Amstrad CPC Game | Making of

Alakran Studio

Index

About us

We are fourth-year Multimedia Engineering students from the University of Alicante.

- Abel Martínez Flores
- Nerea Llorens Martínez
- Jorge Zaragoza Garrigós

Currently, we are three members of the Alakran Studio team, a video game development team. The team has split up to create a game for the Amstrad CPC, which will be presented at #CPCRetroDev2023.

Development

We all started by taking a course from our teacher Fran to learn the basics of Z80 assembly language, which we would later use to create the game.

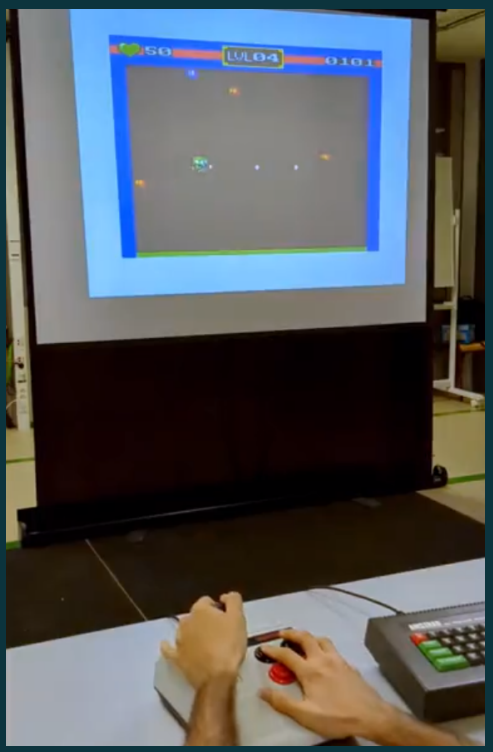
We began by painting a pixel in machine code, but gradually progressed until we faced our first major challenge alone: creating an engine for the **Starfield Effect**, a horizontal star rain.

When we finished creating the first engine, we started working as a team to develop the engine for the Atari console game **'Assault'**. to complete it, and we managed to do it on time.

After finishing the Assault engine, we already had a foundation to start developing our own engine. However, things didn't go as planned.

The code was poorly structured from the beginning and was not scalable, so we had to make a costly decision. We had to redo the Assault engine to make it easier to add mechanics, content, and systems to our game.

We managed to have an initial version of the game fairly quickly so that we could test it on the actual Amstrad CPC during the first week it was brought to class.



For three weeks, we have been developing new mechanics, systems, and managers. In the fourth week of 'content and polishing', we designed all the game sprites, music, and overall improved the game.

The last week has been tough. We let our classmates and teachers try our game to get feedback and improve it as much as possible. Thanks to that, we received a lot of advice and suggestions, especially regarding feedback so that the player knows what is happening while they play.

In the end, we managed to have the game '**Nuke Bug**' finished and published on our Itch.io account.

https://alakranstudio.itch.io/nuke-bug

Technologies

- GIMP: image editor

- Visual Studio Code: code editor

- Arkos Tracker 1: music and sound effect Editor

- WinApe : Amstrad emulator

- Retro VirtualMachine: Amstrad emulator

- CPCTelera: framework for creating Amstrad CPC games

- Clipchamp: online video editor

- VirtualBox: software for running virtual machines

Problems

- When we saw the enemies in action, we realized that some of them were too fast and too difficult for the player to dodge. We tried to make the characters' movement with fixed-point numbers, but we discarded the idea to focus on other aspects of the game. Instead, we solved the problem by updating the position of those enemies fewer times than the rest.

- Flickering has been one of the main issues we faced. To fix it, we limited the number of entities that can appear on the screen at once.

- At one point in the development, based on received feedback, we decided to create hazards that were part of the level. We created a surface that would harm you if you stepped on it, and it remained in the game for a large part of the development process. However, in the end, we had to discard it due to performance issues. Having this entity on the screen made it more likely to experience flickering, so we decided to remove it from the levels.

- Taking feedback into account, we were given the idea to create a boss. Initially, we thought about designing it with large dimensions to make it stand out from the rest of the enemies, but it had a significant impact on the game and caused flickering to occur. That's why we decided to change the idea and draw inspiration from games where there are two simultaneous bosses, making them smaller in size.

Learning

Thanks to the development of Nuke Bug, we have learned the fundamentals of game design and development, how to adapt to an unfamiliar programming language, and how to communicate and make important decisions to deliver the finished product on time.

Learning assembly language has led us to grapple extensively with memory and registers, and has shown us the true importance of space and processing time optimization.

As a team, we have learned how to discuss different points of view regarding aspects of the game, distribute tasks fairly, and make decisions as a team.