Making of

Index

1. Introduction

Utopia is formed by three senior students from Alicante University studying multimedia engineering degree. We are into game development and would like to join videogame industry someday.

We as a team have already developed a game for academic purposes but 1 to 1 soccer is our first assembly game. We didn't have any previous experience in this language, we have learned the language and developed the game in seven weeks.

1 to 1 soccer is a game developed in assembler language using the CPCTelera framework. 1 to 1 soccer is a frenetic multiplayer soccer game with a colorful and childish aesthetics. The gameplay consists of two minutes matches where two players compete to win. The game has multiplayer and single player mode against an AI.

2. Making of

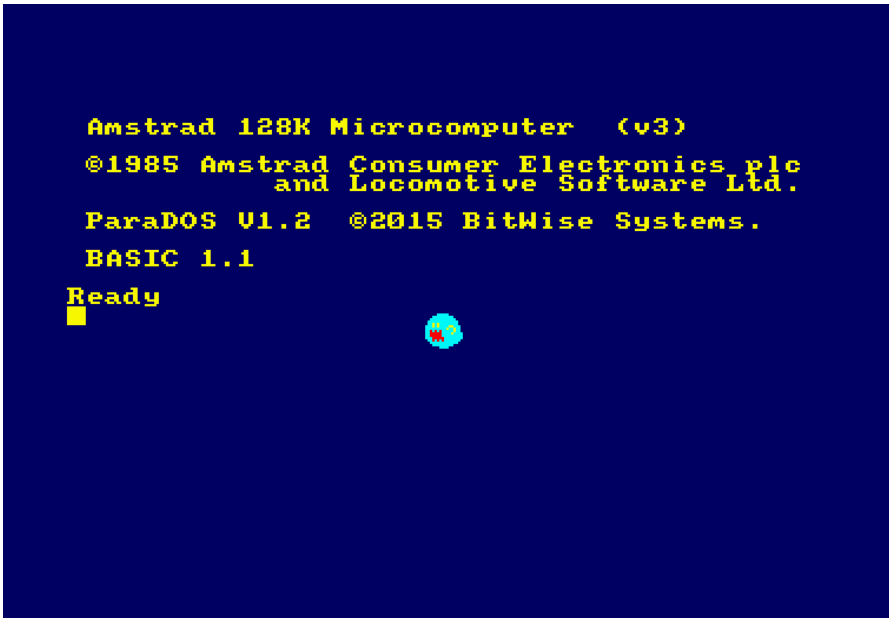
We have followed an iterative process to develop the game. We used the well-known scrum methodology to organize the iterations using Trello. Also we had registered all working hours using Toggl track.

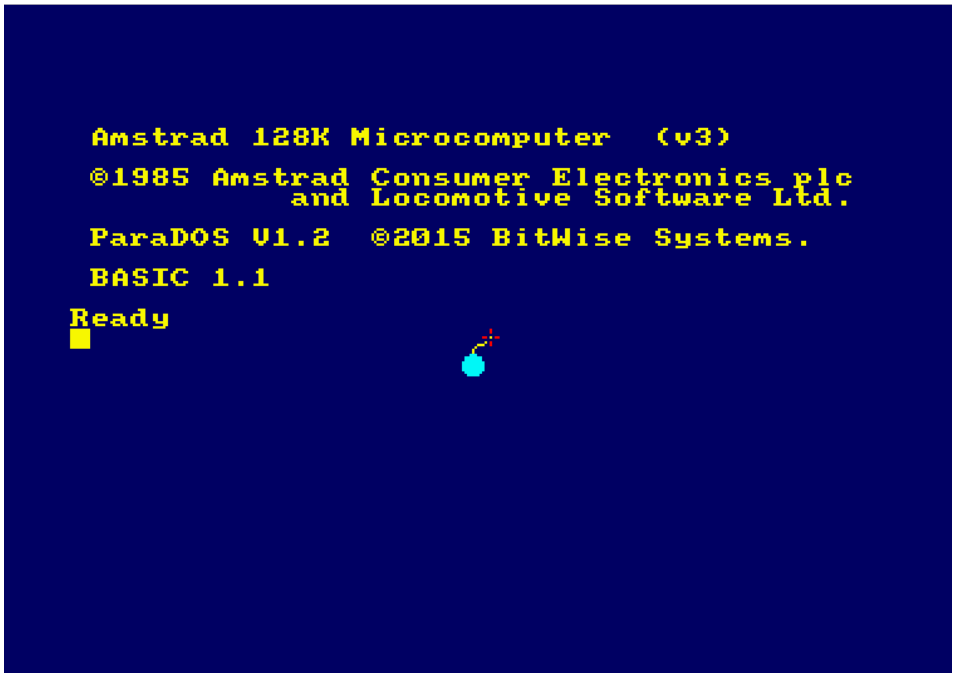
2.1. Learning

The first two weeks we learned machine code as a first approach to assembly language. We followed Profesor Retroman courses to get to learn the language.

Sprites

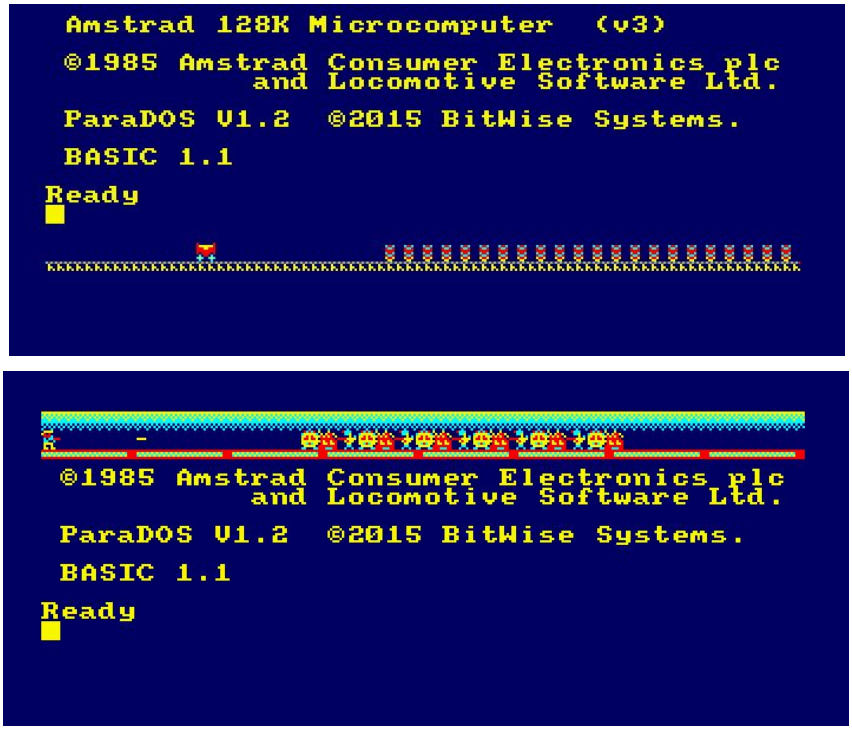
The first week we learned to print sprites using machine code.





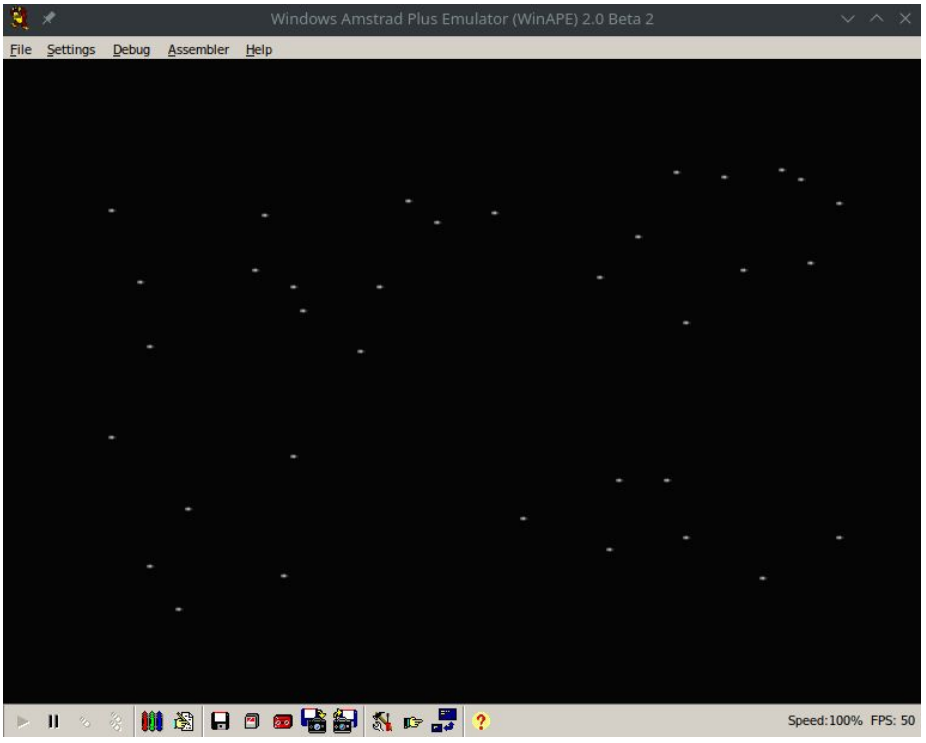
Animations

The first week we also learned to create and control animations in machine language.



Starfield and learning ECS

The second week we started learning assembler and implemented the starfield effect using ECS structure.

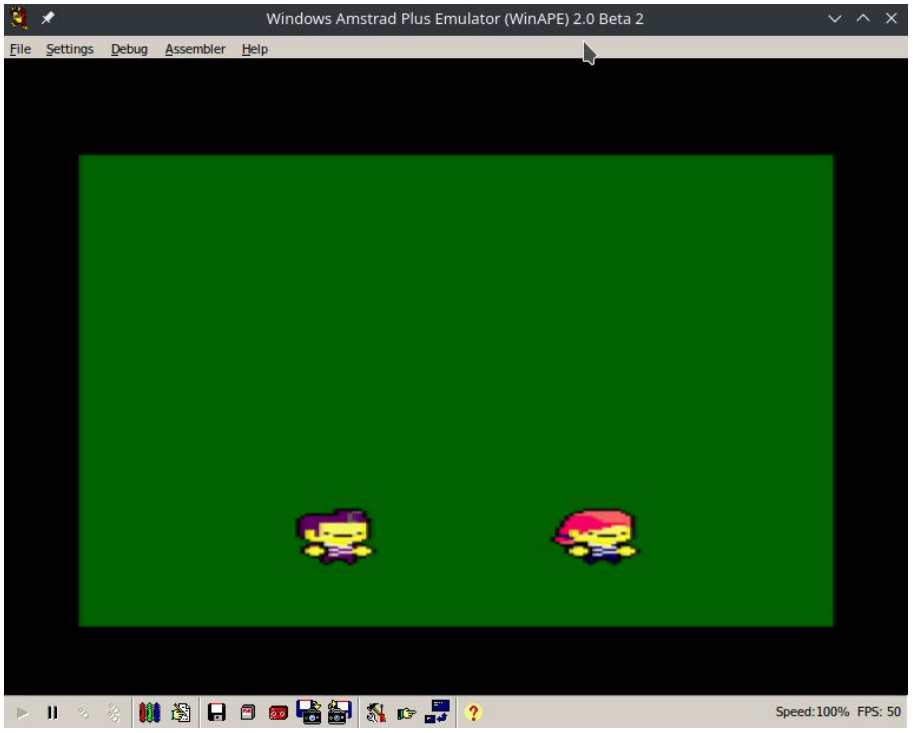


2.2. Technologies

In this part of the document we will explain all the technologies we have implemented in our game.

2.2.1. Physics

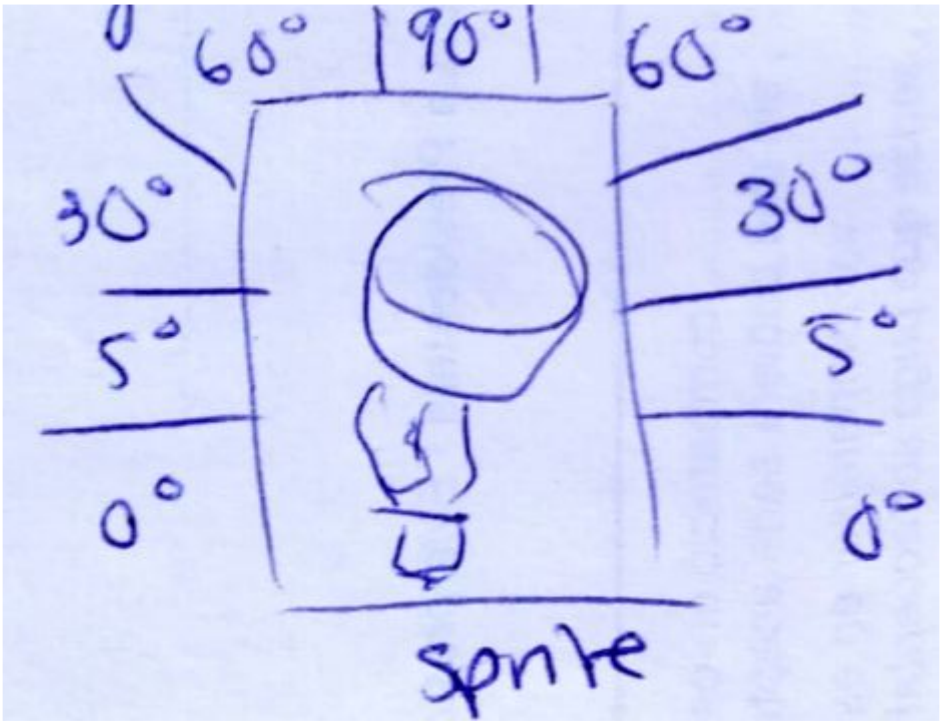
We implemented this in the third week of development. The main mechanics of the game are the player movement: left and right, the jump and the interaction with the ball.



The physics of the ball were made to simulate an elastic ball. This way, when the player hits the ball it bounces each time lower and lower till it stops.

2.2.2. Collision

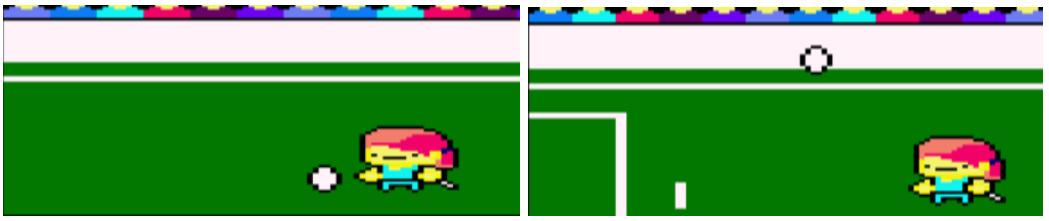
Initially, the collisions of our game consisted of ten sections of the player's body. So depending on which section the ball hits, it will react differently.



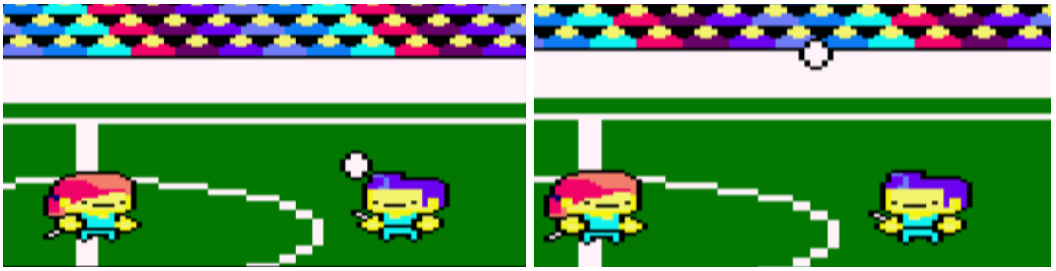
Finally, for reasons of gameplay, we decided to use the following eight sections:

The physics of the ball are made in a way that the direction of the ball depends on where you hit. If you hit with the feet the ball will bounce very low:

If the character hits the ball with the middle side part the ball will bounce more:



If the ball is hit with the top side of a character:



Finally there are two more cases, the upper middle and the lower middle. You can hit the ball with the foot so you will jump on it, this will stop the ball. And you can hit the ball with the center of the head so the ball will jump straight up and down.

2.2.3. AI

In respect of artificial intelligence, It is made in such a way that the reaction time of the AI can be varied by changing only a number of the code. This could make the AI more intelligent or sillier. This was done for the AI because in this way it is not constantly updated.

Mainly, the AI will take the ball as a reference and will try to attack whenever it can, directing the ball to the opponent's goal and jumping when the ball is over him. Besides that, when the ball is going to his goal it will try to jump over the ball to get headed the other way.

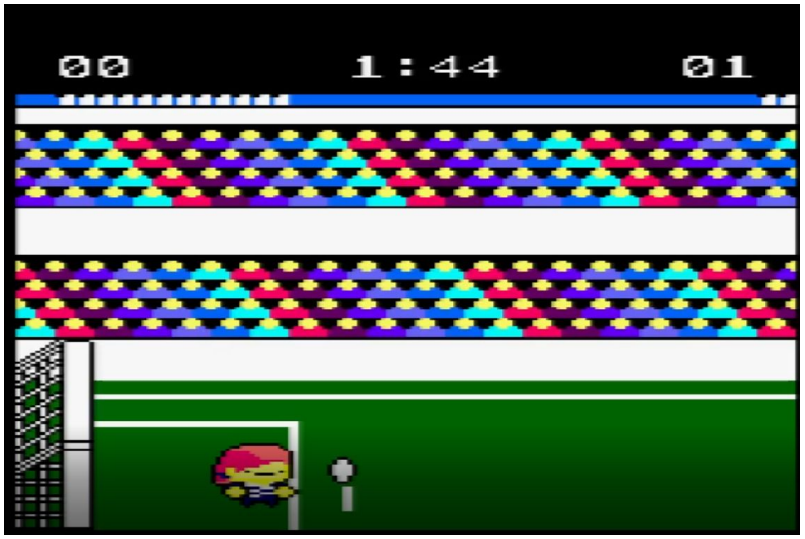
2.2.4. Render and hardware scroll

First of all, we wanted to make the field larger to improve the gameplay. To do so, we have implemented the well-known hardware scroll so that the scroll will be smoother. It moves the camera when the ball crosses the limits of the left or right.

This makes the code faster than software scroll, because we only redraw the new column that appears. The background is made with a tilemap, so we do this by recalculating the tiles that must be painted.

Another thing that we need to change to gain velocity is the draw of sprites. We had to change the function of CPCtelera for pre-calculated sprites. This gives us the opportunity to make a masked sprite. To clean the characters, we calculate the tiles that were modified in order to reconstruct the map.

When the ball is out we replace the camera to fit the goal. We make this by changing the initial value of the scroll.



2.2.5. Interruptions

The interruptions were a critical part of the code. We have several problems with it but it allowed us to add a timer for the game and play music.

We have also used the interruptions to gain time to make the render faster and avoid flickering. This allows us to draw before Vsync occurs.

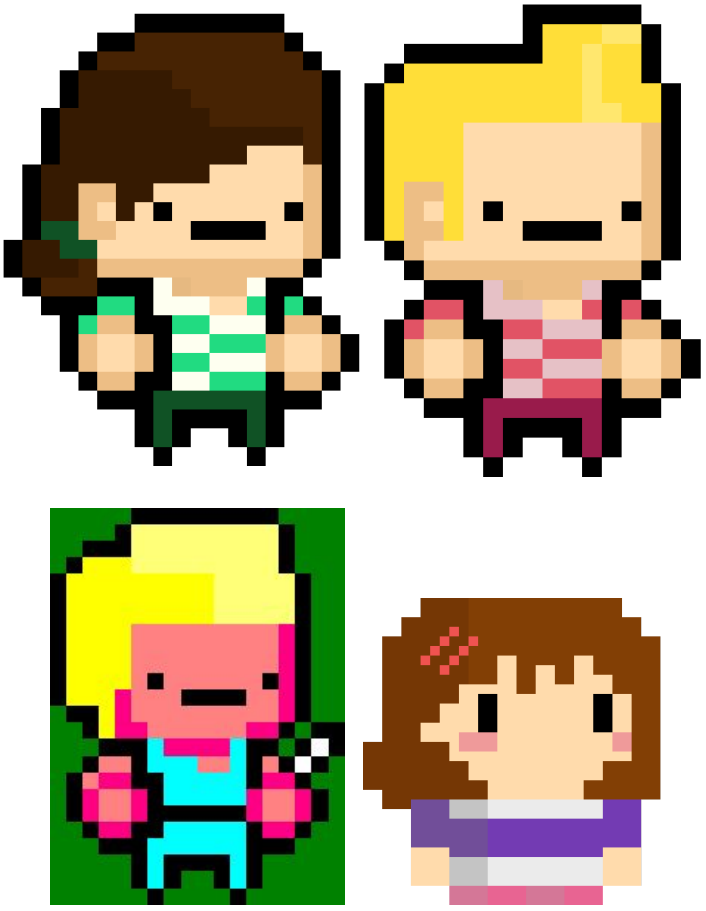
2.2.6. Sprites and music

We had designed all sprites of the game using PhotoShop. All designs are made for amstrad mode 0 which means a maximum resolution of 160x200 pixels and 16 colors.

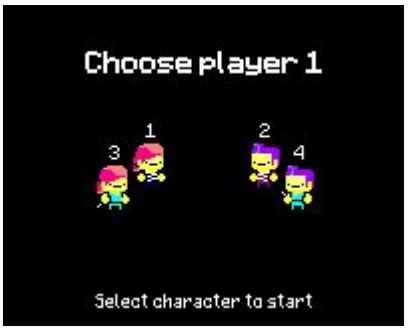
All sprites of our game have been pre-calculated so they are faster to render.

Preview designs
We did some sketches before creating the final sprites and designs for our game.

Players



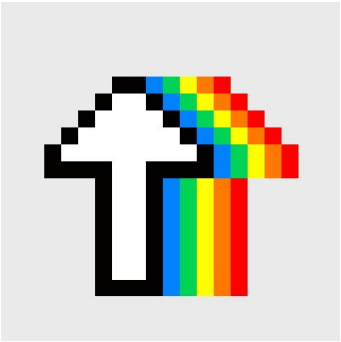
Menu



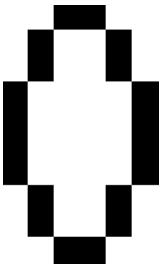
Final designs

Utopîa logo

We wanted to make our logo colorful to evoke peacefulness and amusement but also we wanted it to be retro and eighties.

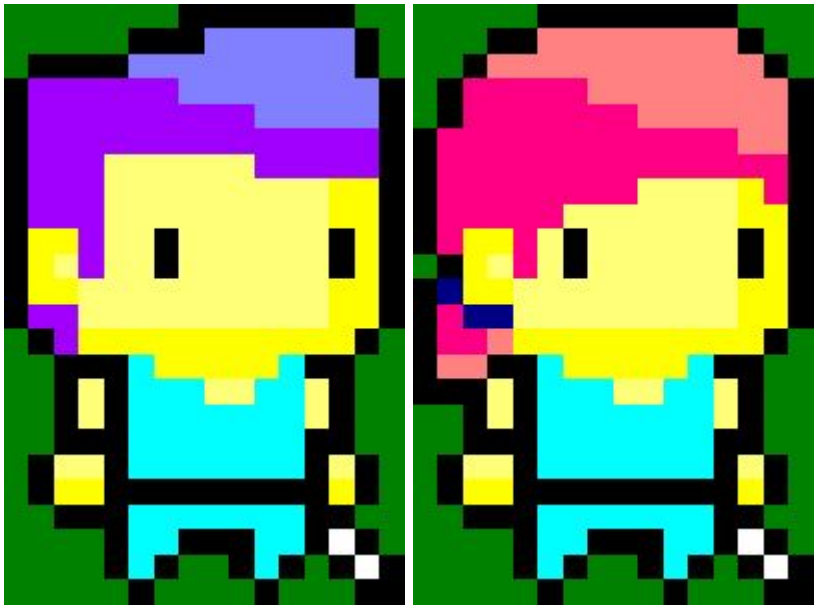


Ball

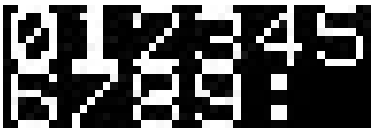


Players

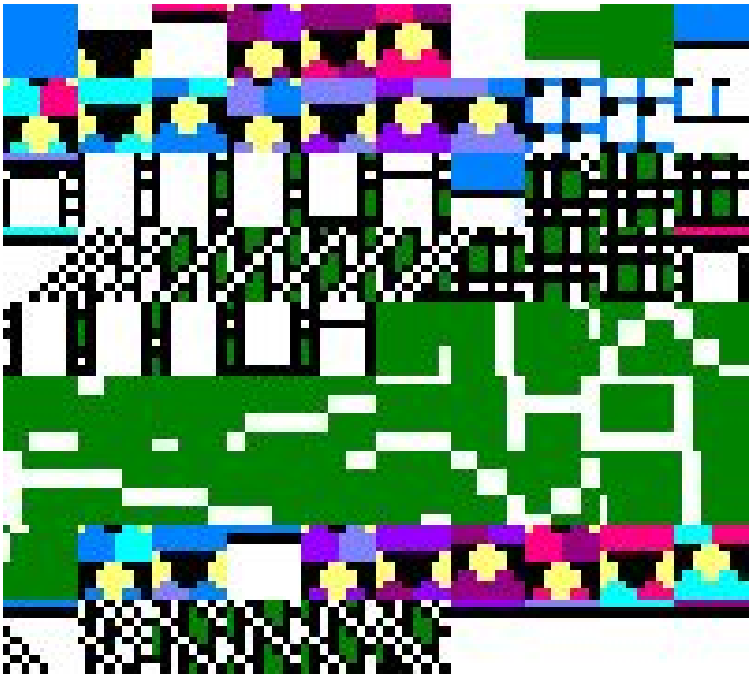
Prince of Persia skins



Numbers tileset



Tilemap and tileset

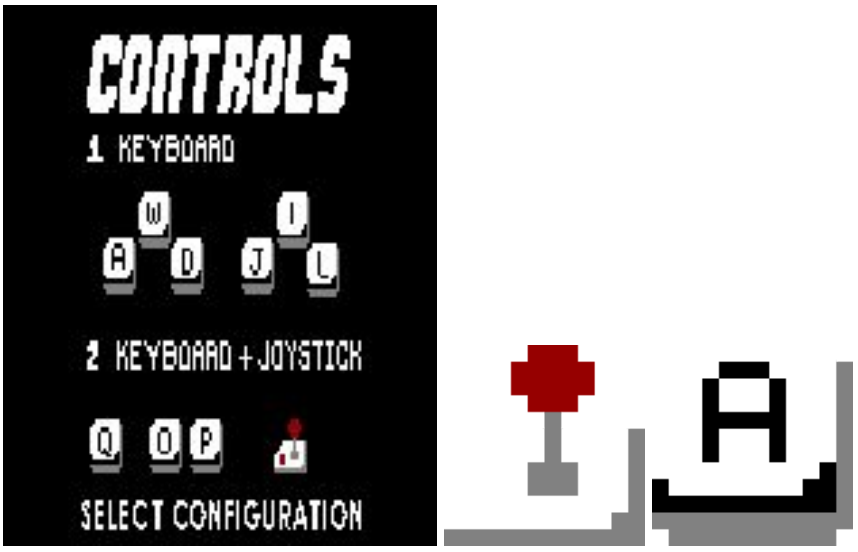


Loading screen

Main menu

Credits

Controls menu



Choose player menu

In game alerts

End menu

Music

The main music called bobline has been composed by Fenyx Kell, a french programmer and musician on CPC and CPC +, and a professional musician also specialised in electronic music.

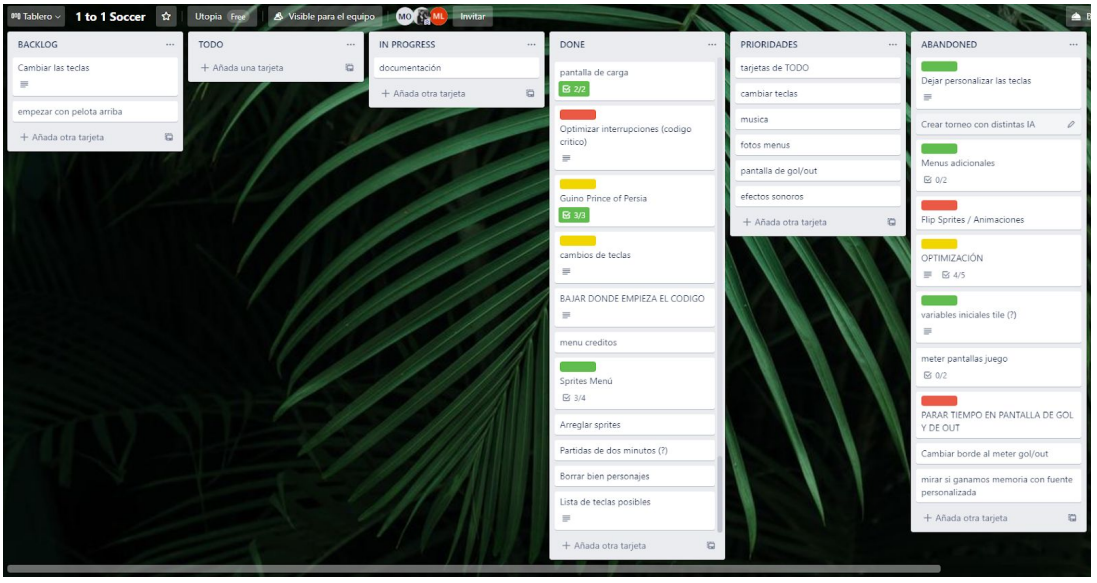
SFX

SFX are simple but made by us, in these sound effects we have a sound when there is a goal or the ball goes out, and the final beep of the game.

Also we have implemented panoramic audio so the sound of the SFX will sound left or right depending on the goal that references it.

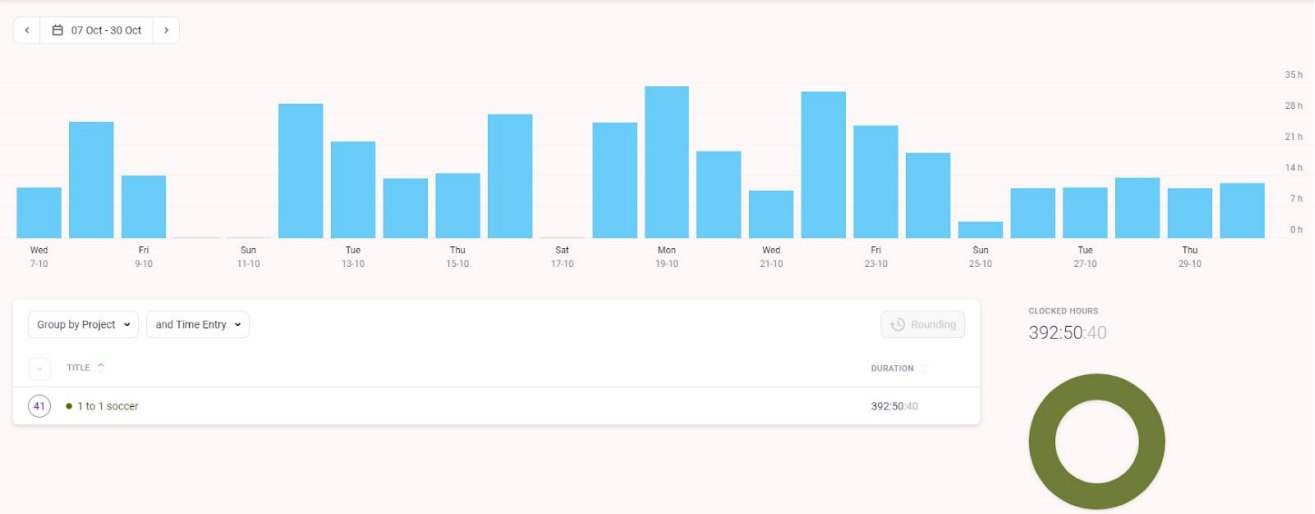
3. Working methodology

We used the well-known scrum methodology to organize the iterations using Trello. Every working day we used the first 10 minutes to decide the tasks we would be doing that day. Also we use Trello to list all tasks we would like to do the next iterations and all new ideas that come to our mind.



Picture 1. Trello

Also we had registered all working hours using Toggl track. This is the final result of the three components of the team without the hours we spend learning the technologies.



Picture 2. Toggl Track

To this amount we will have to include the hours testing and checking the game to see that everything is correct for the CPCRetro Dev.

Also we have commented all functions of the code so it can be easily understable, unfortunately it is written in spanish.

4. Problems and conclusion

Firstly we would like to thank Fenyx Kell again for letting us use his music in the game and also for the compliments and cheers he gave us. We have worked really hard in this game and also we now know we could have made a better game, we are really proud of our final game.

The first problem we encountered was the introduction of animations, we wanted to add animation to all movements of the game but couldn't do it without losing our 50 fps gameplay. Later on we decided to use precalculated sprites to make it faster and to add masks to our sprites, so now we could make animations but due to the complexity of sprites and the memory we still couldn't add animations to the game.

We have implemented hardware scroll but we had a hard time making it work with the 4x8 tiles functions but we make it work. Later we tried to add clipping to the render motor but once again it made the game run lower than 50 fps therefore we decided not to add it to the final version. Also we wanted the HUD to be part of the scenario but we couldn't due to the hardware scroll and for the cost of redrawing it at all time so we decided to paint it above the map.

To avoid interruptions to desynchronize we couldn’t use the CPCtelera functions to draw strings so we had to use sprites instead. Also we had to synchronize interruptions after redrawing the tilemap after a goal/out.

We have not fully controlled sfx as we would like to make it sound a bit higher but couldn't make it. Anyway it is more than audible and playable.

Also we had lots of problems with the AI trying to make it perfect, not too smart nor too silly. At the end we are really happy with the resulting AI and think that makes the game really enjoyable and replayable.

Finally and as a conclusion of this journey we would like to remark how challenging and sacrificed this project has been. We had so many more ideas for this game to develop but couldn't due to the limited amount of time we had. This being said we are happy with the result and proud of ourselves. We hope everyone enjoys the game and has a good time playing it.