Escuela
Politécnica
Superior

# Gestión y manejo de comporatamientos grupales emergentes de entidades en videojuegos

Grado en Ingeniería Multimedia

Trabajo Fin de Grado

Autor:
Borja Pozo Wals
Tutor/es:
Francisco José Gallego Durán

Octubre 2020

Universitat d'Alacant
Universidad de Alicante

# Gestión y manejo de comportamientos grupales de entidades en videojuegos

Subtítulo del TFG

**Autor**
Borja Pozo Wals

**Tutor/es**
Francisco José Gallego Durán
*Departamento de Ciencia de la Computación e Inteligencia Artificial*

Grado en Ingeniería Multimedia

Escuela Politécnica Superior

Universitat d'Alacant
Universidad de Alicante

ALICANTE, 13 de diciembre de 2020

# Resumen

título es una herramienta para simular escenarios bélicos como los que podemos encontrar en juegos del género Real Time Strategy (RTS) como son *'They are Billions'* o la saga *'Age of Empires'*, desarrollado completamente en C++ para Personal Computer (PC).

La simulación se compone de un escenario, una unidad controlada por el jugador, una serie de unidades que conforman el ejercito bajo sus ordenes y por último, tenemos una serie de objetivos que abatir. Una vez conseguido el objetivo terminará el juego pudiendo el jugador elegir entre repetir el nivel o volver al menú principal.

# Justificación y objetivos

A lo largo de la carrera son muchas las asignaturas que requieren y desarrollan habilidades relacionadas con la programación en diversos lenguajes y usos, pero no es hasta el tercer año que se me presenta la oportunidad de desarrollar un videojuego completo.

Durante la asignatura de 'Fundamentos de los Videojuegos' tuve por primera vez la experiencia de enfrentarme al desafio que es crear un videojuego, y fue en ese momento cuando me di cuenta de que a lo que me quería dedicar es a desarrollar juegos de forma profesional. A lo largo del cuarto curso junto a los demás integrantes de *'Sunlight Studio'* desarrollamos *'Cyborgeddon'*, y fue en este momento que terminé de decidir que es a lo quiero dedicarme en el futuro.

A lo largo de mi vida he jugando una considerable cantidad de juegos entre los cuales se puede apreciar una inclinación por los juegos de aventura y exploración, los Rol Play Game (RPG) y los RTS o de estrategia en general, es por esto que me hace especial ilusión desarrollar un juego de uno de estos géneros.
Teniendo en cuenta esto la elección de desarrollar un juego del tipo RTS para la realización del Trabajo Final de Grado (TFG) se debe a que según mi criterio es el que tiene mayor potencial para ayudarme a mejorar mis habilidades como programador, ya que se basa en unas mecánicas con las que no he trabajado con anterioridad.

Una vez dicho esto, los objetivos planteados para el proyecto son los siguientes:

- Aumentar mis conocimientos de C++
- Comprender mejor el funcionamiento del PC
- Aprender nuevas técnicas de Inteligencia Artificial (IA)
- Profundizar en la arquitectura Entity-Component-System (ECS)
- Desarrollar un producto mediante el cual mostrar mis habilidades

# Agradecimientos

*A mi madre Gabriela,*
*por el increíble esfuerzo que realiza cada día por mi.*

*A mi hermana Raquel,*
*por haber sido y ser un faro para mi.*

*A mi buen compañero Jorge Espinosa,*
*por nuestra convivencia estos años.*

*A mi querido amigo Guillermo y su familia,*
*por el cariño recibido incluso a pesar de la distancia.*

*Ni los reyes ni los gobernantes llevan el cetro,*
*sino los que saben mandar*

Sócrates.

*Para saber hablar es preciso saber escuchar*

Plutarco.

# Índice general

# Índice de figuras

# Índice de tablas

# Índice de Listados

# Índice de Acrónimos

A lo largo del documento serán utilizadas una serie de abreviaturas con el fin de hacer más cómoda su lectura. Todos los términos están indicados a continuación:

- **TFG:** Trabajo Final de Grado
- **PC:** Personal Computer
- **RTS:** Real Time Strategy
- **RPG:** Rol Play Game
- **IA:** Inteligencia Artificial
- **NPC:** Non-Player Character
- **AoE:** Age of Empires
- **TaB:** They are Billions
- **ECS:** Entity-Component-System
- **GDD:** Game Design Document

# 1 Introducción

Este proyecto trata sobre el desarrollo de un simulador de batallas para Personal Computer (PC) desarrollado en *Linux*, escrito en C++ y haciendo uso de librerias escritas en C como TinyPTC add reference.

El simulador, título, se trata de una demo del género Real Time Strategy (RTS) compuesta por un escenario a través del cual tendremos que comandar y luchar junto a nuestro ejercito con el fin de abatir al ejercito rival. El escenario estará compuesto por estructuras y obstáculos que deberemos sortear para alcanzar nuestro objetivo.

A nivel técnico el proyecto cuenta con el desafio de desarrollar una Inteligencia Artificial (IA) grupal para los Non-Player Character (NPC) basada en el uso de técnicas como los *Steering behaviours* y el *Flocking* con el fin de emular un comportamiento coordinado entre las distintas entidades. El objetivo es crear una versión inicial sencilla mediante la cual poder profundizar en los conceptos en los cuales se basan las mencionadas técnicas con el fin de desarrollar una versión más compleja y específica para nuestro proyecto de forma que se adapte de la mejor forma posible a nuestras necesidades.

Por otro lado, como objetivo adicional fuera de la demo como tal es realizar un *port* a *Windows* con el fin de poder mostrar los resultados en ambos sistemas operativos.

# 2 Marco teórico

## 2.1. Referentes

Cuando uno busca desarrollar un producto es una buena práctica el investigar que se ha realizado previamente en entregas similares, ver que ténicas se han utilizado y que problemas se han encontrado y como los solventaron. También es interesante ver que mecánicas se han ido consevando y que innovaciones se han ido introduciendo con el paso del tiempo.

A continuación vamos a detallar aspectos de interés encontrados en diversas entregas las cuales nos servirán de referentes.

### 2.1.1. Age of Empires II: The Age of Kings

En primer lugar encontramos el juego *'Age of Empires II: The Age of Kings'*, en este juego encontraremos una serie de campañas a través de las cuales encarnaremos a personajes históricos como 'Juana de Arco' o 'William Wallace' y los acompañaremos en sus conquistas y batallas más icónicas. Además de grandes batallas encontraremos el deber de gestionar nuestra facción, este trabajo nos requerirá recoger rescursos a lo largo del mapa mientras desarrollamos nuestras ciudades y unidades.

El desarrollo tecnológico se divide en cuatro etapas históricas: Alta Edad Media, de los castillos, Feudal e Imperial. Cada una de estas etapas trae consigo una serie de unidades, edificaciones e investigaciones nuevas las cuales nos proporcionaran escenarios más complejos a nivel estratégico, para pasar de una etapa a otra no es necesario completar al 100 % las opciones desbloqueadas, solo unos requisitos mínimos.

El juego cuenta con un total de 35 civilizaciones o facciones, todas tienen acceso al mismo árbol tecnológico pero cuentan con ligeras diferencias, cada una cuenta con una o dos tecnologías únicas y características especiales en una unidad militar[1].

Todas la unidades militares[2] de las que podemos disponer cuentan con una serie características que las hacen más o menos fuertes en función del objetivo al que se enfrenten, podemos encontrar el ejemplo de las máquinas de asedio las cuales son más fuertes contra

[1] Normalmente alguna cualidad mejorada en relación a las demás como puede ser la velocidad o daño.
[2] Lista de unidades: `https://ageofempires.fandom.com/wiki/Units_(Age_of_Empires_II)`.

estructuras pero más débiles contra infanteria, o las unidades a caballo que son fuertes contra arqueros e infanteria pero débiles frente a lanceros.
Este tipo de mécanicas dotan al juego de una importante componente táctica en el manejo de las unidades que debemos dominar si queremos completar los diferentes niveles de forma satisfactoria 2.2.

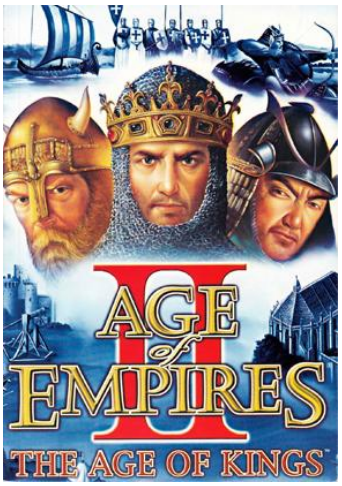


Figura 2.1: Carátula del juego original.

En lo referente a edificaciones[3] podemos encontrar distintos tipos según su aporte al jugador, el edifício printipal es el 'Ayuntamiento' el cual nos permitirá crear colonos para que trabajen recogiendo recursos y/o ayudando en la construcción de nuevas estructuras, en segundo lugar podemos encontrar edifícios como el 'Cuartel' o el 'Campo de tiro' los cuales nos permiten crear unidades militares, por otro lado tenemos edificios como la 'Herreria' la cual nos permitirá realizar investigaciones y crear unidades de asedio.

Por último podemos encontrar estructuras de temática religiosa como el 'Monasterio', otras destinadas a aumentar la productividad en las explotación de recursos naturales como el 'Molino' y defensivas como las 'Murallas'.

En el género RTS hay una tendencia al uso de una perspectiva isométrica con cámara fija la cual solo podemos modificar haciendo zoom o desplazandola por el entorno, esto seguramente se deba a que es la configuración que mejor nos permite observar el mapa y todas las unidades que hay en el. Además, el hecho de que sea fija nos libra de la problemática de ir ajustando la cámara según la parte del mapeado que estemos observando, ya que, seguramente el mapeado y los elementos visuales del juego también hayan sido diseñados teniendo esto en cuenta.

Como podemos ver en el artículo de Pritchard sobre el desarrollo de Age of Empires (AoE) 2, la IA del juego esta desarrollada completamente con un lenguaje de *scripting*

[3]Lista de edificios: `https://ageofempires.fandom.com/wiki/Buildings_(Age_of_Empires_II)`.

Figura 2.2: Ciudad siendo asediada.

desarrollado por ellos 2.1.1, de esta forma el código consiste en una serie de constantes y variables definidas desde un inicio y mediante condicionales realizar las acciones y cambios pertinentes. Gracias a un usuario de *'GitHub'* podemos ver en las imagenes 2.1.1 el código usado en el juego, la extensión es de más de 10.000 líneas de código.

```
(defrule
    (CONDITION 1)
    (CONDITION 2)
=>
    (ACTION 1)
    (ACTION 2)
)
```

Figura 2.3: Descripción Estructura.

```
(defrule
    (can-train villager)
=>
    (train villager)
)
```

Figura 2.4: Ejemplo condición.

**Fuente:** guía de scripting para IA de AoE 2 de redmechanic.

Además de estos *scripts* el juego utiliza un sistema de *pathfinding* en dos niveles, el primero de los algoritmos calcula la ruta a seguir por las unidades sin tener el cuenta los elementos triviales como las demás unidades, mientras que la segunda pasada calcula el camino en tramos específicos con el fin de esquivar otras unidades, edificaciones, etc...

En lo referente al movimiento de las unidades, tanto por la forma en la que está programada la IA como por la sensación que dan al jugar, es posible que se trate de un *"Goto"* clásico y sencillo para evitar realizar muchos cálculos.
Además, con el tiempo el juego se ha convertido en un objetivo para *"Speedrunners"* y con una escena competitiva que se basa en ser el más rápido jugando[4] por lo que una reacción instantánea de las unidades es un factor importante para estos jugadores.

[4]En los enfrentamientos de alto rango se llegan a medir las acciónes por minuto (APM).

```
(defrule
        (taunt-detected any-ally 33)
        (game-time > 300)
=>
        (acknowledge-taunt this-any-ally 33)
        (chat-to-allies-using-id 22157)
        (set-goal train-civ-goal 1)
)
```

Figura 2.5: Ejemplo condición.

```
;Goals
(defconst increase-town-size-goal 1); this increases town size
(defconst attack-goal 2)
(defconst strategy-goal 3)
(defconst unit-goal 4)
(defconst train-civ-goal 5);1=train villagers, !=1 no villager
(defconst control-goal 6); 6 = allow to be shot, 7 = shot, als
(defconst anti-cavalry-threat-goal 7)
```

Figura 2.6: Ejemplo constantes.

**Fuente:** repositorio de *'GitHub'* del usuario Andygmb.

### 2.1.2. The Are Billions

En segundo lugar podemos encontrar el +juego *'They are Billions (TaB)'* el cual reutiliza una serie de características propias del género y como pueden ser la perspectiva isométrica pero sin dejar de innovar introduciendo nuevas mecánicas y/o formas de plantear la jugabilidad.


Figura 2.7: Imágen promocional del juego.

En *'TaB'* podemos encontrar funciones interesantes como la pausa táctica la cual nos permitirá visualizar detenidamente el estado del mapa sin tener que preocuparnos por no estar atendiendo algunos posibles eventos como ataques a nuestras tropas por parte del enemigo. En juegos anteriores como los *'AoE'* es fácil encontrarnos en la situación de tener trabajadores en la ciudad sin hacer tareas un rato y no poder mirar cuales son e ir pensando su ocupación siguiente por estar atrapado en refriegas con otros jugadores, con este tipo de mecánicas estas situaciones se solventan en mayor o menor medida y permiten al jugador tomarse el tiempo que necesite para pensar las acciones que quiere realizar.

Otro aspecto llamativo en el podemos fijarnos es en la aparición de solamente dos facciones. Por un lado encontramos 'El Nuevo Imperio', la facción del jugador, la cual representa una serie de colonias que tendremos que desarrollar a lo largo de los niveles.

En el otro lado encontramos las infinitas hordas de zombis que tratarán de exterminar a la raza humana.

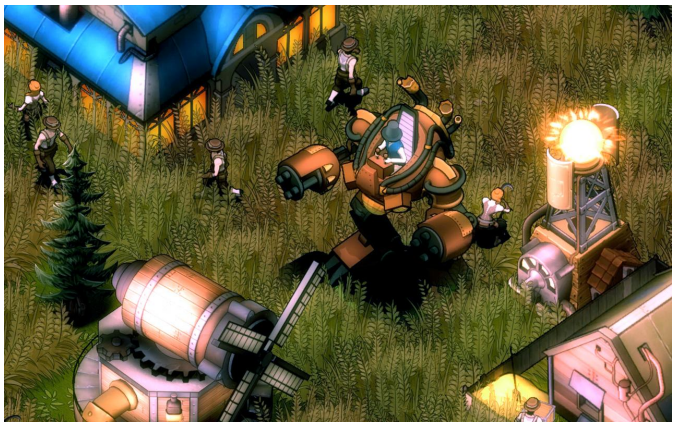

Figura 2.8: Ejemplo de unidad mecánica del imperio y estética del juego

A diferencia de en *'AoE'* que todas la facciones poseen las mismas unidades [5], en *'TaB'* cada facción dispondrá de unidades únicas con acciones propias. Como podemos ver en la imágen 2.8 las tropas del imperio se basan en el uso de maquinaria y armas de fuego para repeler las hordas enemigas, por otro lado los zombis contarán con diversas caraterísticas físicas mejoradas conforme sean de mayor "nivel" puediendo encontrar algunos más rápidos, o más fuertes o más grandes y resistentes al daño. 2.9

Figura 2.9: Infectado gigante

Como último punto a destacar de la entrega tenemos la introducción del un modo de juego de supervivencia donde a lo largo del tiempo irán apareciendo oleadas de zombis

[5] Algunos valores pueden variar según bonos de facción

donde cada vez aparecen más enemigos y más poderosos de forma infinita, haciendo así que la partida termine cuando el jugador deje de aguantar la ofensiva enemiga.

## 2.2. Técnicas de inteligencia articificial

Como ya se ha mencionado anteriormente en la introducción 1 de este Trabajo Final de Grado (TFG) el grueso del desarrollo y uno de los objetivos más importantes del proyecto recaen en el desarrollo de una IA haciendo uso de las técnicas de *'Flocking'* y *'Steering behaviors'*.

Para ello nos basaremos principalmente en las explicaciones y ejemplos que podemos encontrar en el libro (Millington, 2009, ch. 3) donde de forma extensa y detallada se nos introduce en la teoría relacionada a los algoritmos y la forma en la que estos se estructuran e interactuan entre ellos. Para dar un poco de contexto sobre el tema resumiremos brevemente las ideas que se nos presentan a lo largo del capitulo dedicado a estas técnicas.

Los *'Steering behaviors'* pueden ser entendidos como una serie de algoritmos destinados a guiar la forma en la cual los NPC se desplazan por el escenario y/o interactuan con los distintos elementos que puedan encontrase en la escena. Siguen una filosofía de crear movimientos complejos a base de una combinación de movimientos y/o acciones simples, un ejemplo común puede ser la acción de perseguir a un objetivo mientras se sortean obstáculos en el proceso.
En este caso no tendríamos una función llamada *"persigue-enemigo-mientras-esquivas()"* y esta encargarse de todo el trabajo, sino que, tendremos el cálculo de la velocidad y dirección necesarias para alcanzar el objetivo, la compropación para saber si hay algún tipo de obstáculo por el camino y la rectificación de la trayectoria en caso de haberlos cada uno por su lado y es la resultante de todos los pasos la que defina el movimiento final.

Esta forma de estructurar y formar actividades complejas en base a acciones más simples nos permite reutilizar y jugar con los diferentes comportamientos permitiéndonos crear con ellos un amplio espectro de resultados.

En lo referente al *'Flocking'* podemos observar como en esencia es lo mismo que los *'Steering behaviors'* pero añadiendo factores y/o componentes grupales, el origen del modelo lo podemos encontrar en las publicaciones de Reynolds (1986) donde se nos introduce el concepto de *"Boid"* como entidad generica que simula su comportamiento bajo este algoritmo. Además, se nos introducen los tres comportamientos básicos en los que se basa la técnica para generar el movimiento emergente, que son:

La **separación** 2.10 que cada *Boid* mantendrá entre las demás entidades en su vencidad, con esto evitaremos solapamientos y respetar el espacio y movimiento de las demás entidades.

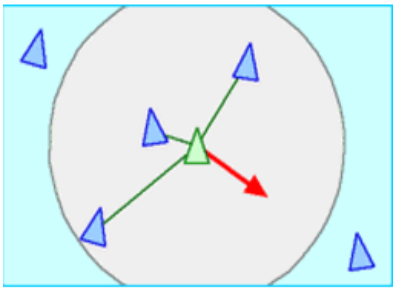

Figura 2.10: Separation behavior

Por otro lado podemos encontrar el **alineamiento** 2.11 de la dirección del movimento propio con las de las entidades más cercanas, de esta forma conseguimos un movimiento armónico entre los *boids* y produciremos una sensación de coordinación entre ellos.

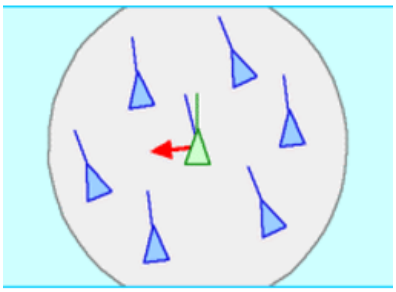

Figura 2.11: Alignment behavior

Por último encontramos la **cohesión** 2.12 la cual se encargará de mantener a las entidades cercanas juntas para crear esa sensación de grupo que buscamos con el algoritmo.

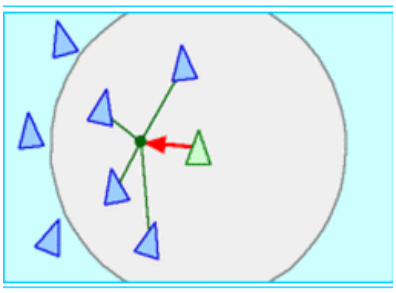

Figura 2.12: Cohesion behavior

Por otro lado, podemos ver en el articulo de 'Raynolds' como a lo largo de los años se ha ido modificando y ampliando el algoritmo con el fin de añadir variaciones en el comportamiento y/o introducir más factores influyentes en la decisión de los *Boids* como puede ser el olor de determinada entidad/es y/o escenario.

Esto sin duda es gracias a la versatílidad que nos proporciona el uso de los *'Steering behaviors'* y jugar con la importancia de las distintas componentes a la hora de hacer la toma de decisiones.

# 3 Documento de Diseño del Juego (GDD)

## 3.1. Características

- **Título:** N/A.
- **Plataforma:** PC (Windows y Linux).
- **Género:** Real Time Strategy (RTS).
- **Idioma:** Inglés.
- **Clasificación:** PEGI 7[1]

## 3.2. Descripción general

El juego que se pretende desarrollar consiste en un RTS haciendo hincapie en el apartado de navegación por el mapa y el combate, dejando así la gestión de recursos y el desarrollo de una civilización o facción a un lado debido a las limitaciones de tiempo y personal. A lo largo de los distintos niveles, el jugador deberá hacer frente a distintos desafios como pueden ser: trasladar todas sus unidades de un punto del mapa a otro mientras se evita la muerte de algún personaje importante y/o eliminar a todas las entidades enemigas del escenario.

## 3.3. Mecánicas

La finalidad del juego es la de completar los distintos niveles de forma satisfactoria, esto sucederá ya sea cuando eliminemos a todas las unidades enemigas desplegadas a lo largo del nivel o alcancemos el punto indicado. Como herramienta para alcanzar este objetivo dispondremos de un ejercito a nuestro mando el cual deberemos gestionar de forma efectiva para sortear los obstáculos y desafios propuestos.

El jugador será capaz de seleccionar con el ratón las unidades exactas sobre las que quiere lanzar la acción y seleccionar o deseleccionar todas la unidades a su disposición

[1] Web de la asociación https://pegi.info

mediante atajos de teclado en cualquier momento. Las acciones disponibles para su uso serán las de desplazar las unidades, hacer que se queden a la espera y realizar un ataque sobre las unidades rivales que se indiquen. Como puede verse en la imágen 3.1

El jugador perderá cuando todas sus unidades mueran.

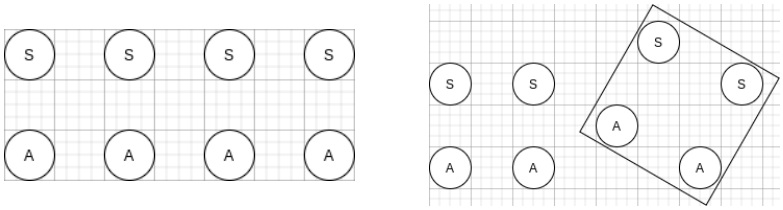


Figura 3.1: MockUp selección.

## 3.4. Unidades

Entre las unidades podemos encontrar diferentes arquetipos con carácteristicas propias que nos permitiran crear variedad en las posibles soluciones a la hora de superar el nivel.

Los distintos tipos son los siguientes:

- **Soldado:** es la unidad más básica que podemos encontrar en el campo de guerra, esta armado con una espada y posee estadisticas bajas.
- **Arquero:** va equipado con arco y flechas para atacar a distancia a sus rivales, tiene menos resistencia que los soldados por lo que tendremos que protegerlos para asegurar su supervivencia.

## 3.5. Controles

A la hora de jugar tendremos una serie de teclas asignadas a las acciones que el jugador puede realizar cuando interactúe con el juego. Para enumerarlas dividiremos las acciones en dos grupos, dependiendo de si son para navegar por los menús o si representan acciones durante el *gameplay*.

Las teclas para navegar por los menús son las siguientes:

- **Enter:** mediante esta tecla podremos avanzar por los menús una vez estemos sobre la opción deseada.
- **Retroceso:** mediante esta otra podremos ir hacía atrás por los menús.
- **Escape:** esta nos permitirá salir de la ejecución del programa.

Las asignadas para jugar son las siguientes:

- **Click derecho:** mediante esta tecla del ratón podremos seleccionar la unidad sobre la que queremos trabajar, si mantenemos pulsado y arrastramos por el escenario podremos seleccionar varias unidades cercanas. Si no se selecciona ninguna unidad, se deshará la selección actual.
- **Click izquierdo:** con esta otra podremos seleccionar donde queremos desplazar a las unidades seleccionadas y/o atacar a otras unidades.
- **R:** usaremos esta tecla para deseleccionar todas las unidades desde el teclado.
- **T:** la usaremos para seleccionar a todas las unidades, independientemente de donde se encuentren.

## 3.6. Pantallas

Al ejecutar el programa la primera pantalla que aparecerá será la de inicio 3.2. Se compone del título del juego, la fecha de lanzamiento, el nombre del desarrollador y un mensaje que nos indica que tenemos que pulsar al tecla *entre* para continuar, esta pantalla se mantendrá hasta que el jugador presione dicha tecla.

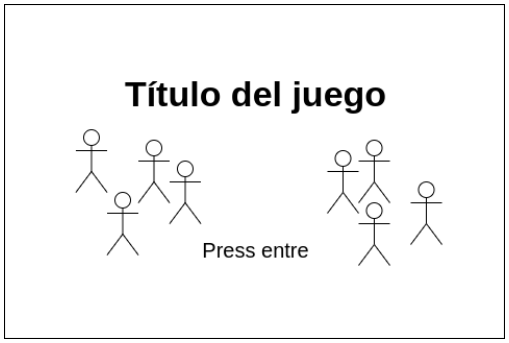


Figura 3.2: MockUp pantalla de inicio.

Una vez pulsado el botón indicado se procederá a cargar el juego mientras se muestra una barra que indica el progreso de cargado 3.3.

A continuación de la carga pasaremos directamente al escenario donde se desarrollará el *gameplay* y nos mantendremos en esta pantalla hasta que termine el juego, ya sea por victoria o derrota del jugador 3.4.

El final de la partida nos trae dos posibles escenarios, la victoria y la derrota. Para cada uno saldrá su respectivo mensaje 3.5 3.6 y pasado un momento se nos mandará automáticamente a la pantalla de inicio 3.2.

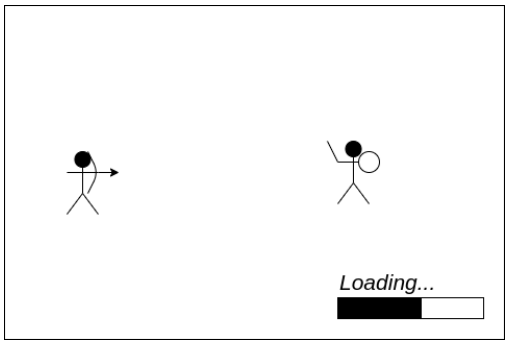


Figura 3.3: MockUp pantalla de carga.

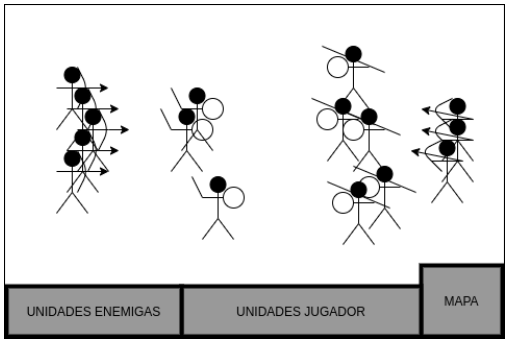


Figura 3.4: MockUp escenario juego.

Como última pantalla con la que el jugador podrá interactuar es la del menú de pausa 3.7 en el cual se le dará la opción de salir o de volver a la partida, mientras esta pantalla este activa la acción en el juego se paralizará hasta que el jugador decida.

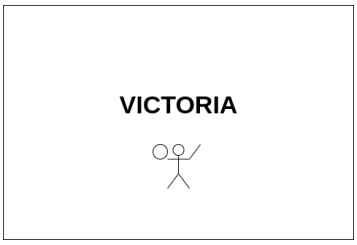


Figura 3.5: MockUp victoria.

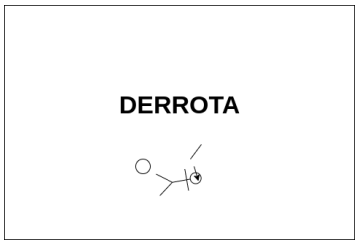


Figura 3.6: MockUp derrota.

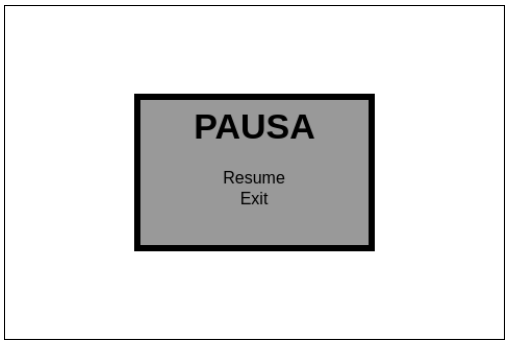


Figura 3.7: MockUp menú de pausa.

## 3.7. Estados del juego

Una vez mostradas todas las posibles pantallas con las que podrá interactuar el jugador es interesante dibujar un diagrama de flujo que plame sus conexiones y posibilidades con el fin de crear una representación gráfica que sirva como esquema global.

Dicho esquema podemos encontrarlo en la figura 3.8.

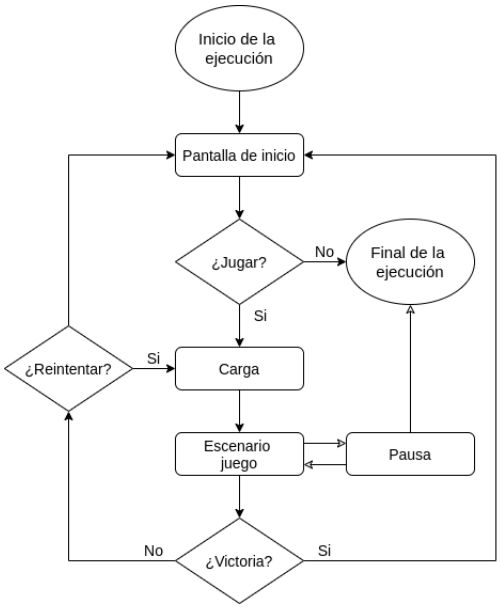


Figura 3.8: MockUp escenario juego.

## 3.8. Escenarios

A lo largo de los niveles la intención es que nos encontremos con distintos biomas como pueden ser praderas, bosque o desiertos como los que podemos encontrar en AoE II como hemos visto anteriormente en la imágen 2.2 o en las 3.9 y 3.10.

Figura 3.9: Ejemplo escenario con vegetación.

Figura 3.10: Ejemplo escenario desértico.

Además de imitar el estilo artístico el juego se desarrollará empleando una cámara aerea fija manteniendo una perspectiva isométrica que nos permitirá visualizar desde un punto elevado una gran porción del escenario, esto lo haremos con el fin de dar al jugador la posibilidad de conocer lo máximo posible el territorio cercano posible y a la vez le libramos de tener que estar preocuparse de situar la cámara para cada momento. Al mantener una posición fija podemos situar todos los elementos en la escena de forma que no obstaculicen la visión o molesten al jugador.

# 4 Metodología y fases del producto

## 4.1. Metodología

La metodología llevada a cabo durante el proyecto es una creación propia basada en caraterísticas copiadas de metodologías ágiles como el **'Scrum'** y adaptadas a la forma de trabajar propia. La idea es estructurar todo el proyecto en diversas iteraciones y durante estas marcarse objetivos y/o tareas con el fin de añadir funcionalidades completas.

El uso de **iteraciones** para dividir la carga de trabajo y agrupar tareas nos ayudará a conseguir acotar la duración de algunas tareas y marcarnos objetivos a corto/medio plazo. Además de separar por funciones el proyecto, las tareas que conllevan su desarrollo también deberemos analizarlas e intentar reducir su tamaño y carga lo máximo posible, esto permite obtener una sensación de éxito de forma rápida lo cual ayuda a mantener una moral y motivación altas.

Para poner un ejemplo de como elaborar esta separación por subtareas, el objetivo que vamos a usar es "dibujar elementos por pantalla". Como estamos en un proyecto básado en Entity-Component-System (ECS) rápidamente vemos que vamos a requerir mínimo un componente de dibujado *(RenderCmp)* y un "RenderSystem" el cual se encargará de recoger todos los elementos visuales y hacer las llamadas necesarias para dibujarlos.
En caso de no tener una librería de dibujado, otra tarea podría ser el buscar una que sea capaz de realizar el trabajo requerio y/o implementar una propia [1].

Para ayudar a la planificación del trabajo usaremos la herramienta 'Trello' 4.1 [2] la cual nos permitirá crear diversas listas según el ámbito o el estado de desarrollo de las distintas tareas que contendrán, cada tarea se ve representada con una tarjeta la cual puede tener una serie de subtareas asociadas dentro. Esto nos permitirá tener un registro de todas las labores que quedan por hacer en cada iteración y cuales ya estan termiandas completamente, además podremos conservar las listas de las tareas realizadas en iteraciones anteriores para futuras revisiones.

Otra herramienta de control que usaremos durante el desarrollo será 'Toggl' 4.1 [3] la cual nos permitirá llevar a cabo un registro de las horas que dedicamos en cada tarea y agrupar tareas en función del campo de estudio o parte del desarrollor al que pertecen.

[1] Esto supondría una serie de tareas en base a la cantidad de funcionalidades que se requieran.
[2] Página de 'Trello': https://trello.com/es
[3] Página de 'Toggl': https://toggl.com


Figura 4.1: Logos de Trello y Toggl.

## 4.2. Iteraciones

En esta sección se comentarán brevemente los objetivos planteados y las tareas que se han llevado a cabo en cada iteración adjuntando el tablero de 'Trello' asociado.

### 4.2.1. Iteración 0

Esta iteración tenía como proposito terminar de definir la idea para el proyecto, ya que, todavía era un poco difusa la imagen del producto final que se quería desarrollar. Para ir entrando en una dinámica productiva e ir definiendo lo que se quería hacer, comenzamos por iniciar el desarrollo de un propotipo a la vez que preparabamos un poco los materiales relacionados con la memoria, como puede ser la lectura de las directrices y/o la revisión de la plantilla de LaTeX 4.2.

Al final de esta iteración el prototipo contaba con una versión básica del bucle principal del juego donde creamos una serie de sistemas, *managers* y entidades además del dibujado y movimiento[4] de estas.

### 4.2.2. Iteración 1

A lo largo de esta iteración seguimos añadiendo sistemas al prototipo como puede ser el de *Input* el cual nos permitirá interactuar con el producto, además de herramietas como el mapeado del teclado o el tipo de dato en coma fija en el cual profundizaremos más tarde.

Por otro lado, se comenzó a trabajar en la IA del juego creando los primeros comportamientos y herramientas para cambiar entre ello. Como veremos más adelante también, aquí caeremos en los primeros errores de concepto a la hora de trabajar con los *'Steering behaviors'*.

Por último, finalizamos la introducción en el uso de LaTeX, *'JabRef'* y la plantilla comenzando así con la redacción de esta memoria 4.3.

[4]En esta versión nos limitamos a un "Goto" a través de 4 puntos.


Figura 4.2: Lista de tareas realizadas en la iteración 0

### 4.2.3. Iteración 2

Durante esta tercera iteración los objetivos eran los de corregir el funcionamiento de los *'Steering behaviors'*, añadir elementos visuales que ayuden al debug de la IA mostrando las componentes del movimiento de las entidades y herramientas para el control del tiempo de ejecución y el tamaño del *'DeltaTime'*.

Como podemos apreciar en la lista 4.4, las tareas de control de la ejecución del programa no fueron realizadas a tiempo, en su lugar se le dió prioridad a mejorar el uso de la coma fija para un código más limpio y crear metodos auxiliares como puede ser el paso de coordenadas continuas a pixeles en pantalla, permitiendo así trabajar a lo largo del programa usando enteros con signo y procesarlos en el sistema de *'Render'*.

En lo referente a la memoria comenzamos la redacción de una versión preliminar del Game Design Document (GDD), Estado del Arte y metodología.


Figura 4.3: Lista de tareas realizadas en la iteración 1

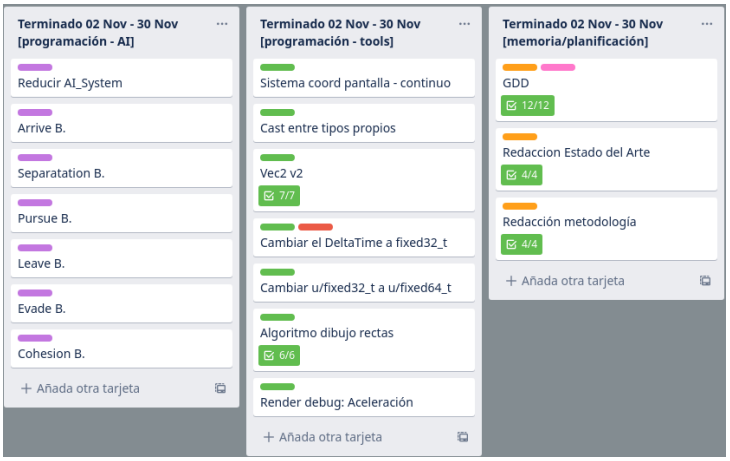


Figura 4.4: Lista de tareas realizadas en la iteración 2

# Bibliografía


Amordron (2019). AI discussion. https://steamcommunity.com/app/644930/discussions/0/1649917420753004999/.

Andygmb (2014). Age of empires II AI Scripting. https://gist.github.com/Andygmb/1e3a6d9d444b2dfa8c40.

Francis, B. (2020). Rebuilding a classic in Age of Empires II: Definitive Edition. https://www.gamasutra.com/view/news/358215/Rebuilding_a_classic_in_Age_of_Empires_II_Definitive_Edition.php.

Millington, I. (2009). *Artificial intelligence for games*. Morgan Kaufmann/Elsevier, Burlington, MA.

Numantian (2019). They are Billions. https://store.steampowered.com/app/644930/They_Are_Billions/.

Pikachunet (2018). Informal level editor. https://www.reddit.com/r/TheyAreBillions/comments/9zvffg/example_of_number_of_zombies_per_wave_800/.

Pritchard, M. (2000). Postmortem: Ensemble Studio's Age of Empires II: Age of Kings. https://www.gamasutra.com/view/feature/131844/postmortem_ensemble_studios_age_.php?page=1.

redmechanic (2017). A guide to scripting your own AI for AoEII. https://steamcommunity.com/sharedfiles/filedetails/?id=1238296169.

Reynolds, C. (1986). Boids: Background and update. http://www.red3d.com/cwr/boids/.

Wikipedia (2020). Age of Empire II. https://en.wikipedia.org/wiki/Age_of_Empires_II.