

Faculdade de Engenharia da Universidade do Porto



Not Cuphead
Jogo em C para Minix

RELATÓRIO DE PROJETO

Rúben Esteves up202006479@fe.up.pt

Antero Morgado up202204971@fe.up.pt

Tiago Pinto up202206280@fe.up.pt

Leonardo Magalhães up202208726@fe.up.pt

Projeto realizado no âmbito da Licenciatura em Engenharia Informática
e Computação para a Unidade Curricular de Laboratório de
Computadores - L.EIC018

Professor das aulas Teórico-Práticas: Nuno Filipe Gomes Cardoso

02-06-2024

Índice

Índice	1
Introdução	3
Lista de Figuras.....	4
1. Instruções de utilização do programa	5
1.1 Menus	5
1.2. Jogo	9
1.2.1 Tipos de Inimigos	10
Bruxo	10
Belzebu	10
2. Estado do projeto	11
2.1. Timer	11
2.2 Keyboard	11
2.3 Mouse	11
2.4 Video Card	12
2.5 Real Time Clock	12
3. Organização do código	13
4. Detalhes da implementação	19
4.1 - Interrupções do RTC	19
4.2 - Aspetos Relacionados ao ScoreBoard.....	19
4.3 - Escolha de Resoluções	19
4.4 - Estruturas utilizadas para criar objetos	20
4.5 - Máquina de Estados relativos ao estado do jogo.....	21
4.6 - Colisão de balas com outros objetos	22
4.7 - Escrita no Ecrã.....	23
4.8 - Resoluções	24
5-Conclusão.....	25
5.1 Url para o link de demonstração.....	25

Introdução

Neste relatório, é descrito o projeto realizado na Unidade Curricular Laboratório de Computadores. Durante este projeto, desenvolveu-se um jogo chamado “Not Cuphead”, adaptado do famoso jogo “Cuphead”.

No “Not Cuphead”, o jogador deve tentar sobreviver o maior tempo possível e enfrentar vários inimigos. Para isso, ele pode fugir e disparar balas. Quanto mais tempo o jogador estiver vivo, maior será a sua pontuação. Também foi incluída uma *leaderboard*, onde podemos ver os maiores scores e os autores dos mesmos.

Lista de Figuras

Figura 1: Menu de Resoluções

Figura 2: Instruções

Figura 3: Menu Inicial

Figura 4: *LeaderBoard*

Figura 5: Menu de Fim de Jogo

Figura 6: Jogo

Figura 7: Bruxo

Figura 8: Belzebu

Figura 9: *Class Graph* do método “*proj_main_loop*”

Figura 10: Lógica do *Score*

Figura 11: *Struct Enemy*

Figura 12: *Struct bullet*

Figura 13: Máquina de Estados

Figura 14: *Check Collision*

Figura 15: *Create Bullet*

Figura 16: Pixmap da fonte

Figura 17: Diretórios das Resoluções com as Imagens

1. Instruções de utilização do programa

1.1 Menus

Ao iniciar o programa, o jogador deve escolher uma de quatro resoluções. Ao escolher uma opção com o rato, é redirecionado para o menu inicial, na resolução escolhida.

As quatro resoluções disponíveis representam as quatro resoluções com modelo de cor indexado.

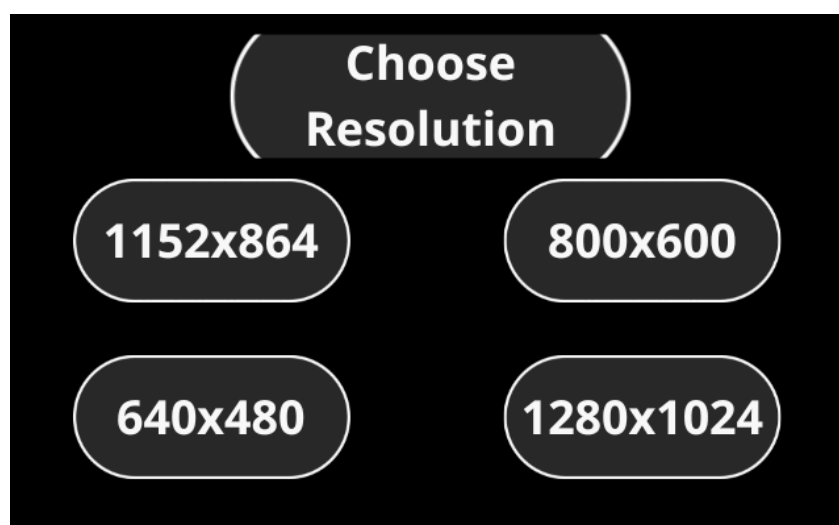


Figura 1: Menu de Resoluções

Antes do menu, aparecem no ecrã as instruções do jogo. Como mostra na figura, para se mover, o jogador usa as teclas W, A e D, botão esquerdo do rato para disparar e ESC para fechar o programa.



Figura 2: Instruções

No menu inicial, o jogador escolhe um de 3 botões:

- *Play*: Redireciona para o jogo.
- *Leaderboard*: Redireciona para o *LeaderBoard*.
- *Exit*: Termina a execução do jogo.

Para escolher a opção, o usuário deve mover o rato para o retângulo correspondente e clicar com o botão esquerdo.



Figura 3: Menu Inicial

No *leaderboard* são revelados os nomes dos jogadores que conseguiram obter as sete maiores pontuações, sendo exibidos o nome do jogador, nome esse que pode ser inserido no menu de fim de jogo, a sua pontuação e a data em que foi obtida. A primeira linha corresponde ao melhor score e a sétima ao sétimo melhor score obtido.

Para retornar ao menu, o usuário deve clicar no botão Menu, retornando ao menu inicial.



Figura 4: *LeaderBoard*

Quando o jogador fica sem vida é redirecionado para o menu de fim de jogo, onde é apresentado o score final do jogo e onde é possível escrever o username do jogador, que irá ficar registado no leaderboard no caso de se ter obtido um score superior a algum dos sete scores anteriores do leaderboard, ficando na respetiva linha do leaderboard.

Clicando no botão “Menu” ou clicando na tecla “Enter” é possível retornar ao menu inicial.

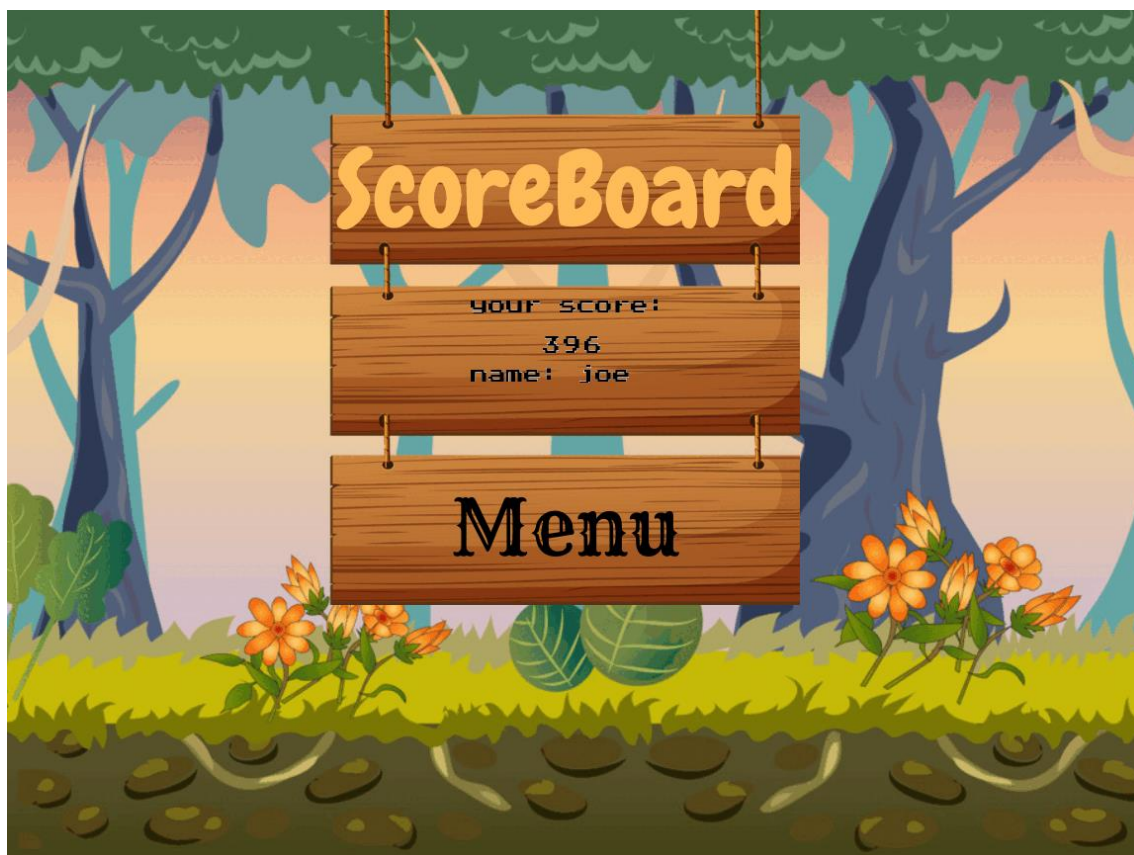


Figura 5: Menu de Fim de Jogo

1.2. Jogo

Durante o jogo, o jogador deve tentar sobreviver o maior tempo possível. Para isso, deve desviar-se de projéteis e inimigos. No canto superior direito é possível ver o score instantâneo, já no canto superior esquerdo a vida restante do jogador.

Para se movimentar, o utilizador usa as teclas “WASD”. O jogador deve disparar contra os inimigos, para os eliminar. Para isso, o jogador aponta com o rato e dispara com o botão direito. O jogo contém 2 tipos de inimigos.



Figura 6: Jogo

1.2.1 Tipos de Inimigos

Bruxo

Este inimigo causa dano ao colidir com o jogador. Para o eliminar, basta disparar acertando-lhe uma vez.



Figura 7: Bruxo

Belzebu

Este inimigo assustador voa e dispara projéteis laranjas que causam dano ao jogador. Para o eliminar, é necessário disparar e acertar duas vezes.



Figura 8: Belzebu

2. Estado do projeto

Dispositivo	Propósito	Interrupções
Timer	Controlar a frame-rate e realizar operações temporais	Sim
KBD	Movimentação do jogador e escrever nome	Sim
Mouse	Disparar e selecionar opções no menu	Sim
Video Card	Visualização do jogo e menus	Não
RTC	Apresentação do dia/hora na leaderboard	Não
Serial Port	Não implementado	Não

2.1. Timer

O timer é usado, com a frequência de 30 Hz para controlar a *frame-rate*. Também controla a técnica de *double buffering*, através da chamada da função *switch_buffers*. Também é usado para diversas operações temporais no jogo, como *spawn* de inimigos, *cooldowns*, controlar o multiplicador do score e outros. A implementação encontra-se no ficheiro “*timer.c*”, na pasta dos dispositivos.

2.2 Keyboard

Durante o jogo, o jogador movimenta-se com as teclas W, A e D. O “*makecode*”/“*breakcode*” da tecla permite que a mesma seja interpretada, através da função *processScanCode*. Para sair do jogo, o jogador deve usar a tecla “ESC”.

O teclado também é usado para o input do nome do jogador, após o jogo terminar, para ser guardado na *leaderboard*, se aplicável. A implementação encontra-se no ficheiro “*keyboard.c*”, na pasta dos dispositivos.

2.3 Mouse

O mouse é utilizado para selecionar a opção nos diversos menus e para disparar balas durante o jogo. Ao mover o mouse, a *sprite* do mesmo é movida e ao clicar com o botão esquerdo, é possível escolher a opção ou disparar a bala na direção do cursor. Em relação à posição do mouse, é lido o seu deslocamento através das coordenadas x e y. A implementação do rato encontra-se no ficheiro “*mouse.c*”, na pasta dos dispositivos.

2.4 Video Card

Inicialmente, para a escolha de resolução, o menu abre com resolução 640x480. De seguida, o utilizador pode escolher entre as resoluções 640x480, 800x600, 1280x1024, 1152x864, correspondendo aos modos 0x110, 0x115, 0x11A e 0x14C. Usamos a técnica de *double buffering*, com *page flipping*, através da função `switch_buffers`, de forma a melhorar a fluidez do jogo. Para as imagens, em formato XPM, usamos *sprites*, que são desenhados através da placa gráfica. A implementação encontra-se no ficheiro “*graphics.c*”, na pasta dos dispositivos.

2.5 Real Time Clock

O RTC é usado para guardar e mostrar a data e hora das entradas. São guardados o ano, mês, dia, hora, minutos e segundos. A leitura destes valores é apenas realizada no momento de escrita e cancelada caso o RTC esteja a atualizar os seus valores internamente. Dado isto, a leitura é colocada num loop em que a condição de saída é uma leitura correta.

3. Organização do código

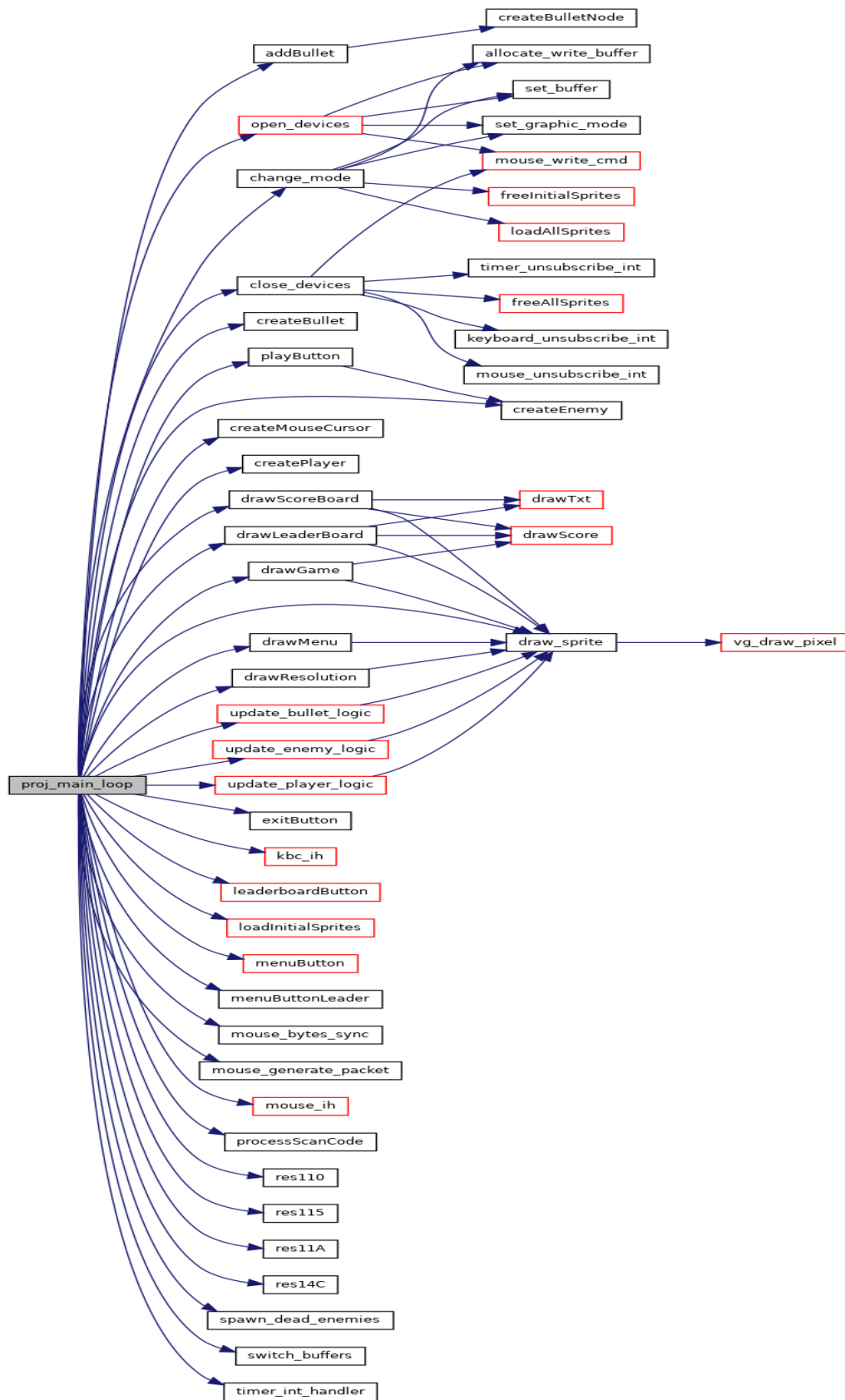


Figura 9: Class Graph do método “proj_main_loop”

Timer – 10%

Not Cuphead

Este módulo contém tudo aquilo o que foi feito no Lab2 de maneira a cumprir com as necessidades de acordo com o projeto. Foi bastante usado, uma vez que tudo dependia de espaços de tempo, como as animações e a lógica do jogo.

Desenvolvedores:

- Rúben Esteves (25%)
- Leonardo Teixeira (25%)
- Tiago Pinto (25%)
- Antero Morgado (25%)

Graphics – 10%

Este módulo contém tudo aquilo o que foi feito no Lab5 de maneira a poder livremente desenhar pixeis na janela no ecrã.

Desenvolvedores:

- Rúben Esteves (25%)
- Leonardo Teixeira (25%)
- Tiago Pinto (25%)
- Antero Morgado (25%)

Teclado – 8%

Este módulo contém tudo aquilo o que foi feito no Lab3 de maneira a poder usar o teclado.

Desenvolvedores:

- Rúben Esteves (25%)
- Leonardo Teixeira (25%)
- Tiago Pinto (25%)
- Antero Morgado (25%)

Rato – 8%

Este módulo contém tudo aquilo o que foi feito no Lab4 de maneira a poder usar o rato.

Desenvolvedores:

- Rúben Esteves (25%)
- Leonardo Teixeira (25%)
- Tiago Pinto (25%)
- Antero Morgado (25%)

RTC – 4%

Este módulo é responsável por apresentar as datas da realização de cada pontuação apresentada na Leaderboard.

Desenvolvedores:

- Rúben Esteves (100%)

Game Logic / Save Score – 15%

Este módulo é responsável por guardar e ler scores e por controlar o comportamento dos inimigos e do *player*.

Desenvolvedores:

- Rúben Esteves (25%)
- Leonardo Teixeira (25%)
- Tiago Pinto (25%)
- Antero Morgado (25%)

Sprite – 12%

Este modulo deu vida ao jogo, com ele foi possível mostrar os objetos do jogo e até animá-los, dando assim uma outra perspetiva ao jogador. Com este módulo também foi possível implementar alguma lógica, como a colisão de objetos.

Desenvolvedores:

- Leonardo Teixeira (33,3%)
- Tiago Pinto (33,3%)
- Antero Morgado (33,3%)

Utils – 1%

Ficheiro auxiliar dos dispositivos.

Desenvolvedores:

- Rúben Esteves (25%)
- Leonardo Teixeira (25%)
- Tiago Pinto (25%)
- Antero Morgado (25%)

Classes – 10%

Declaração dos *structs* e respetivos métodos.

Desenvolvedores:

- Rúben Esteves (25%)
- Leonardo Teixeira (25%)
- Tiago Pinto (25%)
- Antero Morgado (25%)

Documentação– 3%

A documentação inclui os comentários em Doxygen e os XPM.

Desenvolvedores:

- Rúben Esteves (25%)
- Leonardo Teixeira (25%)
- Tiago Pinto (25%)
- Antero Morgado (25%)

Main – 10%

Ficheiro que contem o *loop* responsável pela atualização continua do jogo e dos dispositivos utilizados.

Desenvolvedores:

- Rúben Esteves (25%)
- Leonardo Teixeira (25%)
- Tiago Pinto (25%)
- Antero Morgado (25%)

Menu – 9%

Este módulo é responsável por tratar as escolhas que o utilizador pode fazer no menu, tratando dos cliques do jogador. Ele também é responsável pelo desenho de todos os fundos de ecrã apresentados ao longo do jogo.

Desenvolvedores:

- Rúben Esteves (25%)
- Leonardo Teixeira (25%)
- Tiago Pinto (25%)
- Antero Morgado (25%)

4. Detalhes da implementação

4.1 - Interrupções do RTC

Embora o RTC seja capaz de lançar interrupções ao atualizar os seus valores, não consideramos apropriado ao nosso projeto a utilização das mesmas dado que os valores são apenas necessários espontaneamente (momento de escrita do *highscore*) e não constantemente. Sentimos que a utilização das interrupções, neste caso, levaria a uma diminuição da performance.

4.2 - Aspetos Relacionados ao *ScoreBoard*

O score é atualizado a cada dois terços de segundo, aumentando em 10 * multiplicador, multiplicador este que a cada dois segundos aumenta 0.1, este multiplicador começa em 1.1 no início do jogo e o score a 0.

```
if (currentState == GAME) {  
    if (counter_timer % 60 == 0) {  
        multiplier+=0.1;  
    }  
    if(counter_timer%20 == 0) {  
        score+=10*multiplier;  
    }  
}
```

Figura 10: Lógica do *Score*

As informações do score como nome, data e score ficam guardadas no ficheiro "scoreboard.txt" no qual estão já guardadas por ordem decrescente, caso o score de uma partida não ultrapasse nenhum dos sete scores guardados não há alteração ao ficheiro de texto.

4.3 - Escolha de Resoluções

A nossa escolha para fazer a mudança de resoluções obrigou a que fosse preciso outro ciclo, parecido ao ciclo principal, no entanto, este ciclo apenas serve para a escolha de resolução pretendida. Assim que a resolução for escolhida, irá reiniciar as interrupções da gráfica, voltando a iniciá-la no modo pretendido a seguir, seguindo depois para o jogo normalmente.

4.4 - Estruturas utilizadas para criar objetos

De forma a ser possível criar objetos jogo, foi preciso recorrer a estruturas de dados, assim foi criada uma estrutura diferente para cada um dos objetos implementados no jogo, bem como funções para os criar. Disso são exemplos:

Os Inimigos:

```
/**
 * @brief Struct representing an enemy entity in the game.
 *
 * Stores information about the enemy's attributes and behavior.
 */
typedef struct
{
    int life;
    int damage;
    int x;
    int y;
    int speed_x;
    int speed_y;
    Sprite *sprite;
    bool alive;
    enemy_bullet *shot;
} enemy;
```

Figura 11: *Struct Enemy*

As Balas:

```
/**
 * @brief Struct representing a bullet fired by a player.
 *
 * Stores information about the bullet's position, velocity, damage, and associated sprite.
 */
typedef struct {
    int x;
    int y;
    int mouseX;
    int mouseY;
    int damage;
    float dx;
    float dy;
    Sprite *sprite;
} bullet;
```

Figura 12: Struct bullet

4.5 - Máquina de Estados relativos ao estado do jogo

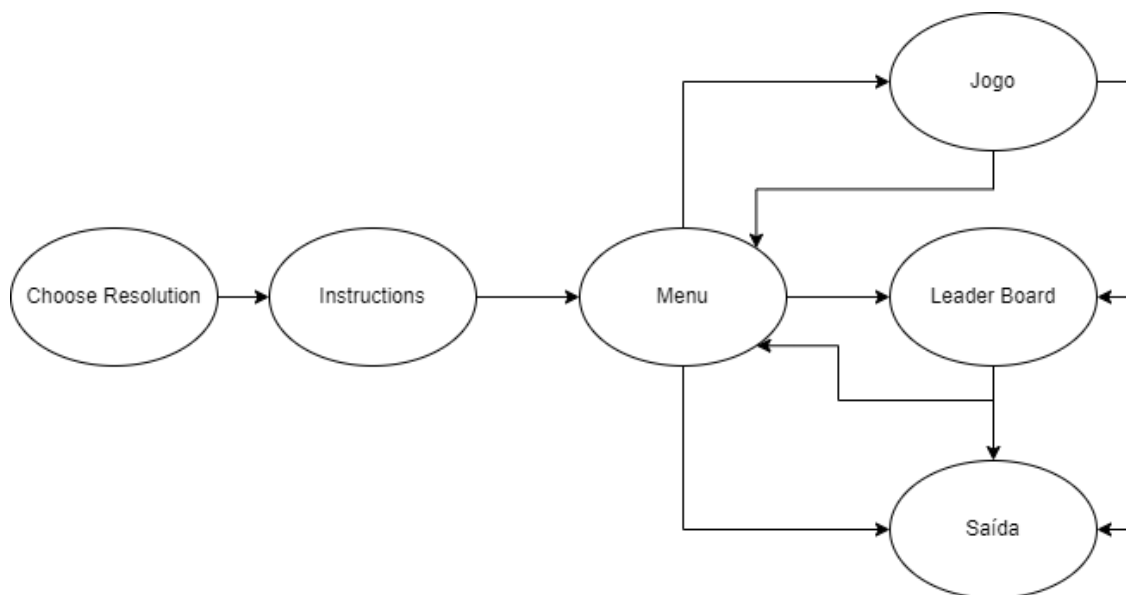


Figura 13: Máquina de Estados

4.6 - Colisão de balas com outros objetos

Uma das coisas que foi preciso aprender remotamente, foi a colisão das balas com os inimigos e com o boneco principal. Este processo foi feito comparando os limites das *Sprites* de ambos os objetos:

```
bool check_collision(Sprite *sp1, int x1, int y1, Sprite *sp2, int x2, int y2) {
    if (sp1 == NULL || sp2 == NULL) return false;

    int sp1_x_end = x1 + sp1->width;
    int sp1_y_end = y1 + sp1->height;
    int sp2_x_end = x2 + sp2->width;
    int sp2_y_end = y2 + sp2->height;

    if ((x1 < sp2_x_end && sp1_x_end > x2) && (y1 < sp2_y_end && sp1_y_end > y2)) {
        return true;
    }
    return false;
}
```

Figura 14: Check Collision

Outro ponto importante que foi adquirido fora das aulas, foi a criação de uma bala que fosse de encontro à localização do rato/*player*. Para isso, recorreremos à normalização da diferença da localização de envio da bala com a localização pretendida, permitindo assim criar uma trajetória para a bala:

```
bullet *createBullet(int x,int y, int mouseX, int mouseY,int damage,Sprite *sp){
    bullet *p = (bullet *)malloc(sizeof(*p));
    if(p == NULL)
        return NULL;
    p->x = x;
    p->y = y;
    p->mouseX = mouseX;
    p->mouseY = mouseY;
    p->damage = damage;
    p->sprite = sp;
    p->dx = mouseX - x;
    p->dy = mouseY - y;

    float norm = sqrt(p->dx * p->dx + p->dy * p->dy);
    if (norm != 0) {
        p->dx = p->dx / norm;
        p->dy = p->dy / norm;
    }

    return p;
}
```

Figura 15: Create Bullet

4.7 - Escrita no Ecrã

Para escrever no ecrã, foi usada uma fonte xpm convertida a partir de uma ttf e depois cada letra foi mapeada por posições (x,y) e posteriormente desenhada com a sua altura e largura.

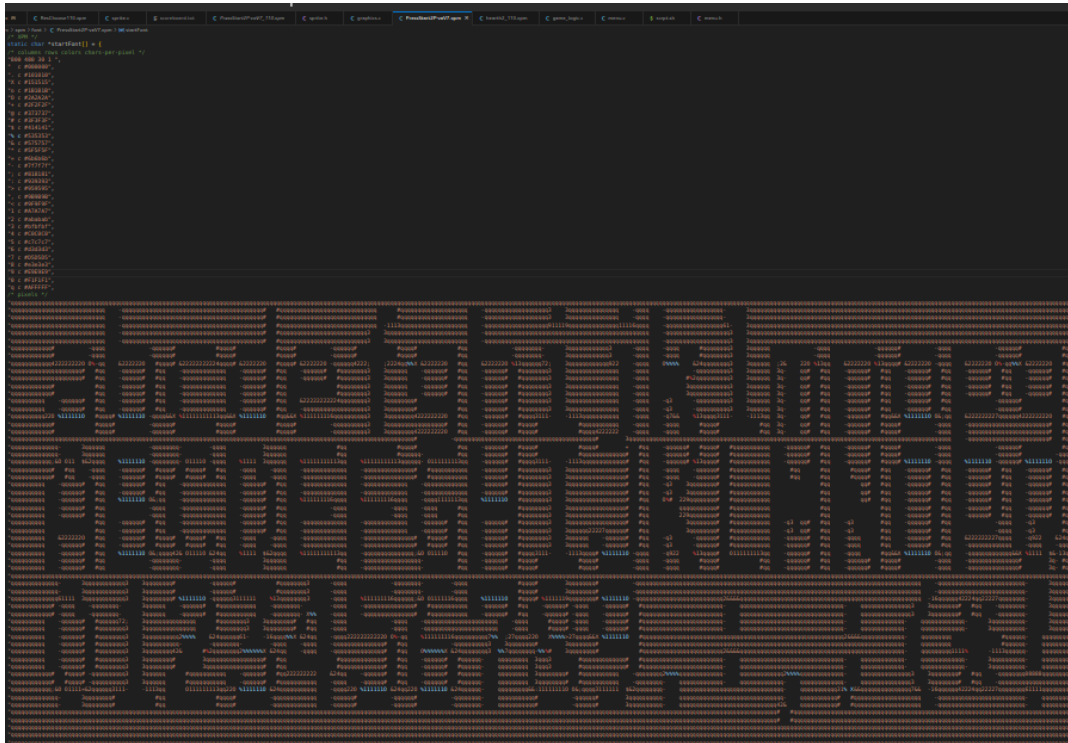


Figura 16: Pixmap da fonte

4.8 - Resoluções

Para que houvesse suporte para várias resoluções foi necessário converter imagens xpm para outros tamanhos. Para tal foi usado programa ImageMagick. O jogo foi, numa primeira instância, feito para o modo 0x14C e posteriormente convertido para os restantes modos.

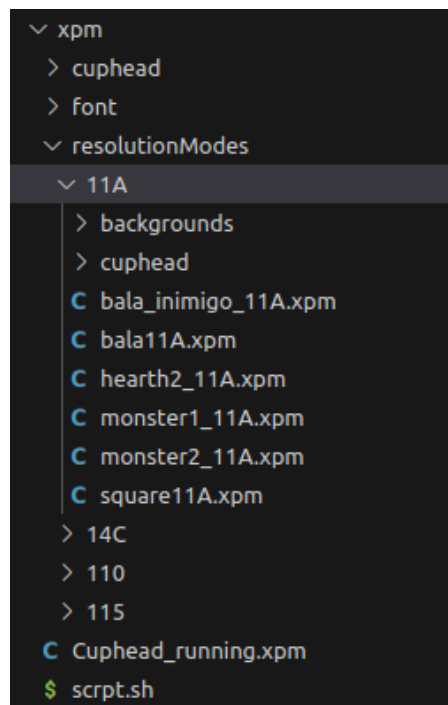


Figura 17: Diretórios das Resoluções com as Imagens

5-Conclusão

Embora não tenha sido possível realizar tudo a que nos propusemos, ainda assim, o resultado pode ser considerado muito positivo, já que o ficou mais conciso do que era previsto na proposta inicial.

O jogo foi uma tentativa de recriação do jogo mundialmente conhecido “CupHead”. Consideramos que foi um sucesso tanto a nível de resultado como a nível de aprendizagem, já que nos permitiu evoluir e compreender ainda mais os drivers e dispositivos que utilizamos ao longo do mesmo.

A maior dificuldade foi a implementação de RTC e porta série. Em relação à última, apesar de tentarmos, não tivemos tempo suficiente para realizar, pelo que decidimos descartar a implementação efetuada.

O trabalho foi distribuído igualmente pelos membros, pelo que consideramos que todos tiveram um papel ativo e fundamental na realização do mesmo, promovendo a partilha de conhecimento e fomentando o crescimento tanto a nível pessoal como a nível de conhecimento dos assuntos tratados ao longo do projeto.

5.1 Url para o link de demonstração

https://drive.google.com/file/d/1OKGuWT0-689qolxFq_w5trtvnWgCQ3b6/view