

Universidad Complutense de Madrid
Facultad de Informática
Máster en Ingeniería Informática - Desarrollo de Videojuegos

Heroes of Dungeons & Dragons

20 de octubre de 2020

Daniel Bastarrica Lacalle
Jose Javier Cortés Tejada

Índice

1. Descripción del proyecto	4
2. Versionado	4
3. Vista general	5
3.1. Ejemplo de partida típica	5
3.2. Ejemplo de uso de la herramienta	7
4. Detalles de la herramienta	8
4.1. Estructuras de datos	8
4.1.1. ST_Tile	8
4.1.2. ST_Unit	9
4.1.3. ST_Level	10
4.2. Generación del nivel	11
4.2.1. Uso del trazado de rayos en el grid	11
4.3. Generación de caminos óptimos con A*	12
4.4. Movimiento por el mapa	13
5. Menús y modos de juego	14
5.1. Menús <i>out-game</i>	14
5.1.1. Pantalla de título	14
5.1.2. Pantalla de controles	15
5.2. Menús de configuración	16
5.3. Interfaz y control	17
5.4. Pantallas de fin de juego	18
6. Jugabilidad	19
6.1. Mecánica	19
6.1.1. Turnos	19
6.1.2. Movimientos	19
6.1.3. Ataques	20
6.1.4. Inteligencia Artificial	21
6.2. Dinámica	21
6.2.1. Victoria y derrota	21
6.2.2. <i>M.E.T.A.</i>	22
6.3. Estética	22

7. Contenido	23
7.1. Niveles	23
7.2. Personajes	24
7.3. Araña	24
7.4. Orco	25
7.5. Oso	26
7.6. Trol	27
7.7. Mago esqueleto	28
7.8. Esqueleto rey	29
7.9. Esqueleto guerrillero	30

1. Descripción del proyecto

El presente proyecto es una herramienta para crear juegos del género *TRPG* ([Tactical Role-playing Game](#)) de vista isométrica. Algunos ejemplos de juegos de este género podrían ser *Final Fantasy Tactics A2* o la mayoría de títulos de la saga *Fire Emblem*. Junto con la herramienta se han desarrollado dos niveles para probar el desempeño de la misma.

Este proyecto permite construir niveles haciendo uso de archivos *JSON* y trazado de rayos para el control de los elementos presentes en el entorno. Como principales fuentes de inspiración hemos tenido los juegos ya mencionados y hemos tenido como referente el proyecto [IsoUnity](#).

2. Versionado

De cara al control de versiones del proyecto nos planteamos diversas alternativas. La primera de ellas fue *Github*, sin embargo los archivos de más de 100 MB (límite de subida en *GitHub* por archivo) fueron un problema a la hora de tener en cuenta esta solución. Además la principal ventaja de usar *GitHub* es controlar los cambios en el código, sin embargo al tratarse de un proyecto que solo usa *Blueprints* esta opción carecía de sentido.

Otra alternativa que exploramos fue *Perforce*. Esta herramienta nos da la capacidad de bloquear archivos de forma que solo pueden ser modificados por una persona de forma simultánea, lo cual no nos convenía dado que es un proyecto relativamente pequeño sin módulos aislados que aprovechen esta ventaja.

En vista de que es un proyecto de ámbito reducido, y la baja o casi nula complejidad en la organización de los desarrolladores al ser solo 2 personas, optamos por usar un control de versiones simplificado. Por ello hemos usado *Google Drive*, de forma que cada día se subía una versión nueva del proyecto indicando los cambios realizados. Además organizamos reuniones periódicas para agrupar los cambios si había habido dos ramas de trabajo en paralelo, así como para comentar el estado y las próximas tareas de cara al desarrollo.

3. Vista general

3.1. Ejemplo de partida típica

Al tratarse de una herramienta lo primero que verá el jugador es una pantalla donde puede seleccionar el nivel que quiere jugar, así como el modo de juego que desee. Las unidades de cada equipo son asignadas por defecto.

Tras haber seleccionado un nivel y un modo de juego empezará la partida. Lo primero que verá el jugador será el mapa del nivel, las unidades que están en él, el orden que siguen los turnos de las unidades y la información de la unidad actual.

Cuando sea el turno de una de sus unidades el jugador podrá seleccionarla y ver las acciones disponibles para dicha unidad en su turno (moverse, atacar o esperar). Si selecciona la opción de moverse aparecerán en color azul claro las casillas del mapa a las que esa unidad puede llegar. Hecho esto si pasamos el cursor sobre una de ellas cambiará el color de dicha casilla y el camino hasta ella, es decir, tomarán un color azul oscuro para aclarar cuál se está señalando con el cursor y el camino óptimo hasta ella. En caso de que se haga click sobre dicha casilla la unidad se moverá hasta allí siguiendo el camino marcado.



Figura 1: Acciones disponibles y rango de movimiento

Si es seleccionada la opción de atacar pero no existen unidades dentro de su rango de ataque aparecerá un mensaje indicando dicha situación. Por el contrario si tiene enemigos a su alcance, verá resaltadas en color amarillo las casillas a las que la unidad actual es capaz de atacar. Si posiciona el cursor del ratón encima de una de las unidades alcanzables, aparecerá encima de dicha unidad un recuadro que indicará los puntos de salud actuales de la unidad y los puntos restantes después de realizar dicho ataque, además del daño infligido.

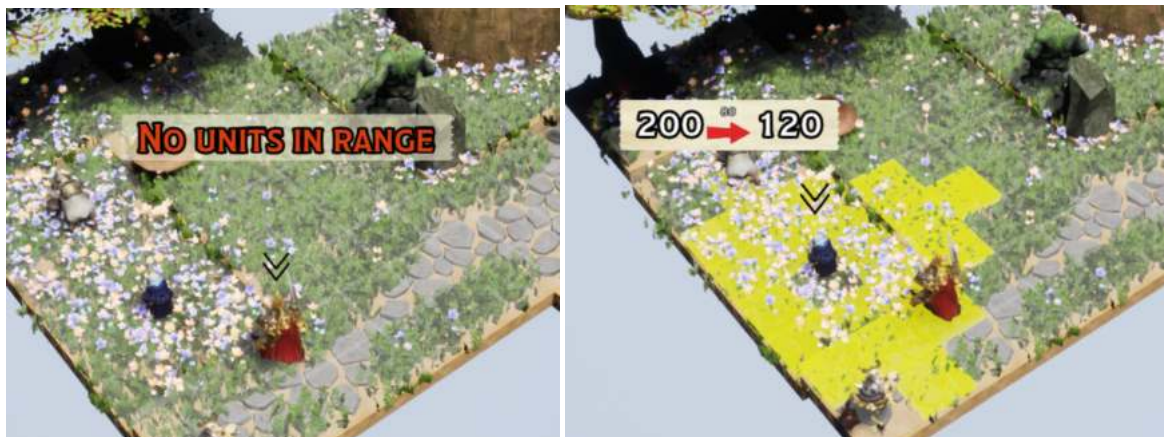


Figura 2: Sin unidades en rango y rango de ataque

En caso de que se hayan agotado la opción de moverse y atacar ambos botones estarán deshabilitados del menú de acciones. La última opción hace que la unidad pase turno, de forma que se verá cómo la barra superior izquierda de turnos avanza y la nueva unidad con el turno actual activo aparecerá en la barra inferior de información. También se actualizará el cursor que hay encima de la unidad con el turno actual.



Figura 3: Menú sin acciones disponibles

Cuando al pasar turno le toque a una unidad controlada por la IA esta se moverá sola y actuará de la forma más adecuada respecto al posicionamiento de las unidades enemigas en el mapa. Una vez haya terminado el turno de todas las unidades se recalculará de nuevo la lista de turnos y seguirá el juego como ya se ha explicado. El juego finalizará cuando se haya satisfecho la condición de victoria especificada por el modo de juego o el jugador pierda todas sus unidades.

En resumidas cuentas, a medida que el jugador vaya avanzando turnos y el rival controlado por la IA haga sus jugadas, el jugador irá aprendiendo qué estrategias son recomendables seguir y cuáles usa la IA. Además aprenderá qué alcances tiene cada unidad tanto de movimiento como de ataque, así como el daño que realiza. Con esta información, el jugador podrá ir depurando sus acciones para intentar superar los niveles, de forma que en función del modo de juego desarrolle una determinada estrategia a seguir.

3.2. Ejemplo de uso de la herramienta

En primer lugar la principal funcionalidad que proporciona la herramienta es la de crear y editar mapas mediante ficheros de texto en formato *JSON*. Para ello en cada elemento del fichero se deberá indicar el nombre que se le quiere dar a la casilla, sus posiciones en los 3 ejes cardinales, el *asset* del material que se quiere utilizar e indicar si dicha casilla será una de las posiciones de destino en el modo de juego de alcanzar la zona.

Por otro lado para construir el escenario la herramienta proporciona mediante trazado de rayos una lectura automática de los elementos presentes de cara a inhabilitar como alcanzables las casillas sobre las que se sitúen dichos elementos. De esta manera se simplifica la construcción de escenarios más ricos y elaborados ya que basta con posicionar elementos sobre el mapa para que sean procesados.

Para añadir personajes basta con heredar del *blueprint* base de unidad y modificar la malla del personaje para que sea la deseada, así como las animaciones correspondientes a dicha malla.

4. Detalles de la herramienta

Los *TRPG* son títulos que generalmente constan de un *grid* sobre el que se mueven las unidades y un conjunto de reglas que definen la jugabilidad (vease fórmulas de daño, reglas para el movimiento de las unidades o condiciones de finalización del juego).

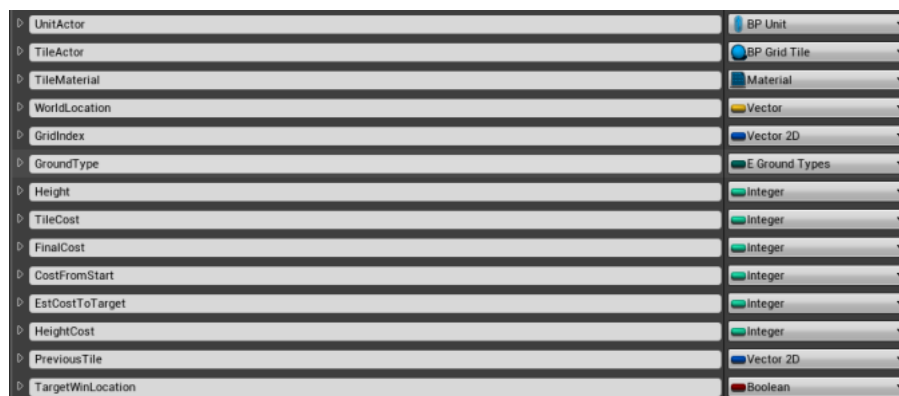
No obstante otros títulos dentro del mismo género como **Mario + Rabbids Kingdom Battle** o **X-COM 2** añaden nuevas mecánicas innovadoras para dotar de personalidad propia al juego, como son las coberturas o la posibilidad de usar unidades enemigas como trampolines. Cada título incorpora nuevas ideas que enriquecen el género, sin embargo las características base que definen un TRPG son las comentadas al principio.

Con este proyecto hemos creado una herramienta que permite a otros desarrolladores elaborar TRPG con las características base de dicho género. En los siguientes puntos de este capítulo se explican todos los aspectos de la herramienta, desde las estructuras de datos que funcionan por debajo hasta el uso de la misma y de sus componentes.

4.1. Estructuras de datos

4.1.1. ST_Tile

Esta estructura de datos guarda información sobre las casillas del tablero. En la figura 4 tenemos una captura de pantalla de sus campos con sus respectivos tipos de datos.



UnitActor	BP Unit
TileActor	BP Grid Tile
TileMaterial	Material
WorldLocation	Vector
GridIndex	Vector 2D
GroundType	E Ground Types
Height	Integer
TileCost	Integer
FinalCost	Integer
CostFromStart	Integer
EstCostToTarget	Integer
HeightCost	Integer
PreviousTile	Vector 2D
TargetWinLocation	Boolean

Figura 4: Campos de la estrucutra de datos **ST_Tile**

Los campos `UnitActor` y `TileActor` hacen referencia a la unidad que ocupa esa casilla y la casilla en sí, respectivamente. Por otro lado tenemos `WorldLocation`, que determina donde está ubicada la casilla a nivel global. `GridIndex` determina las coordenadas (X,Y) de dicha casilla como si se tratase de una matriz, es decir, normaliza `WorldLocation` en base al ancho de la propia casilla.

Por ejemplo, dado un tamaño de casilla de 150 unidades y una *WoldLocation* con valores (300, 600, 0), tenemos que el *GridIndex* que ocupará dicha casilla en el tablero es el (2, 4).

Los parámetros restantes son usados para la generación de caminos mínimos con A*. Estos son:

- **GroundType**: determina el tipo de suelo de una casilla. Este puede ser:
 1. **None**: no hay casilla en esa localización.
 2. **Obstacle**: esa casilla está ocupada por un obstáculo que impide a los personajes moverse por ella.
 3. **Water**: esa casilla es agua.
 4. **Ground**: casilla estándar por la que se puede caminar.
- **Height**: altura de la casilla respecto a la altura mínima (0).
- **TileCost**: coste de desplazarse a la casilla, determinado por el tipo de suelo (por ejemplo, el agua puede ralentizar al jugador a la hora de moverse...).
- **FinalCost**: coste final de moverse a una casilla.
- **CostFromStart**: coste parcial de moverse a una casilla desde el origen.
- **EstCostToTarget**: coste estimado para alcanzar dicha casilla desde otra.
- **PreviousTile**: tile previo por el que pasó la unidad antes de llegar a esta casilla.

4.1.2. ST_Unit

Esta estructura almacena la información de las unidades del juego (figura 5). Para elaborar las unidades hemos tomado como referencia el título **Final Fantasy Tactics A2**. Con ello tenemos dos tipos de ataques y defensas (físico y mágico), además de un parámetro *Lucky* que se tendrá en cuenta a la hora de realizar golpes críticos y un parámetro *Speed* que será usado para establecer el turno de cada unidad.

Los atributos *Team* y *Attack_Range* determinan el equipo al que pertenece la unidad y el rango de ataque de la misma, respectivamente. También disponemos de atributos para almacenar el nivel de la unidad (*Level*), su nombre (*Name*) y su bando (*UnitType*), así como otros para su salud y puntos mágicos (*MaxHealth* y *MaxMana* respectivamente).

▷ Level	Integer
▷ Name	Text
▷ UnitType	E Units
▷ MaxHealth	Integer
▷ MaxMana	Integer
▷ PhysicalAttack	Integer
▷ PhysicalDefense	Integer
▷ MagicAttack	Integer
▷ MagicDefense	Integer
▷ Lucky	Integer
▷ Speed	Integer
▷ Steps	Integer
▷ Height	Integer
▷ CriticalRatio	Float
▷ Team	String
▷ Attack_Range	ST Attacks
▷ Image_150x150	Texture 2D
▷ Image_450x200	Texture 2D
▷ Image_Full	Texture 2D

Figura 5: Campos de la estructura de datos **ST_Unit**

Por último todas las unidades disponen de 3 imágenes que son utilizadas en la UI del juego y de dos parámetros más, Steps y Height. Steps es usado a la hora de calcular las casillas alcanzables para la unidad así como el camino hasta ellas, mientras que Height determina si es posible alcanzar una casilla que presente un desnivel respecto al jugador.

4.1.3. ST_Level

Los niveles del juego son cargados desde archivos externos, y su contenido está determinado por lo campos de esta estructura de datos (figura 6).

▷ X	Float
▷ Y	Float
▷ Z	Float
▷ Material	Material
▷ TargetTile?	Boolean
▷ SpawnUnit?	Boolean
▷ SpawnEnemy?	Boolean

Figura 6: Campos de la estructura de datos **ST_Level**

Los atributos *X*, *Y* y *Z* determinan la posición absoluta de la casilla en el mundo, mientras que *Material* define el material que ha de ser aplicado en dicha casilla.

Por otro lado tenemos el atributo *TargetTile?*, el cual determina si una casilla es una casilla objetivo cuando el modo de juego es Conquistar una posición (los modos de juego se explican más adelante). Los atributos *SpawnUnit?* y *SpawnEnemy?* determinan sobre que casillas es posible *spawnear* unidades y enemigos respectivamente.

4.2. Generación del nivel

Los niveles presentes en la herramienta cuentan con 2 componentes principales:

1. Geometría del nivel: es un *Blueprint* que contiene todas las casillas del nivel y es cargado de un fichero externo.
2. Grid autoajustable: este *Blueprint* se coloca encima de la geometría del nivel y realiza un *spawn* de las casillas por las que se desplazarán las unidades.

4.2.1. Uso del trazado de rayos en el grid

Para realizar el *spawn* de las casillas transitables por los personajes hemos usado trazado de rayos. El trazado de rayos es una herramienta donde se crea una línea entre dos puntos y nos devuelve información sobre los elementos con los que ha colisionado.

Para ello hemos definido dos canales de colisión, uno que permite saber si se ha colisionado con el suelo y otro que nos avisa cuando se ha colisionado con un obstáculo. Para llevar a cabo el *spawn* en el mapa de las casillas disponemos de la geometría del nivel, las dimensiones del *grid* y el ancho de la casilla, por lo tanto crearemos una línea en cada casilla de forma que la atraviese de arriba a abajo.

Para cada casilla usaremos dos trazados, el primero tendrá como canal de colisión el suelo y el segundo los obstáculos. Con esto obtendremos la siguiente información:

- si el primer trazado no devuelve nada en esa posición del mapa no es necesario proceder al *spawn* de ninguna casilla.
- si el segundo trazado no devuelve nada y el anterior sí lo hizo tenemos que hacer el *spawn* de una casilla de tipo suelo. En cambio si ambos devuelven algo esa casilla estará ocupada por un obstáculo.

4.3. Generación de caminos óptimos con A*

Para la generación de caminos entre dos casillas, así como para obtener las casillas a las que puede moverse el jugador, hemos usado el algoritmo A*. Este algoritmo hace uso de una función heurística para decidir qué nodo expandir. La función que hemos elegido está relacionada con el coste de llegar a cada casilla, de forma que siempre expande aquellas con el menor coste hasta llegar a una solución óptima.

A* hace uso de dos conjuntos, uno de ellos es el *conjunto de abiertos* donde tenemos las casillas que están pendientes de ser exploradas (es una cola de prioridad ordenada de menor a mayor por coste) y el otro es el llamado *conjunto de cerrados*, es decir, aquellas casillas que ya han sido exploradas.

El algoritmo devuelve como resultado el *path* completo por el que ha de moverse una unidad para llegar a otro punto del mapa y el conjunto de cerrados, que son todas aquellas casillas a las que se puede desplazar (figura 8).

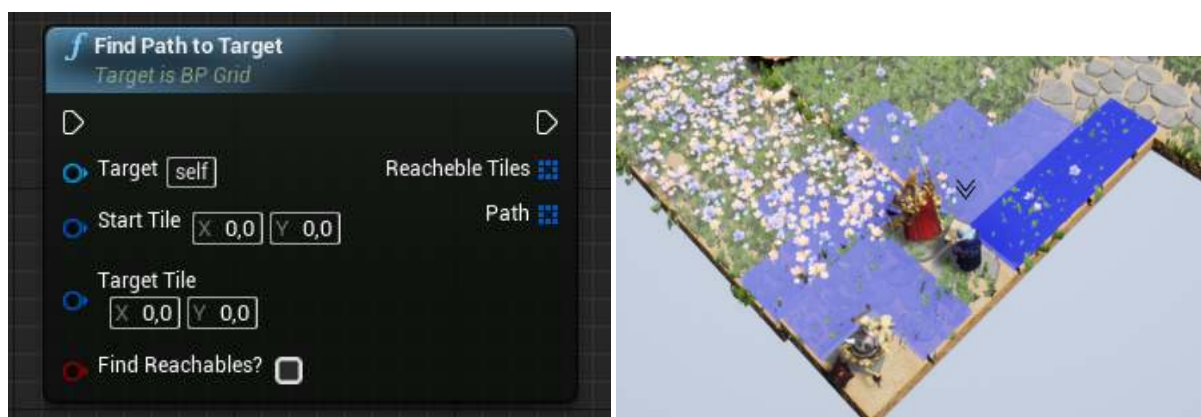


Figura 7: Ejemplo de la información que devuelve el *path* y resultado final tras actualizar las casillas con dicha información. En azul claro tenemos las casillas alcanzables por la unidad mientras que en azul oscuro tenemos el *path* por el que va a moverse.

Dentro del algoritmo se realizan podas a la hora de buscar el camino óptimo. Se tiene en cuenta que la casilla no esté ocupada por un obstáculo, al no ser transitables por las unidades, y además se evitan aquellas casillas que no son alcanzables por la unidad en relación a la altura. Por ejemplo, si nuestra unidad puede alcanzar bloques de una altura de 3 como máximo y el siguiente bloque presenta una diferencia de altura de 5, dicho bloque será ignorado por el algoritmo.

4.4. Movimiento por el mapa

De cara a como desplazar las unidades por el mapa optamos por no usar *NavigationMesh*. Usarlo implicaba usar el calculo del *path* que proporciona *Unreal Engine*, sin embargo no encajaba con el proyecto.

En su defecto optamos por usar un *spline* para el movimiento de las unidades. Los *splines* son componentes que definen un *path* por el que se pueden mover actores u otros componentes. Para generar los puntos del *spline* se ha usado el *path* generado por A*, de forma que cada elemento del mismo es una casilla por la que pasa la unidad.

Para tratar los desniveles del mapa el *spline* toma las casillas más altas o bajas como puntos con distinta elevación (en su eje correspondiente) y a la hora de mover la unidad por ellos se traza una línea recta de uno a otro.

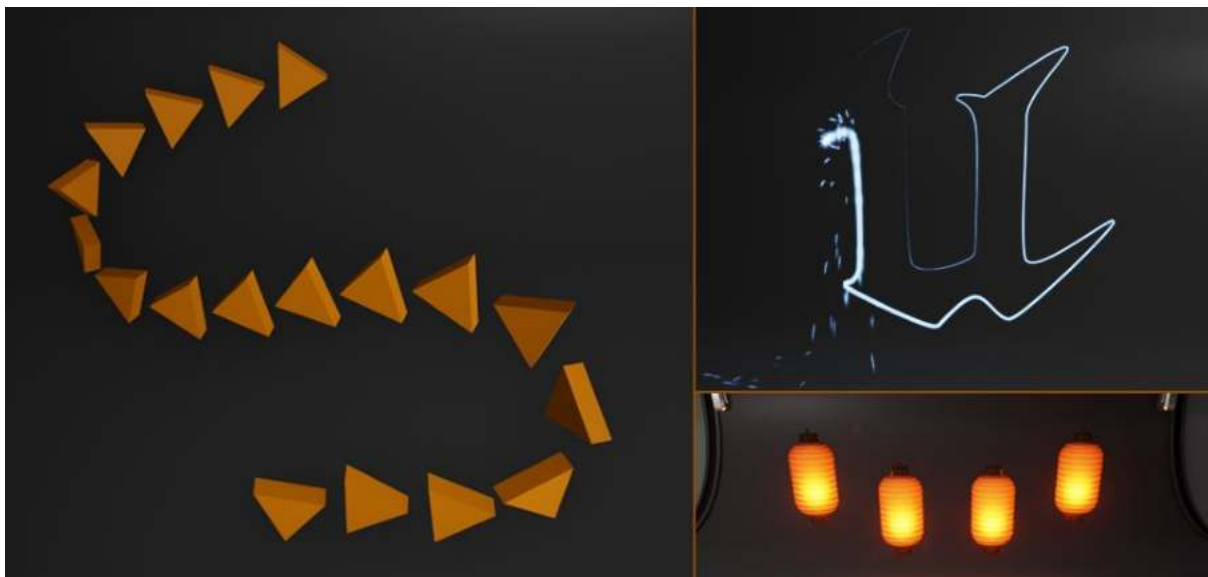


Figura 8: Ejemplo de *splines* de la página web de Unreal Engine.

5. Menús y modos de juego

En el juego solo está presente un modo de juego, donde el jugador tendrá que superar los niveles enfrentándose a unidades controladas por la IA.

5.1. Menús *out-game*

5.1.1. Pantalla de título

Nada más iniciar el juego aparecerá la pantalla que se muestra en la figura 9. Esta pantalla es la primera parte de la pantalla de título donde deberemos pulsar una tecla para acceder a las opciones principales del juego.



Figura 9: Pantalla que aparece nada más iniciar el juego.

Una vez hecho esto pasaremos a la pantalla que se muestra en la figura 10. En ella tendremos las siguientes opciones:

- Play: nos muestra una pantalla nueva donde podremos elegir el nivel que queremos jugar, el modo de juego y los personajes que usaremos en la partida.
- Controls: nos muestra la lista de controles del juego.
- Quit: cierra el juego.

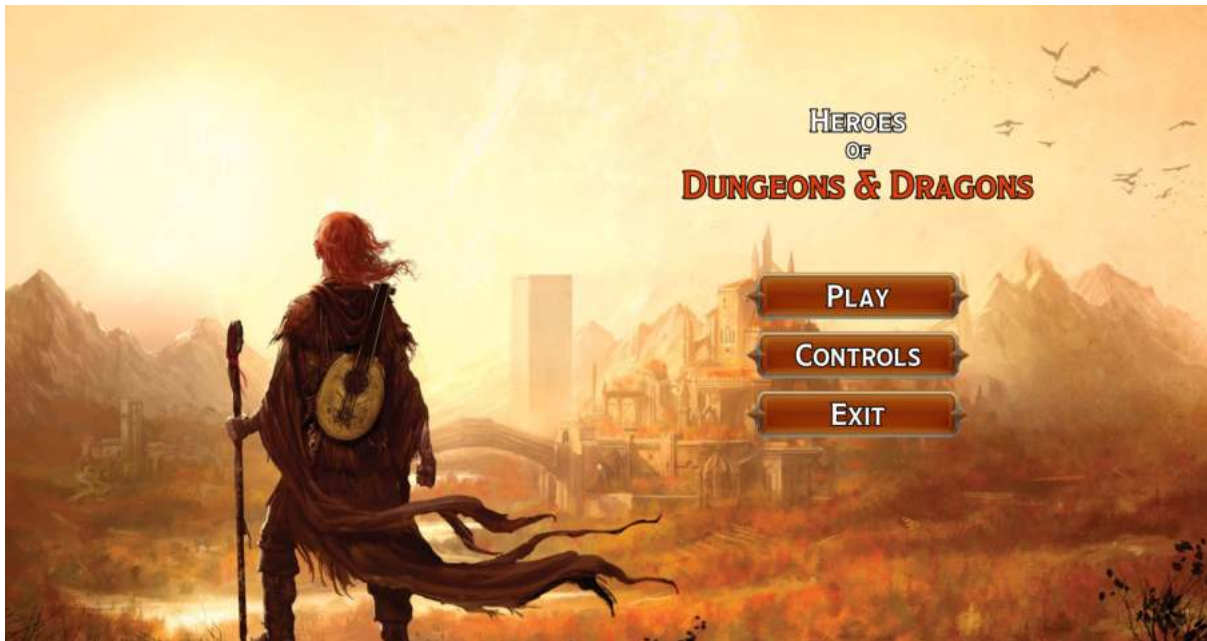


Figura 10: Pantalla de título con las opciones principales del juego.

5.1.2. Pantalla de controles

En la figura 11 tenemos la pantalla de controles del juego. Para controlar el juego solo es necesario usar el ratón, y además se da la posibilidad de quitar información en pantalla cuando no sea necesaria (información de la unidad actual, botón H). El ratón ofrece dos funcionalidades a la hora de controlar el juego:

1. Click izquierdo: esta acción se puede realizar sobre la unidad en turno para elegir que acción debe tomar. También permite usar los menús del juego.
2. Control de cámara: moviendo el puntero del mouse hacia los laterales de la pantalla podemos girar la cámara del jugador con respecto al centro del nivel.



Figura 11: Pantalla con los controles del juego.

5.2. Menús de configuración

En la figura 12 tenemos el menú de configuración de la herramienta. En él se puede seleccionar que nivel que se quiere jugar y el modo de juego (matar a todos, matar al jefe final o alcanzar una zona). Además se muestran los enemigos que intervendrán en la batalla por parte de cada equipo.

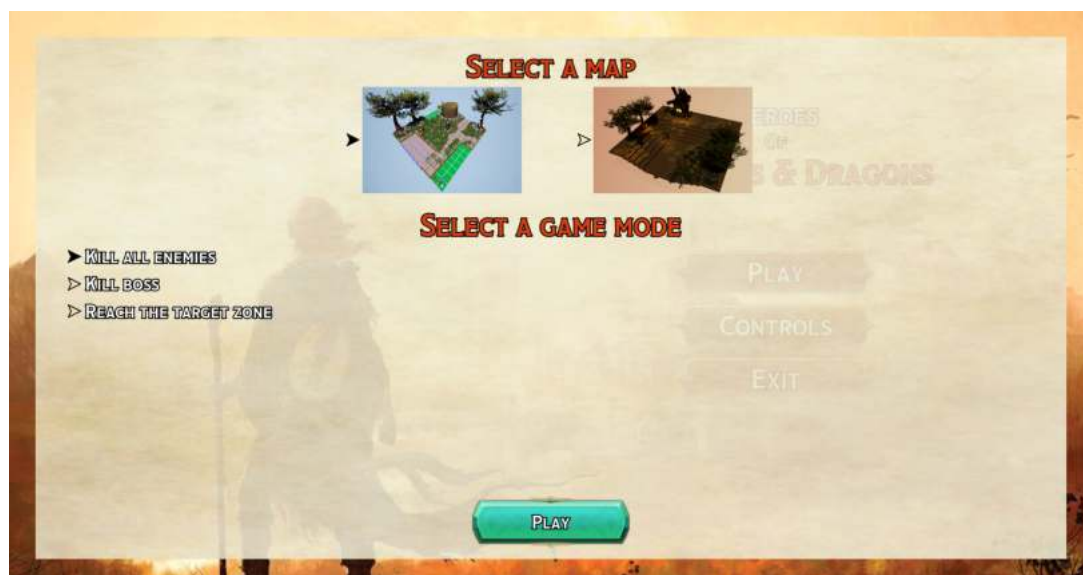


Figura 12: Pantalla de configuración del nivel.

5.3. Interfaz y control

El HUD del usuario está formado por varios componentes. El primero de ellos es una barra horizontal que indica el turno de las unidades que aún no han hecho nada en el turno actual (figura 13). Las unidades del equipo del jugador tienen un color de fondo azul claro mientras que las del equipo rival tienen un color rojo, con el fin de distinguir a quien pertenece cada unidad.

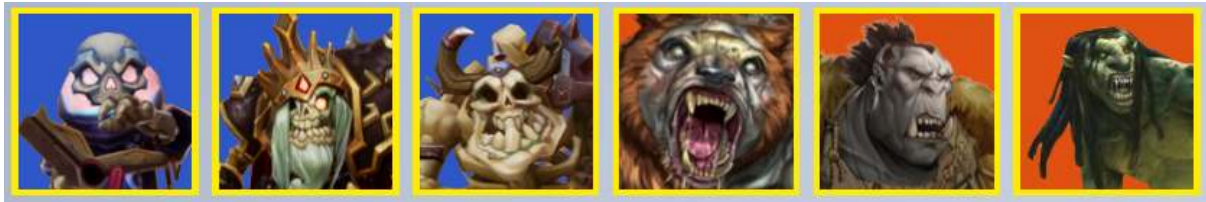


Figura 13: HUD que muestra al jugador el turno de las unidades.

En la figura 14 tenemos parte del HUD que informa al jugador de las estadísticas de la unidad que está en turno. Estas son:

- HP: puntos de salud.
- MP: puntos mágicos o maná.
- XP: experiencia.
- PHY. ATK.: ataque físico.
- PHY. DEF.: defensa física.
- MAG. ATK.: ataque mágico.
- MAG. DEF.: defensa mágica.
- Speed: velocidad de la unidad.
- Lucky: suerte.
- Steps: cantidad de casillas que puede avanzar la unidad como máximo en su turno.
- Height: altura máxima alcanzable por la unidad cuando se está desplazando por el escenario.

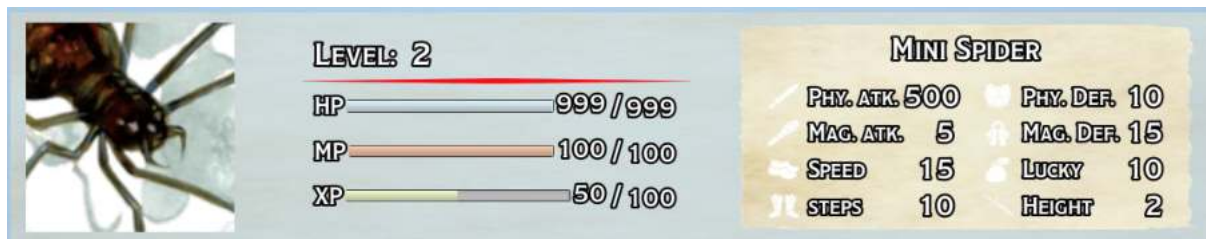


Figura 14: HUD con stats de la unidad en turno.

5.4. Pantallas de fin de juego

A continuación tenemos las pantallas que aparecerán una vez termine el juego. En ambos casos podremos volver al menú inicial o salir.



Figura 15: Pantallas de fin de juego.

6. Jugabilidad

6.1. Mecánica

6.1.1. Turnos

Respecto al progreso de las partidas, como ya se ha comentado anteriormente, es llevado a cabo mediante turnos. Para determinar el orden de los turnos se utiliza el atributo *Speed* de las unidades, de forma que se ordenan de mayor a menor velocidad y ese resultado es el orden en el que realizan las acciones en cada turno. Si durante un turno de juego una unidad es destruida su turno será eliminado de la lista.

Una vez que todas las unidades hayan agotado su turno se volverán a ordenar todas ellas en función su velocidad y con ello se creará el nuevo orden de turnos. Además de esta forma también se añade una cierta aleatoriedad en caso de que 2 o más unidades empaten a velocidad, ya que el algoritmo no siempre devolverá el mismo resultado.

6.1.2. Movimientos

La mecánica del movimiento dependerá del atributo *Steps* de cada unidad, de forma que cada casilla atravesada contará como un paso dado. Sólo se contarán como adyacentes las casillas situadas en los 4 ejes cardinales, y por tanto llegar a las casillas situadas en las diagonales supone una distancia mínima de 2 pasos.

Además hay que tener en cuenta el atributo *Height*, que determinará las alturas que puede superar. Más concretamente este parámetro limitará cuál es la diferencia máxima de altura entre casillas que puede superar. De esta forma se pueden establecer unidades que “salten” y otras que solamente “suban escalones”.

En el momento que el jugador seleccione la opción de moverse o la IA busque dónde puede desplazarse se calcularán las posibles casillas de destino mediante el algoritmo A* explicado anteriormente. Cabe destacar que la casilla donde esté situada la unidad no contará como casilla alcanzable (es decir, moverse implica cambiar de casilla) y tampoco podrá ocupar ni pasar por casillas ocupadas por otra unidad.

6.1.3. Ataques

Respecto a la mecánica relativa a los ataques cabe destacar dos apartados:

- **Alcance:** para determinar el alcance de cada unidad se utiliza la estructura de datos "*DT_Attacks*", donde se almacenan los rangos de ataque de cada tipo de unidad. De esta forma las posiciones almacenadas se sumarán a la posición actual de la unidad y esto da lugar a las casillas alcanzables para el ataque. Por último tras haber calculado todas las casillas a las que se puede atacar se filtran las casillas en las que haya unidades que sean del mismo equipo, ya que no se contempla el *fuego amigo*. Hecho este proceso, se obtendrán todas las casillas que contengan enemigos.

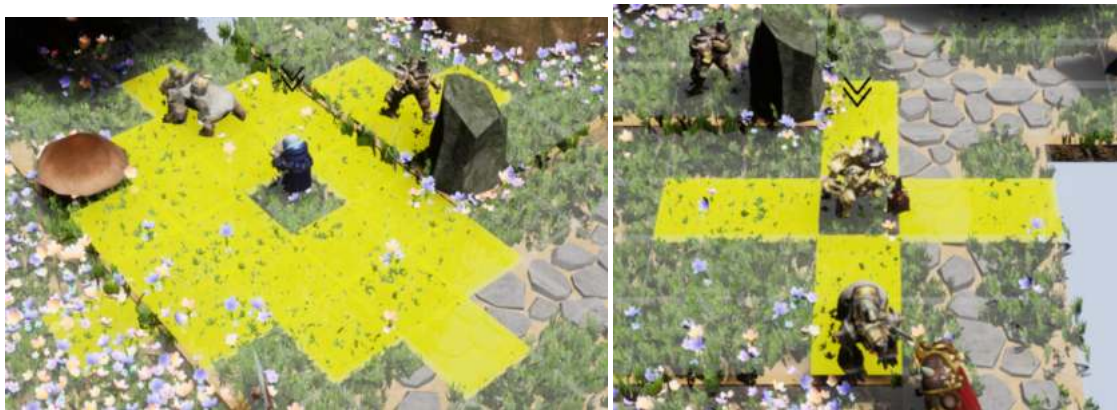


Figura 16: Ejemplos de rangos de ataque

- **Daño:** para el cálculo del daño se tienen en cuenta los diferentes atributos de ataque y defensa de la unidad atacante y atacada, respectivamente. Además se ha añadido una mecánica que potencia o debilita los ataques en función de la dirección a la que mire la unidad atacada. Más concretamente, los ataques frontales se verán reducidos a la mitad, los ataques laterales tendrán un multiplicador de 1.5 y los acometidos desde la retaguardia se potenciarán al doble.

$$Damage = \frac{Attack}{Defense/2} SideFactor \quad (1)$$

6.1.4. Inteligencia Artificial

En lo referente al comportamiento de los enemigos controlados por la IA su comportamiento se divide en 3 posibles ramas. Como precondition para que la IA decida actuar está que sea el turno de dicha unidad. Los 3 posibles comportamientos son los siguientes:

- **Ataque:** si la única opción de la que dispone es atacar, buscará entre sus casillas de ataque si hay alguna unidad enemiga, y en caso afirmativo atacará a dicha unidad.
- **Ataque y movimiento:** en caso de tener ambas acciones disponibles la IA intentará atacar a algún enemigo que tenga al alcance. Concretamente si antes de moverse ya tiene alguno dentro del rango, directamente le atacará. En este caso además intentará moverse a continuación a una “posición segura”, es decir, que esté fuera del rango de ataque de cualquier enemigo. En caso de no encontrar ninguna segura, escogerá una al azar.
Por el contrario, si al comenzar su turno no tiene enemigos al alcance intentará buscar enemigos a los que pueda atacar después de moverse en este turno. En caso de existir alguna casilla alcanzable desde la que pueda atacar a un enemigo procederá a moverse y a continuación a atacar. Si no encuentra ninguna posición que cumpla estas condiciones descartará esta estrategia y seguirá con la siguiente.
- **Movimiento:** si la única acción que le queda disponible es la de moverse, intentará hacerlo a alguna casilla que considere “no peligrosa”, es decir, que no entre dentro del rango de ataque de alguna unidad enemiga.

6.2. Dinámica

6.2.1. Victoria y derrota

Actualmente el juego presenta 3 modos de juego diferentes:

- **“Kill all”:** en este modo de juego el objetivo será eliminar a todos los enemigos del bando rival, por lo tanto la condición de victoria será erradicar a los enemigos.
- **“Boss”:** en este caso el objetivo será eliminar al enemigo principal, considerado el *jefe* del nivel. Esta unidad tendrá una etiqueta de “Boss” y en caso de ser eliminada se considerará victoria para el jugador.
- **“Reach zone”:** en dicho modo habrá una serie de casillas marcadas como “destino” que el jugador deberá tratar de alcanzar, de forma que el jugador ganará en cuanto consiga que cualquiera de sus unidades llegue a alguna de las casillas.

En último lugar cabe destacar que en todos los casos la condición para perder la partida será que el jugador se quede sin unidades vivas.

6.2.2. M.E.T.A.

La “*Most Efficient Tactic Available*” hace referencia al término inglés utilizado para referirse a la mejor estrategia disponible a seguir para ganar la partida. En este caso dicha estrategia dependerá del modo de juego, ya que en cada uno el objetivo es claramente diferente. A continuación se presentan los principales factores que el jugador deberá tener en cuenta según el modo de juego:

- **“Kill all”**: en este modo el jugador deberá tener en cuenta que debe maximizar la velocidad a la que destruye a las unidades enemigas, ya que la supervivencia de sus unidades es vital. En este caso esperamos que el jugador intente llevar a sus unidades cerca las unas de las otras para concentrar los ataques en un enemigo tras otro.
- **“Boss”**: en este caso, al contrario que ocurre en el modo anterior, la única unidad enemiga que importa destruir es el *Jefe* y por tanto no es recomendable concentrarse en eliminar todas y cada una de las demás unidades. Sin embargo hay que tener en cuenta que en caso de estar concentrado en atacar al *Jefe* puede darse la situación de verse rodeado por unidades enemigas y verse claramente superado. Por tanto, esperamos que el jugador busque un equilibrio entre concentrarse en el *Jefe* pero no dejarse superar por las demás unidades.
- **“Reach zone”**: en dicho modo eliminar unidades enemigas no aporta ningún progreso a la hora de ganar, ya que lo único que importa es reducir la distancia hacia el objetivo. Sin embargo, y de manera análoga al caso anterior, descuidar demasiado la eliminación de enemigos puede traer consigo un avasallamiento de las unidades del jugador y llevar a la derrota. Por esto, el jugador deberá encontrar un cierto equilibrio entre avanzar hacia su destino y no dejar que los enemigos se concentren a su alrededor.

6.3. Estética

Dado que el principal desarrollo de este proyecto ha sido la herramienta, comentaremos brevemente la estética utilizada en los niveles creados para mostrar sus capacidades. Sin embargo no comentaremos nada al respecto de narrativa o historia ya que sólo trata de ser una demo técnica.

Para la ambientación de los 2 niveles creados hemos utilizado una estética naturalista centrada en la vegetación y construcciones humanas rurales simples, como pueden ser molinos o farolillos. Además cada nivel presenta un ambiente de iluminación opuesto para transmitir las sensaciones de ver con mayor facilidad el campo de batalla (ambiente diurno con alta iluminación) o la sensación de acorralamiento al no ver con tanta claridad a todos los enemigos (ambiente nocturno).

7. Contenido

7.1. Niveles

Para la presentación de la herramienta hemos preparado dos niveles los cuales se pueden ver en las figuras 17 y 18. Ambos niveles plantean una localización similar (misma estética y assets) pero cambiando el terreno de los mismos, el decorado del mapa y la geometría del nivel.



Figura 17: Nivel con ambientación de bosque de día.



Figura 18: Nivel con ambientación de valle de noche.

7.2. Personajes

Para la demo de la herramienta hemos preparado 6 personajes, 3 para el usuario y otros 4 que son controlados por la IA.

7.3. Araña

La araña es uno de los enemigos controlados por la IA. Al tratarse de un ser de aspecto pequeño y frágil (figura 19) posee pocos puntos de vida y unas defensas reducidas, sin embargo es capaz de moverse por gran parte del terreno sin dificultad y su capacidad de salto es elevada, como se muestra a continuación:

Salud	Maná	A. físico	D. Física	A. mágico	D. Mágica
120	100	60	15	5	15
Suerte	Velocidad	Movimiento	Salto	Crítico	
10	7	6	3	0.2%	

Cuadro 1: Tabla con las estadísticas de la araña



Figura 19: *Concept art* de la araña.

7.4. Orco

El orco es también controlado por la IA. En este caso se trata de la típica unidad a *melee* equilibrada y todoterreno. Por ello no destaca especialmente en ningún atributo:

Salud	Maná	A. físico	D. Física	A. mágico	D. Mágica
150	50	70	60	55	45
Suerte	Velocidad	Movimiento	Salto	Crítico	
2	4	3	2	0.1%	

Cuadro 2: Tabla con las estadísticas del orco



Figura 20: *Concept art* del orco.

7.5. Oso

El oso es la tercera unidad controlada por la IA. En este caso estamos ante una unidad más robusta que las anteriores, la cual se centra en el apartado físico. Debido a ello su maná y tanto su ataque como su defensa mágica son bastante reducidas, aunque se compensa bastante bien con una gran cantidad de salud.

Salud	Maná	A. físico	D. Física	A. mágico	D. Mágica
200	25	80	60	20	10
Suerte	Velocidad	Movimiento	Salto	Crítico	
20	6	4	2	0.1 %	

Cuadro 3: Tabla con las estadísticas del oso



Figura 21: Concept art del oso.

7.6. Trol

El Trol es el último enemigo controlado por la IA y es la unidad que aparecerá como jefe cuando se seleccione el modo de juego pertinente. Debido a ello es la unidad que mas daño hace tanto a nivel físico como mágico, y además posee un nivel de salud elevado y una defensa física alta.

Salud	Maná	A. físico	D. Física	A. mágico	D. Mágica
250	100	90	80	30	10
Suerte	Velocidad	Movimiento	Salto	Crítico	
10	3	3	1	0.1 %	

Cuadro 4: Tabla con las estadísticas del trol



Figura 22: Concept art del trol.

7.7. Mago esqueleto

Esta unidad es controlada por el jugador. Se trata de una unidad de ataque a distancia, por tanto su rango de alcance es mayor respecto al resto de unidades. De esta forma es capaz de alcanzar con sus ataque a cualquier unidad que esté a dos casillas de distancia.

Al ser una unidad con la capacidad de atacar desde lejos posee pocos puntos de vida y defensa física para que esté equilibrado. Además dispone de gran cantidad de maná, ataque mágico y defensa mágica al tratarse de un mago. En su defecto el ataque físico es muy reducido.

Salud	Maná	A. físico	D. Física	A. mágico	D. Mágica
120	200	30	40	60	80
Suerte	Velocidad	Movimiento	Salto	Crítico	
10	10	4	2	0.1 %	

Cuadro 5: Tabla con las estadísticas del mago esqueleto



Figura 23: Concept art del mago esqueleto.

7.8. Esqueleto rey

El esqueleto rey solo es controlado por el jugador. Al igual que el Orco se trata de una unidad todo-terreno que ataca a melle. Sus estadísticas se muestran a continuación y son bastante equilibradas:

Salud	Maná	A. físico	D. Física	A. mágico	D. Mágica
170	100	70	60	30	40
Suerte	Velocidad	Movimiento	Salto	Crítico	
10	9	5	2	0.1 %	

Cuadro 6: Tabla con las estadísticas del esqueleto rey.



Figura 24: *Concept art* del esqueleto rey.

7.9. Esqueleto guerrillero

Esta unidad solo es controlada por el jugador. A diferencia de las demás unidades presentadas hasta ahora el guerrillero cuenta con un ataque a melle de distancia dos, es decir, puede golpear a unidades que se encuentren a dos casillas de distancia en las 4 direcciones. Esta unidad posee unas estadísticas altas semejantes a las del trol con el objetivo de equilibrar el combate.

Salud	Maná	A. físico	D. Física	A. mágico	D. Mágica
220	100	85	65	30	50
Suerte	Velocidad	Movimiento	Salto	Crítico	
10	8	3	1	0.1 %	

Cuadro 7: Tabla con las estadísticas del esqueleto guerrillero.



Figura 25: *Concept art* del esqueleto guerrillero.