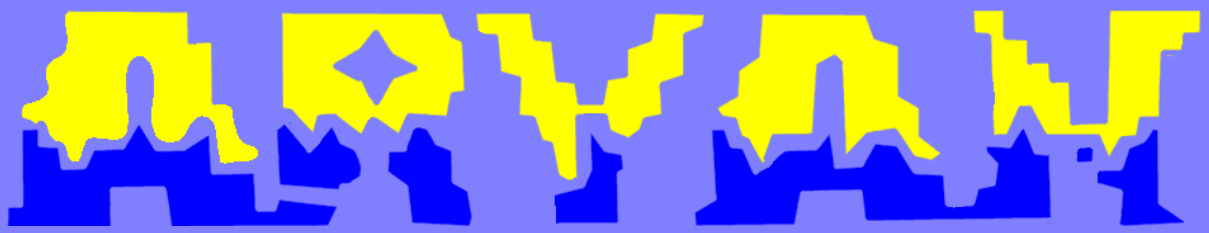
# ASÍ SE HIZO
# HOW IT WAS MADE



Desarrolladores:
Mari Cruz Miranda Torres
Jose A Parron Serna
Juan José Richarte Valera

# ÍNDICE

# 1. Explicación del desarrollo

El juego de Aryan es un proyecto prototipo donde se quieren probar las mecánicas básicas y la jugabilidad de los juegos de peleas. Ha sido la primera vez que el equipo ha desarrollado un juego en ensamblador, pero contiene las bases de un mini juego que posteriormente se utilizarán en un proyecto para el ABP de la carrera de ingeniería multimedia en la UA
La idea era desarrollar un videojuego donde el jugador se dedicase a ir por distintos mapas eliminando enemigos. Conforme el jugador gana experiencia este debe ir superando niveles de mayor dificultad hasta llegar al final de los niveles.
En el juego se pueden observar distintos tipos de enemigos. Con esto se pretende evitar la monotonía mientras que se aumenta la dificultad.

Ya que se desconocía el lenguaje y es uno de los primeros juegos desarrollados por el equipo, fue difícil encontrar mecánicas y características para los enemigos que fuesen adecuadas. Pese a contar con personas expertas y con años de experiencia para consultar, se tuvo margen para probar tanto las capacidades de programación como de planificación del equipo dando como resultado el juego presente.

La parte más difícil fue probamente encontrar mecánicas simples y útiles que se pudiesen usar frecuentemente sin cansar.
El equipo no solo tuvo que aprender un lenguaje desconocido, plantear y desarrollar el juego durante 2 meses, sino que también tuvo que aprender un elemento esencial para todo buen programador: la gestión de la memoria.
Gracias a este proyecto se adquirieron conocimientos y experiencia cuyo valor no se puede tasar.

# 2. Problemas

Aparte de los problemas intrínsecos que trae ensamblador y Amstrad consigo, véase: que nada te avise de cuando sobrescribes en memoria, no saber cuándo tus punteros no apuntan a donde deberían y cuando te confundes operando en hexadecimal.
La mayor dificultad fue la falta de tiempo. Se tuvo que programar contrarreloj provocando que el estrés de la fecha de entrega llevase a priorizar en ocasiones el resultado a la calidad.
La falta de conocimiento en diseño de juegos llevo a tener que plantear y replantear en varias ocasiones los enemigos, debido a que plantear un enemigo no es lo mismo que desarrollarlo y colocarlo en el contexto del juego. Teniendo que recular en varias ocasiones.

El desconocimiento total de un lenguaje tan alejado del lenguaje humano, ha provocado una cantidad incontable de errores bastante simples a primera vista, pero no tanto a la hora de ver una pantalla llena de palabras sin sentido para cualquier conocedor de ensamblado.

La gestión de tiempo fue un gran error del equipo. Se intentó gestionar el tiempo lo mejor posible y pese a que se cometieron varios errores durante el desarrollo el resultado final se obtuvo con mucho esfuerzo.
Finalmente, otro problema, fue la necesidad de plantear todo el juego de una forma distinta a la que estamos acostumbrados. Había que replantearse la forma más óptima y sencilla de con unos pocos registros y muchos bytes sacar el mejor partido a todos los elementos.

# 3. Anécdotas

Como anécdotas más personales por parte un proyecto de este tamaño genera una cantidad de roces entre los miembros del equipo y una cantidad de emociones en ocasiones difíciles de llevar. Pese a que hemos tenido problemas como hacer git push sin comprobar el código de antemano y borrar cosas que no se deberían haber borrado- nada grave- Todos seguimos vivos.

# 4. Lecciones aprendidas

Lecciones aprendidas:

1. No asumas nada demasiado rápido
2. Conoce bien tanto tus limitaciones como las de tus compañeros
3. La gente con más experiencia suele tener razón, así que escúchalos.
4. Ensamblador es muy necesario de conocer pero muy duro.
5. Comunícate bien con tus compañeros.
6. Si piensas que algo te va a llevar una hora en ensamblador, probablemente te lleve 3 si no te fijas y eres principiante.

# 1. Explained Development

Aryan's game is a prototype project where it was wanted to test the basic mechanics and gameplay of fighting games. It has been the first time that our team has developed a game in assembler, but it contains the bases of a minigame that will later be used in a project for the PBL of the multimedia engineering degree at the UA

The idea was to develop a video game where the player dedicated himself to going through different maps eliminating enemies. As the player gains experience, he must overcome levels of greater difficulty until reaching the end of the levels. In the game, you can see different types of enemies. This is intended to avoid monotony while increasing the difficulty

Since the language was unknown and it was one of the first games developed by the team, it was difficult to find suitable mechanics and enemy features. Despite having experts and years of experience to consult, there was room to test both the programming and planning capabilities of the team, resulting in the current game.

The hardest part was probably finding simple and useful mechanics that could be used frequently without getting tired.
The team not only had to learn an unknown language, plan and develop the game for 2 months, but also, had to learn an essential element for any good programmer: memory management.

Thanks to this project, knowledge and experience were acquired whose value cannot be appraised.

# 2. Issues

Apart from the intrinsic problems that assembler and Amstrad bring with it: that nothing warns you when you overwrite memory, not knowing when your pointers do not point where they should and when you get confused operating in hexadecimal.

The biggest difficulty was the lack of time. It had to be scheduled against the clock, causing the stress of the delivery date to sometimes prioritize the result over quality.

The lack of knowledge in game design led to having to pose and rethink enemies several times since thinking of an enemy is not the same as developing it and placing it in the context of the game. Having to back down several times.

The total ignorance of a language so far removed from human language has caused a countless number of errors that are quite simple at first glance, but not so much when seeing a screen full of nonsense words for any assembly connoisseur.

Time management was a big team mistake. We tried to manage the time as best as possible and even though several mistakes were made during the development, the final result was obtained with a lot of effort.
Finally, another problem was the need to approach the entire game in a different way than we are used to. It was necessary to rethink the most optimal and simple way to get the best out of all the elements with a few registers and many bytes.

# 3. Anecdotes

As more personal anecdotes, a project of this size generates several frictions between the team members and many emotions that are sometimes difficult to deal with. Even though we've had issues like git pushing without checking the code beforehand and deleting stuff that shouldn't have been deleted - nothing serious - We're all still alive.

# 4. Lessons learned

Learned lessons:

1. Don't assume anything too quickly
2. Be well aware of both your limitations and those of your peers
3. People with more experience than you are usually right, so listen to them.
4. Assembler is very necessary to know but very hard.
5. Communication with your colleagues is a must
6. If you think something is going to take you an hour in assembler, it will probably take three if you don't pay attention and you're a beginner.
7. The CPCtelera Reference´s Manual it´s not just a decorative web and we could found some information about the code we are implementing.