Making of “Poisoned Escape”

Carlos Garrido Muñoz

Adrián Sánchez Hernández

Samuel Pernas Muñoz

Computer Engineering

University of Alicante

Index

1. How the game was made

How was the game made? With a lot of effort and dedication on our part, obviously, but also with many hours on our backs, sleepless nights solving bugs to meet our own deadlines and with an illusion that at times seemed to get lost between the lines of our code.

We are students at the university of alicante, so this game for us has a plus of difficulty, as it is a practice that must be evaluated by our teacher, so we could not take risks when designing the game and the mechanics that we have incorporated, as we risked not meeting the deadline.

To develop the game, we had a little more than 5 weeks, not counting the 2 previous weeks of assembler training. A total of 7 weeks to learn the assembly language, to have a good domain of the registers and how they work, and to have the capacity to solve specific problems because of the difficulties of programming at a very low level.

Due to the lack of time, we could not follow a normal and healthy development, we were in crunch time by the first week of developing, and every week we had to finish a part of the gameplay and polish the work of the previous week.

The first week of development we focused on having the main mechanics, since our game is a platform game, the jump is the most important part of development.

The first input system was very rudimentary, it worked in both axes, but a succession of unnecessary ifs.

This same week the first sprite of the game was designed, when we still didn't have well defined neither the setting nor the gameplay, in the end it was discarded, but it helped us during the first 2 weeks of development to see our game grow in many ways.

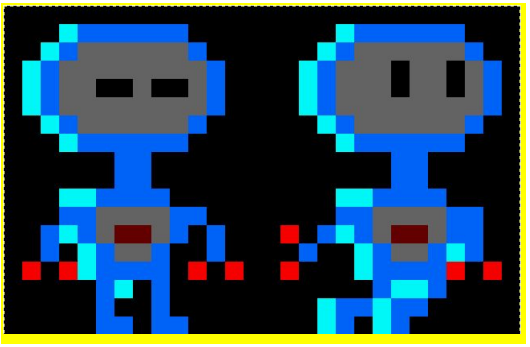

Figure 1: First attempt of our protagonist

As you can see, our first attempt at designing our protagonist made him look like a person in a diving suit. Of course, at this point we had not yet decided how or what the setting of the game was going to be. We hadn't thought about the video mode or the color palette yet, but having a happy face in the first weeks of development really helped.

The second week we polished up the mechanics of the jump, we started to design the final sprite of the game, that is, the character we were going to handle during our adventure.

It was easier for us to name the character in the middle of development, this way we created a bond with him and it even seemed that we cared

more every time we changed something related to the game. And the name we gave her/him was Atheliere, Athe for friends.

We also started to implement the animation system, for this, we created a routine for each available animation (right, left, up-left, etc.), to select the routine in question according to the movement of the character, there is an array with pointer to all available animations of the entity.

Nowadays, there is an array for each kind of enemy and for the main character. Now we will show you some of our finals sprites:

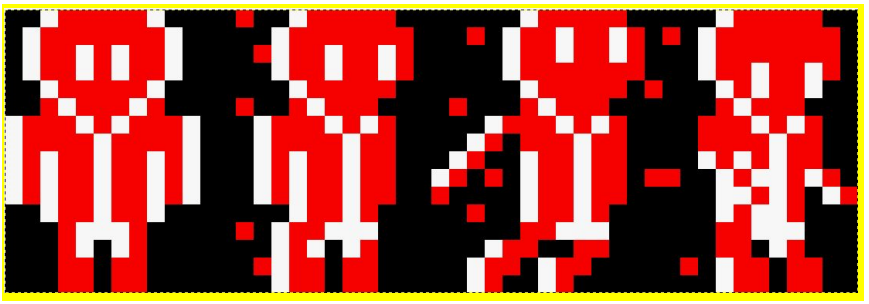

Figure 2: Athe's body sprites

The first was created to better understand the shape of Athe's body. The rest are part of their movements.

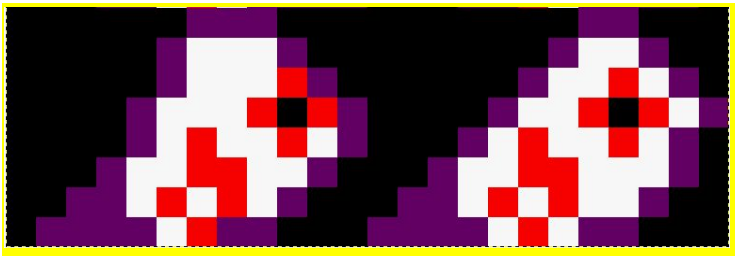

Figure 3: Slug enemy sprite

This is one of the enemies that we will have to face throughout the different levels.

The choice of the animation is made by checking the speeds of the entity in question, according to its speed in the x or y axis should show an animation or another.

Also, in this second week the sprites of the first 2 enemies of the game were designed, it was verified that they worked well with the system of animations and they stayed to make a revision of these the following week.

On the other hand, we started the development of the collisions with enemies system, in both axes, and we started to investigate how the tiles and the tilemap work, but without implementing anything.

The following week, apart from the polishing of the previous week, the creation of sprites of the main character and the implementation of his animations was completed.

Once the collisions with enemies were developed, we started to implement collisions with tiles, creating a very basic tileset with the game palette and some first test tilemaps.

In addition, it was necessary to make a readjustment of the physics implemented in the first weeks, since some bugs appeared when collisions with tiles were detected in the y axis.

The next thing that was done was the correct implementation of an AI to the enemies. At this point we had to decide whether the enemies were going to deploy a patrol or just indicate movement on both axes.

After consulting with our teacher, we agreed not to implement the patrol model in the enemies that we had designed, since it would not be necessary for the types of enemies we want to put in our game. Also, we had to prioritize other parts of the game development and discard unnecessary parts.

In the penultimate week of development, we tried to close the collisions with tiles, but until you do many tests, the bugs do not appear, which have to come out yes or yes.

Similarly with the animation system, since there were certain conditions that did not respect only the two coordinate axes, for example, you have to take into account when you should be standing looking to the left and not to the right. In the end, more than half of the time is dedicated to correcting the bugs that appear.

We also had to change again the physics system of the game, in particular, change the calculation of speeds. We found the problem that the characters were moving at very high speeds, and it was very difficult to have a satisfactory gameplay.

The change in speeds was based on switching from natural numbers to numbers in fix point and a rounding off to facilitate the calculations.

This change meant to retouch in the same way the calculation of the new state of the animation that the character must show, so there was no longer a speed in the X axis of 0, there was also between 0 and 1.

This generated several bugs that were solved by changing the calculation of the new state taking into account the fractional part.

Once we had almost all the mechanics polished, collisions with tiles and enemies well done, and corrected small bugs, we went on to close the game cycle.

For this we designed a load screen, menu screens, developer credits, game controls, next level screen and endgame. We also began to polish the final tileset, so that only the levels of the game remain to be implemented.

This was when we decided the final name of the game, how you can see if you play, everything that kills the player is purple, so we consider that everything purple could be considered a kind of poison that kills everything it touches, except enemies, it does not affect them.

Since the main concept of the game is to dodge all the poison that is distributed in the different levels in addition to your enemies, several names were proposed for the game.

The final decision was between 3 names, the first:

Figure 4: First attempt of cover with the first name

The second name:

Figure 5: Cover with our second name

And the last one:

Figure 6: Cover with our third name

We made the decision by voting, in which the second name, "Poisoned Escape", won.

At this point, the cycle of the game was closed, we continued correcting small bugs and polishing the different systems to have everything well controlled.

The next thing was to implement the music of the game, create levels and continue solving the problems that appear.

To create the music we used the Arkos Tracker 1 tool and we contacted a friend of ours, outside the team and the university, his name is Daniel Carrillo, but his artistic name is Palomo Palomo.

We met with him several times, both to create the song for the start menu and the ingame song. He made several demos according to what the gameplay of the game inspired him, to result in the song we have today.

The last days of development we focused almost exclusively on creating levels, since enough levels had to be implemented so that the game did not fall short and that it had a good difficulty and learning curve.

The problem with this is that none of the development team has studied game design or level design, so combining everyone's ideas and having good levels emerge was the most difficult thing to do in this final stretch of the project.

Once all the levels had been created, we got down to the little details that we most wanted to implement in the game.

The easter egg of the video game "Prince of persia", pause button, button to mute the ambient music, sound in the death animation and when surviving the current level and finally, a death counter in each level, which shows at the end of the gameplay (if you survive all levels), the total number of deaths.

Things that were left out due to lack of time, a counter with the time elapsed since you start the first level until you finish the game, design many more levels, polish certain details in the collision with enemies, etc.

In short, we have invested everything we had, hours, health, effort and desire to get the game we have today, it may not be perfect, perhaps we would change certain things now that we have finished it, but we have no more time than that we have already consumed.

The whole team is very proud of the work done on this project and we hope that the people who decide to play our game enjoy as much as we do, we hope to see your comments and results as soon as possible, greetings from the development team.

2. Technologies

Different programs and technologies have been used for the development of the project. The first and foremost, the video game development library CPCTelera, a framework that includes a low-level game library for C and assembler programmers wanting to create games on Amstrad CPC.

Thanks to it and its guide we have been able to implement our entire game, following the steps indicated in their manuals and learning how the different functions implemented by the library work.

In the development of levels we have used the Tiled tool. Tiled is a 2D level editor that helps you develop the content of your game. Its primary feature is to edit tile maps of various forms, but it also supports free image placement. With a very intuitive interface and a level creation system that works only with a tileset and a couple of clicks, we have been able to develop all the levels of the game.

The tool used in the creation of the music is Arkos Tracker 1, we have used this tool due to the ease of integration provided by the CPCTelera framework with the arkos tracker files.

It works with 3 audio channels, you can create music to be heard in stereo or mono, and control the speed of music playback (50 hz, 25 hz, etc.). Thanks to it and our composer, we have the amazing soundtrack that plays in our game.