminuyen un 20% su daño.
  + Accuracy: precisión. Probabilidad de que acierte el golpe.
  + Critical Probability: probabilidad de hacer un golpe crítico.
  + Critical Multipier: multiplicador del daño crítico
* En el caso del jugador, su personaje tiene una serie de características base y éstas se modifican en base a los objetos que se equipe.

**Ataques**:

* Una vez que el jugador da la orden de atacar, el personaje se moverá hacia el enemigo y le atacará automáticamente.
* Tipos de golpe:
  + **Básico**: el objetivo recibe el daño base.
  + **Crítico**: el enemigo recibe el porcentaje del multiplicador de crítico.
  + **Miss**: el objetivo no recibe daño.

# Índice

## Clases principales (y uso de propiedades con get y set):

* **Char\_Base.cs**. Clase padre de los enemigos y del jugador. A continuación podemos ver algunas de las propiedades de la clase, cuyo get se usa en algunos casos para poder acceder a las variables desde otros scripts y cuyo set se usa para clampear las variables entre dos valores.

Texto

Descripción generada automáticamente

## Uso de scriptable objects en el constructor:

* **Item\_Base.cs**. El parámetro del contructor es el scriptable object de ItemData, que alberga toda la información necesaria para el item\_Base. Esta clase hace de padre para el resto de las clases de objetos (Item\_Consumable.cs e Item\_Equipable.cs, que a su vez es la clase padre de Item\_Armor.cs e Item\_Weapon.cs), por lo que estas también tienen un constructor con un parámetro de scriptable object específico para cada clase.

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

## Clases abstractas de la que hereden dos clases y que implemente 2 métodos

* **Item\_Base.cs**. Es una clase abstracta que implementa un solo método, que se usa para cuando el jugador usa el objeto. Dos de las clases que heredan directamente de esta son **Item\_Equipable.cs** e **Item\_Consumable.cs** y sobrescriben el método de Action

Pantalla de un reloj

Descripción generada automáticamente con confianza mediaTexto

Descripción generada automáticamente

Texto

Descripción generada automáticamente con confianza media



Texto

Descripción generada automáticamente con confianza media

## Uso de interfaces en las clases derivadas

* **IEquipable**. Esta interfaz se usa en los objectos equipables (**Item\_Equipable.cs**) e implementa los métodos de equipar y desequipar.

Texto

Descripción generada automáticamente



* **IConsumable**. Esta interfaz se usa para los objectos consumibles (**Item\_Consumable.cs**) e implementa el método de consumir.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente



## Colección de la librería System.Collection

* **CharBase.cs**. Esta es una de las clases en las que se usa las listas, que forman parte de la librería de System.Collection.

Una captura de pantalla de una computadora

Descripción generada automáticamente

## Colección propia

* **GenericEnemyQueue.cs**. Se trata de una colección genérica que acepta valores nulos y semántica de referencia. Se usa en el script de **WavesManager.cs** para almacenar las oleadas de los enemigos

Texto

Descripción generada automáticamente

## Evento (EventHandler)

* **Char\_Base.cs**. En esta clase se implementa un método de tipo EventHandler para actualizar las barra de vida de los enemigos y del jugador cada vez que se modifica la vida actual. La clase de EventArgs que se usa tiene dos propiedades enteras, una para la vida actual y otra para la máxima, que se asignan en su contructor. Las clases de **PlayerHpBar.cs** y **EnemyHpBar.cs** se suscriben a este evento para actualizar la barra de vida del jugador y de los enemigos.



Imagen que contiene naranja, conjunto, oscuro, hombre

Descripción generada automáticamente

Texto

Descripción generada automáticamente



Texto

Descripción generada automáticamente

## Evento propio

* **Char\_Base.cs**. En esta clase se implementan un evento propios para actualizar las estadísticas que se muestran en el inventario del jugador. Se invoca en esta misma clase cada vez que se actualizan las estadísticas del jugador. La clase **InventoryStatsPanel.cs** se suscribe a este evento para actualizar los valores cada vez que se invoca el método.



Texto

Descripción generada automáticamente



