2020

BABYLON FLOTILLA

MAKING OF

ANTONIO BRI PÉREZ,
SAUL VERDÚ APARICIO
JOSÉ VICENTE MARTINEZ MELLADO

Content

1. How the game was done?

1.1. Authors

We are 3 students at University of Alicante who were committed to develop a video game for Amstrad CPC for the competition CPCRetrodev2020. Our names and contact mails are:

- Antonio Bri Pérez: abp81@alu.ua.es
- Saul Verdú Aparicio: sva17@alu.ua.es
- José Vicente Martínez Mellado: jvmm4@alu.ua.es

1.2. Technologies and tools used

We used a variety of technologies to develop our videogame from game logic to sprites. Concretely, we used:

- **CPCTelera**: Game engine for Amstrad used during development.

- **Visual Studio Code / Sublime text**: Text editor used for the code.

- **Gimp**: Free & Open Source image editor for sprites and images

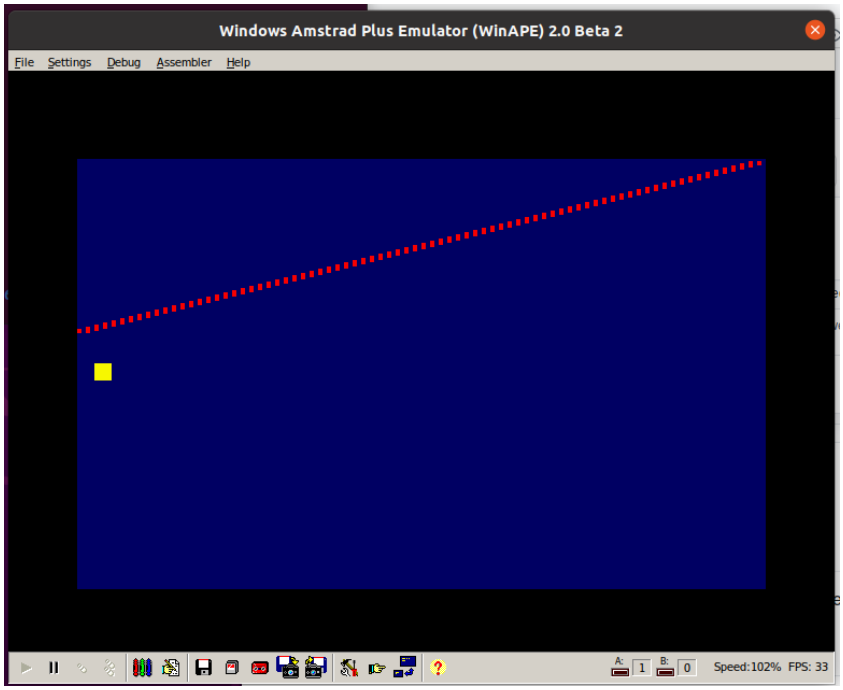
- **WinAPE**: Amstrad CPC emulator.

- **GitHub**: Development platform used to manage the project.

- **PyxelEdit**: Pixel art and tile creation tool for sprites

2. Developing problems & learned lessons

2.1. Early stages



During these early stages we only had a simple system of physics, render and an entity manager. That little yellow square you can see on the left of the screen is our main character and that red line crossing the display is an enemy. As you can see, we even had not a function to clear the movement of the enemy and render it again.

Also, in this version you were not able to move the main character with the keyboard or similar. However, our entity manager was solid and thanks to that we were able to progress quickly.

```
; Funcion para inicializar el  juego
entity_man_init::
    xor     a                       ; Colocamos un 0 en el registro A
    ld      (_num_entities), a      ; Ponemos a 0 la variable _num_entities

    ld      hl, #_entity_array      ; Colocamos en principio del Array en HL
    ld      (_last_elem_ptr), hl    ; Indicamos que  el _last_elem_ptr esta en el principio del Array
    ret


; Funcion que nos sirve para introducir lo que  hay en la posicion HL en nuestro array de entidades
; IMPUT
;   HL: Puentero a los bytes de inicializacion
entity_man_create::
    ld      de, (_last_elem_ptr)
    ld      bc, #entity_size
    ldir

    ld      a, (_num_entities)
    inc     a
    ld      (_num_entities), a

    ld      hl, (_last_elem_ptr)
    ld      bc, #entity_size
    add     hl, bc
    ld      (_last_elem_ptr), hl

    ret

; Funcion que nos sirve para introducir lo que  hay en la posicion HL en nuestro array de entidades
;   IMPUT
;       HL: Puntero a la entidad a destruir
entity_man_destroy::
    ret
```

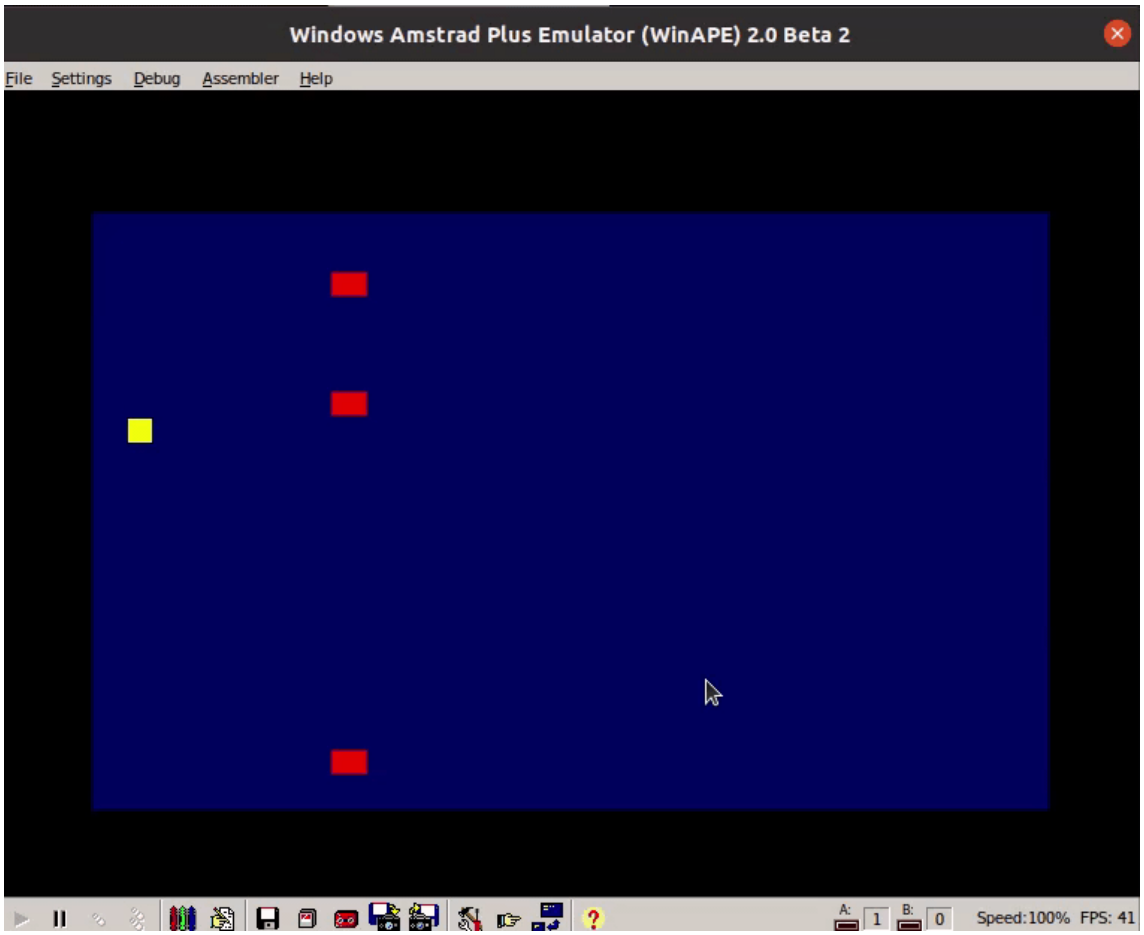
This was our entity manager, as you can see the code was able to initialize the manager and also create entities in memory. Those entities had also some attributes, and were able to be gathered in arrays

```
.macro DefineEntityAnnonimous _x, _y, _h, _w, _vx, _vy, _color, _prevX, _prevY
    .db _x
    .db _y
    .db _w
    .db _h
    .db _vx
    .db _vy
    .db _color
    .db _prevX
    .db _prevY
.endm

.macro DefineEntity _name, _x, _y, _w, _h, _vx, _vy, _color, _prevX, _prevY
    _name::
        DefineEntityAnnonimous _x, _y, _w, _h, _vx, _vy, _color, _prevX, _prevY
.endm

.macro DefineEntityArray _name, _N
    _name::
        .rept _N
            DefineEntityAnnonimous 0, 0, 0, 0, 0, 0, 0, 0, 0
        .endm
.endm
```

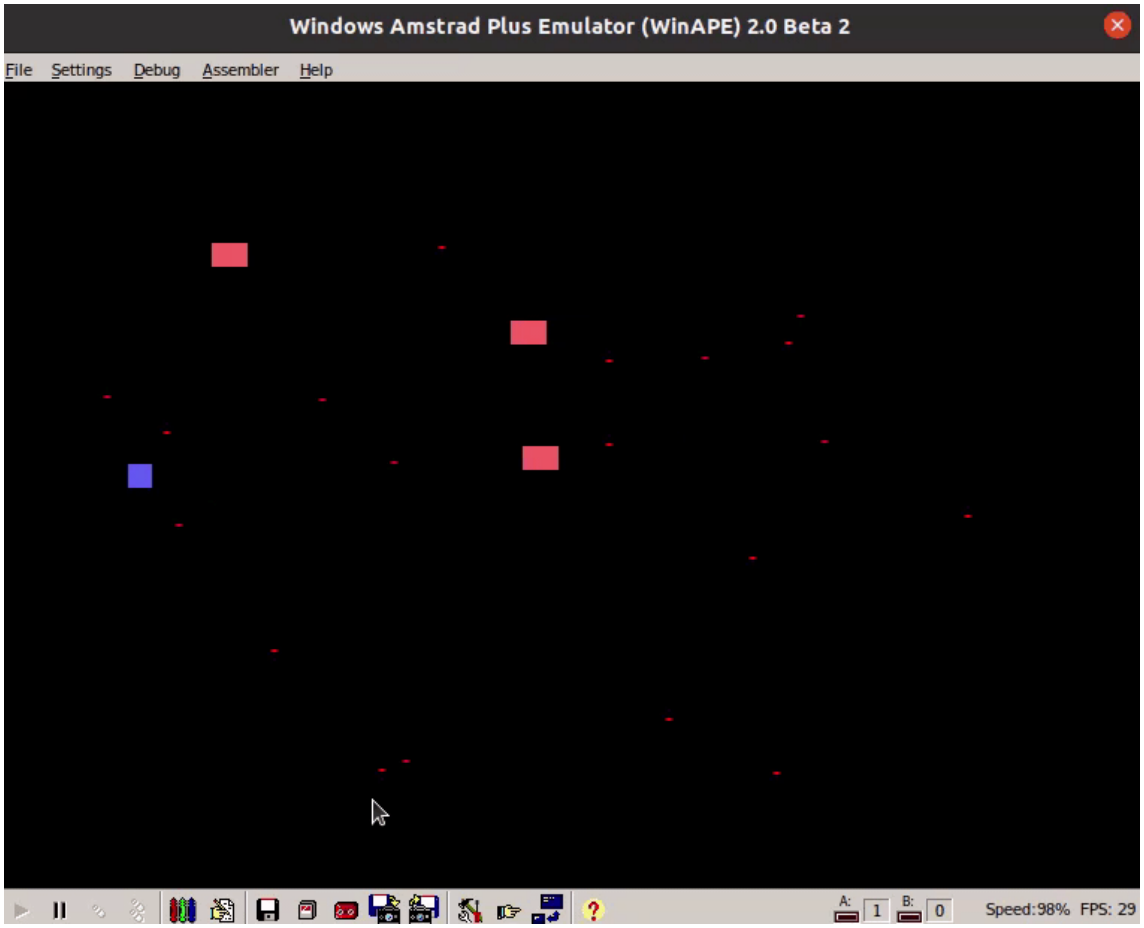
2.2. Basic movements and mechanical



As you can see, in this stage you were able to move the main character with the keyboard and also shoot and kill enemies. In this stage the collision physics were a challenge due to the way we had our entities gathered. We decided to gather all of our entities (main player, bullets and enemies) all in one single array. As we will see later, that was a huge mistake that lead us to change the collision loop practically entirely.

2.3. Movement effect and enemy's generation

After having the basic mechanical implemented, we were in condition to create a movement effect and to make enemies appear as long as the game advance.



In this version you were able to kill enemies, be killed by enemies and were able to implement all these features in a nice movement star field. However, enemies were created randomly and that is something we had to fix in order to make waves of enemies.

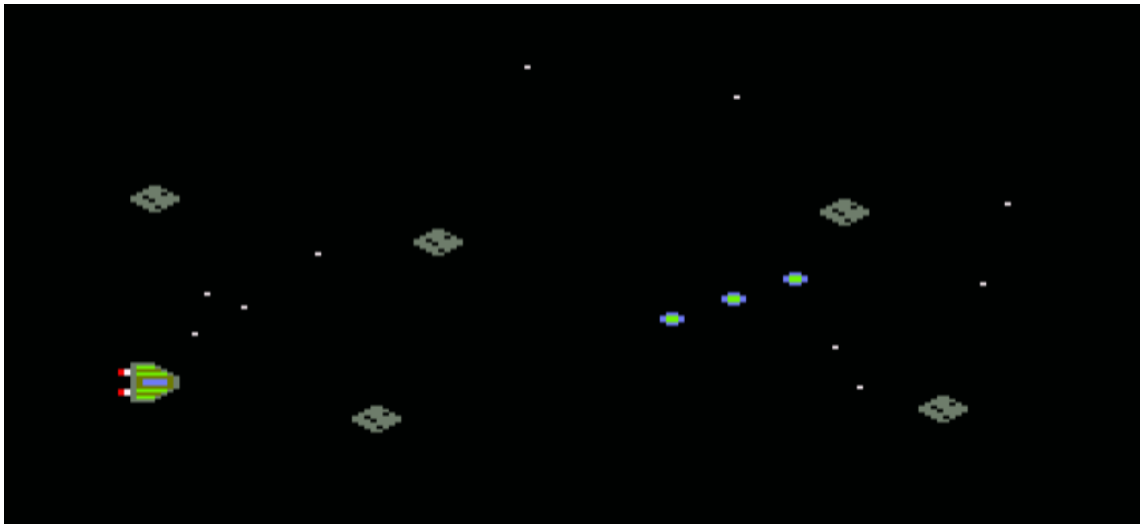
2.4. Sprites, loading screen, levels, and enemy waves

Finally, when we had our enemies and our mechanical basics we decided to implement 4 levels in which you had to kill a certain number of enemies to advance to the next level. Here it is a sort of captures of what we had so far:

A loading screen:

A nice start-up screen.

And our enemies with their nice sprites and our bullets:



The major problems with this iteration of developing was all the logic related to the levels, restarting the game once you die or once you overpass the game. We had to spend practically a whole week just fixing some annoying bugs that made the game quite hard to play.

2.5. Saving the Earth, only 3 chances

Once we did sprites and levels, we realized that the player had no other possibility more than just shot at enemies. The player was even capable of overpass our game without moving at all, just shooting. We thought it was a good idea to make our player to defend the planet earth from asteroids. That is why we draw an Earth sprite at the left of the screen and gave our character 3 lives to overpass the game. With these little changes we made the player to interact actively with the game; the game is not easy to play.

As we can see in the image, now the player must move and be aware than none of the asteroids collide against the Earth.

3. Final thoughts and conclusion

We really had a good time developing this game and we have learnt a lot in only 8 weeks. We would have loved to add more things like more interesting enemies, some IA to those enemies or even a score system for the player maybe with the last level infinite instead of just 4 levels. However, we did not have time to do that due to our duties we had to attend. If we were about to do another game for CPC we would absolutely made some things in a different way.