

OVERHEAP

Amstrad CPC 464 game



ÍNDICE

1. Ficha técnica
2. Making of
3. Sistema de Rondas
4. Sistema de Disparos
5. Scroll Hardware y Doble Buffer
6. IA
7. Análisis del proyecto



1. Ficha Técnica

Lenguaje utilizado: **Z80, C**

Tamaño del juego: 32.776 Bytes (Doble Buffer) + 21.447 Bytes (código y sprites) = **54.223 Bytes**

Modo Gráfico

Uso de **Modo 0** (1 byte , 2 píxeles ancho) con paleta de 16 colores

Detalles de implementación

Sistema de sprites

Librerías

IA

Modelos a seguir



2. Making of

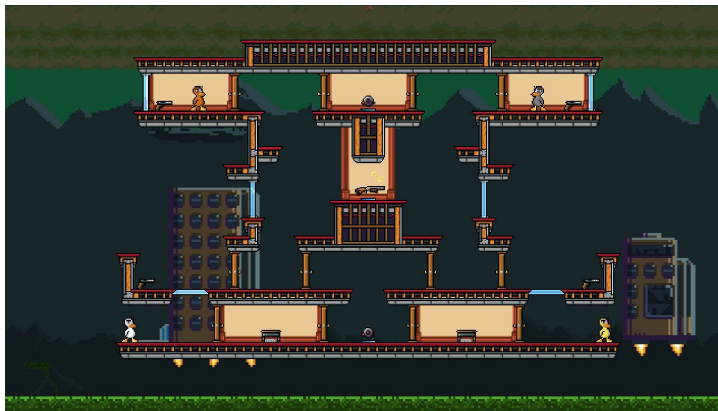


1. Pasos desde la idea original hasta la producción
2. Anécdotas del desarrollo

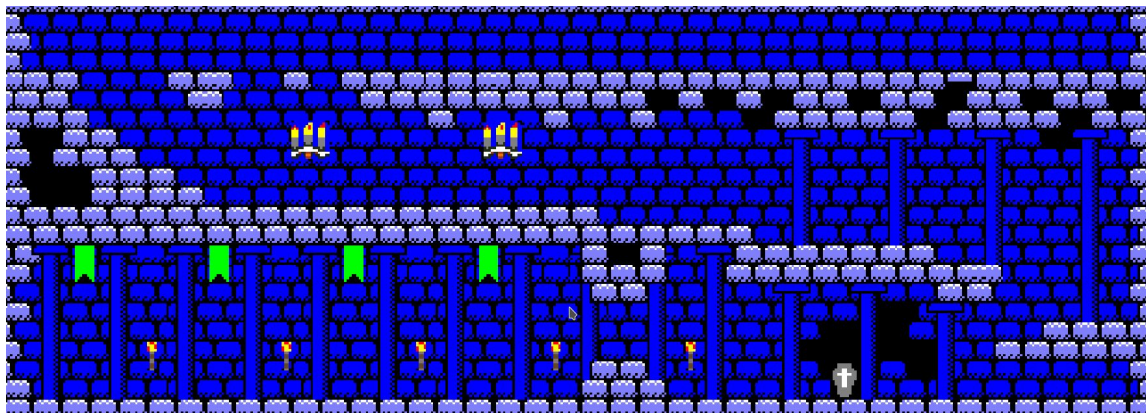
Galactic Tomb



The Duck Game



OverHeap



Tecnologías usadas:

- **Gimp** - para el tileset de los gráficos
- **Tiled** - para hacer los mapas y sprites
- **Arkos Tracker** - para la música del menú
- **Visual Studio Code** - para la programación en ASM - (VS Live Share)
- **CDT2WAV** - para cargar el juego en un Amstrad CPC
- **CPCTelera** - Game Engine para Amstrad CPC
- **Wine** - para emular Winape en Linux
- **Manjaro** - sistema operativo durante el desarrollo
- **Google Docs** - para hacer la documentación
- **Trello** - Organizar las tareas
- **GitHub** - Repositorio de código

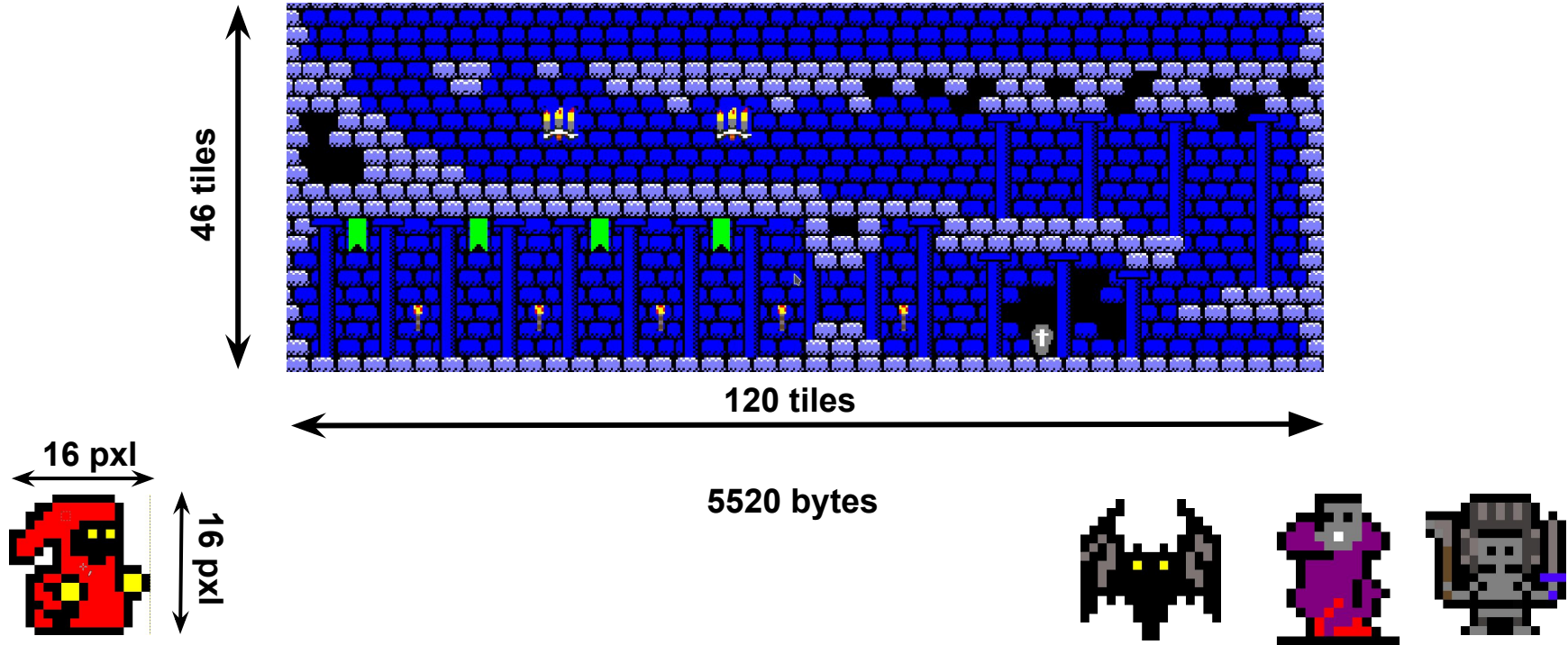


3. Sistema de Rondas

3.1 Diseño de la idea

3.2 Solución

3.3 Funcionamiento






























3. Sistema de Rondas


3.1 Diseño de la idea

3.1 Solución

3.3 Funcionamiento

1. Ronda						
2. Ronda					x5	
3. Ronda						x7
4. Ronda						 x9
5. Ronda						  x11

STOP PLS



3. Sistema de Rondas

3.1 Diseño de la idea

3.2 Solución

3.3 Funcionamiento

menu.s

Variables:

ronda: .db #0

max_ronda: .db #5

Funcion:

check_rondas:



enemy.s

Variables:

enm_map_alive: .db

#k_total_enm

enemies: .db #k_total_enm

life: .db 0x03

Constantes:

k_total_enm = #2

Funciones:

increase_enemies:

reset_posi:

increase_life:

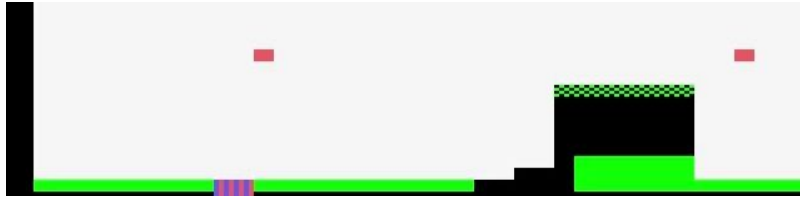
aplica_life:

enemy_improve:

4. Sistema de disparos

4.1 Primer disparo

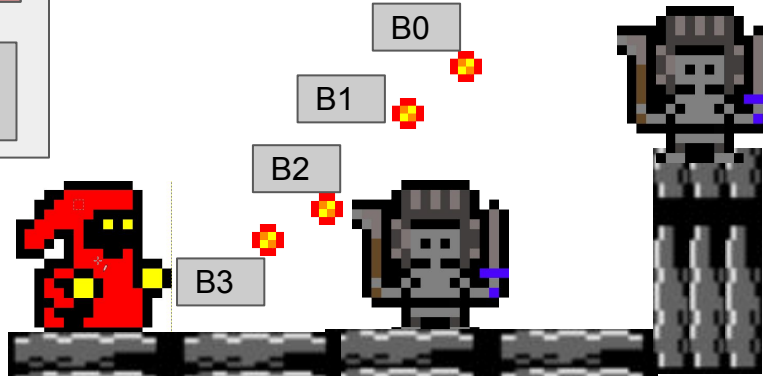
4.2 Generalizar



Obstacle.s

Bullet_0 0 / 1

Lógica para las balas



Obstacle.s

Bullet_0 0

Bullet_1 0

Bullet_2 0

Bullet_3 0

Lógica para las balas



4. Sistema de disparos

4.1 Primer disparo

4.2 Generalizar

Hero.h.s - Define Hero

Hero data (14B)

k_max_balas (+0B)

array pointer (+1B)

Define N Obstacles

$14B + 2B + (N * 8B)$

Enemy.h.s - Define Enemy Shoot

Define Enemy (14B)

k_max_balas (+0B)

array pointer (+1B)

Define N Obstacles

Enemy.h.s - Define Emey Shoot

Enemy + Enemy Shoot

Bullet_0

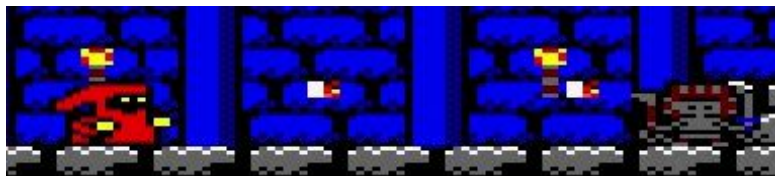
Bullet_1

Bullet_2

Bullet_3

Problemas a resolver generalizando el código:

- Mismo código para todas las entidades
- Shoot Rate distinto en cada entidad
- Dirección del disparo
- Limitar el alcance de la bala



5. Scroll Hardware y Doble Buffer

5.1 Doble Buffer

5.2 Scroll Hardware

tileManager.s

TScreenTilemapFront:

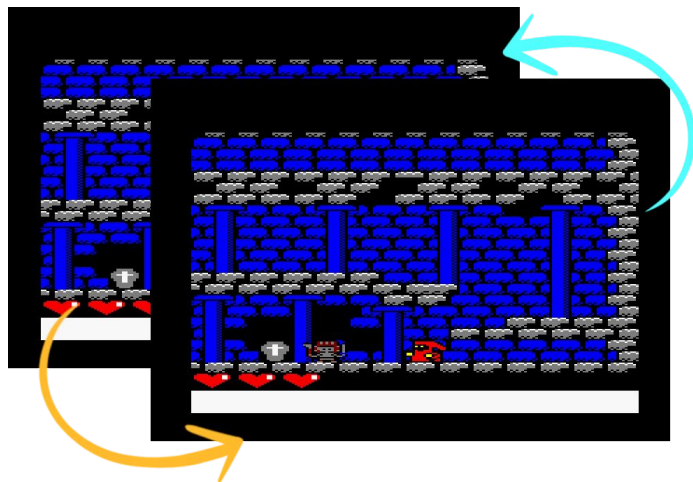
```
.dw #0x8000 | pVideo  
.dw #_g_tilemap | pTilemap  
.db 0x00 | scroll
```

TScreenTilemapBack:

```
.dw #0xC000 | pVideo  
.dw #_g_tilemap | pTilemap  
.db 0x00 | scroll
```

render.s

```
m_front_tileMap:: .dw #TScreenTilemapFront  
m_back_tileMap:: .dw #TScreenTilemapBack
```



ren_switchBuffers



setVideoMemoryPage

setVideoMemoryOffset

hero.s

hero_moveKeyboard

column
scroll

tileManager.s

```
pVideo    += 2*scroll  
pTilemap  += scroll  
scroll    += scroll
```

```
call cpct_etm_drawTileBox2x4_asm
```

```
;; Limpiar parte de abajo de pantalla porque los  
;; píxeles aparecen debido a la distribución de la  
;; memoria.
```

6. IA

Máquina de estados Finitos

enm_update:

```
;;#####  
;; Funciones previas  
;;#####
```

```
ld h, e_up_h(ix)  
ld l, e_up_l(ix)  
jp (hl) ;; llamada al estado
```

FSM:

```
;; Comprobar evento de entrada Y  
;; Sobreescibir estado
```

```
;; Código mutable para llamar al estado  
method= . + 1  
call #0x0000
```

Entidad

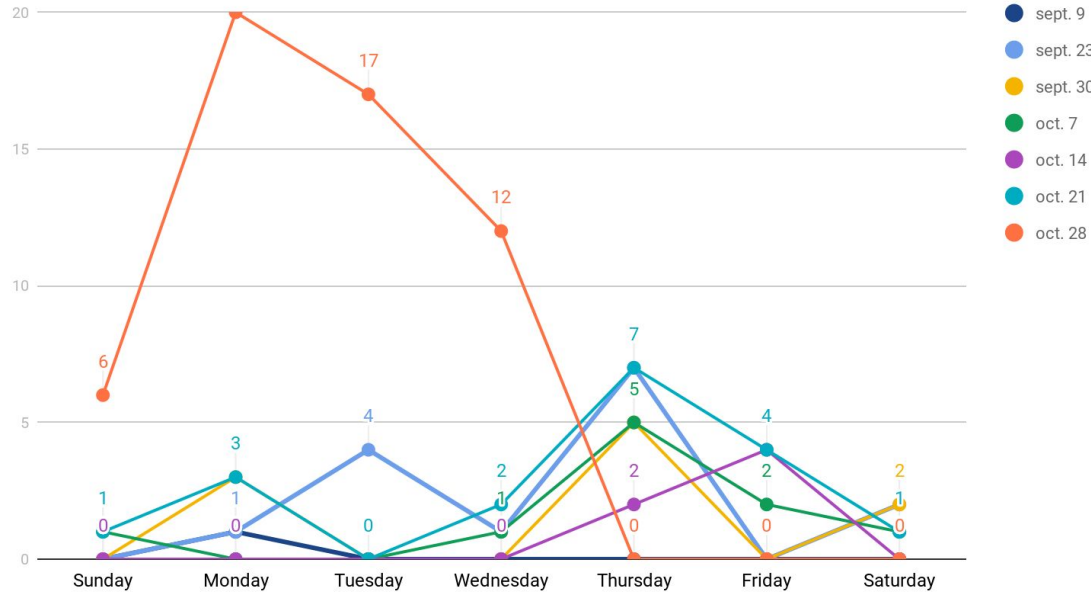


7. Análisis del proyecto

7.1 Datos recabados

7.2 Planificación

Commits



- ❖ Horas invertidas
- ❖ Distribución en el tiempo
- ❖ Uso de los recursos
- ❖ Costes sobre/infa-estimados
- ❖ Aciertos/fallos de planificación
- ❖ Toma de decisión

TODO

- Sonidos Ambiente
- Diseñar nivel
- Maquina de estados

+ Añada otra tarjeta

WIP

- Sistema de niveles

3/3
- Bugs

1 8/12
- Guiño Chicago's 30

2/3

+ Añada otra tarjeta

Done

- HUD- Vida

1
- Menu Inicial

1
- Sistema de Disparos

9/9
- Arreglar bug que el juego se cuelga

2
- Dibujar Sprites && Canal Alpha
- Arreglar salto


1
- Sistema de Controles-Teclas
- Sistema de Vidas
- Urgente Entity Enemies
- Urgente Entity Player
- Urgente Sistema de Salto Y movimiento
- Urgente Sistema Mapa de Tiles BASICO

Descartado

- Carga de mapas

1 1/6
- Logo juego Animado
- Menu controles
- Compresion de mapas
- Sprites Animados
- Dificil IA

+ Añada otra tarjeta





¿Preguntas?

@BastaCPC

Juan López Quiles

jlq2@alu.ua.es

<https://github.com/psjuan97>

José Luis Gómez Antón

jlga10@alu.ua.es

<https://github.com/jlga10>

Alejandro Aliaga Hyder

ajah1@alu.ua.es

<https://github.com/ajah1>

