Instituto Politécnico de Viana do Castelo

Escola Superior de Tecnologia e Gestão

Engenharia da Computação Gráfica e Multimédia
Sistemas Multimédia

Exercício Prático – Phaser

"Le Rambo"

Leandro Martins Magalhães, nº 17714.

Dezembro de 2018.

Índice

1. Introdução

Este projeto foi desenvolvido no âmbito da unidade curricular de Sistemas Multimédia, do curso de Engenharia da Computação Gráfica e Multimédia, da Escola Superior de Tecnologia e Gestão, que pertence ao Instituto Politécnico de Viana do Castelo, utilizando a linguagem de programação gráfica Phaser.

O projeto é de tema livre possuindo alguns requisitos, como por exemplo, fazer referência ao curso. Tendo em conta isso escolhi desenvolver o "Metal Slug", com elementos gráficos e sonoros originais e personalizados por mim.

Espero com isto, apresentar um projeto original e único, e com fácil controlo sobre as ações dentro do jogo.

2. Manual

O jogador é controlado com o teclado, clicando nas setas e espaço.

Com as setas esquerda e direita, movimenta o jogador nas respetivas direções, já com o espaço, irá fazer com que este salte.

De seguida, a seta de cima faz com que o jogador dispare balas, já a seta de baixo, lança granadas.

No menu, a interação é completamente feita com o cursor do rato.

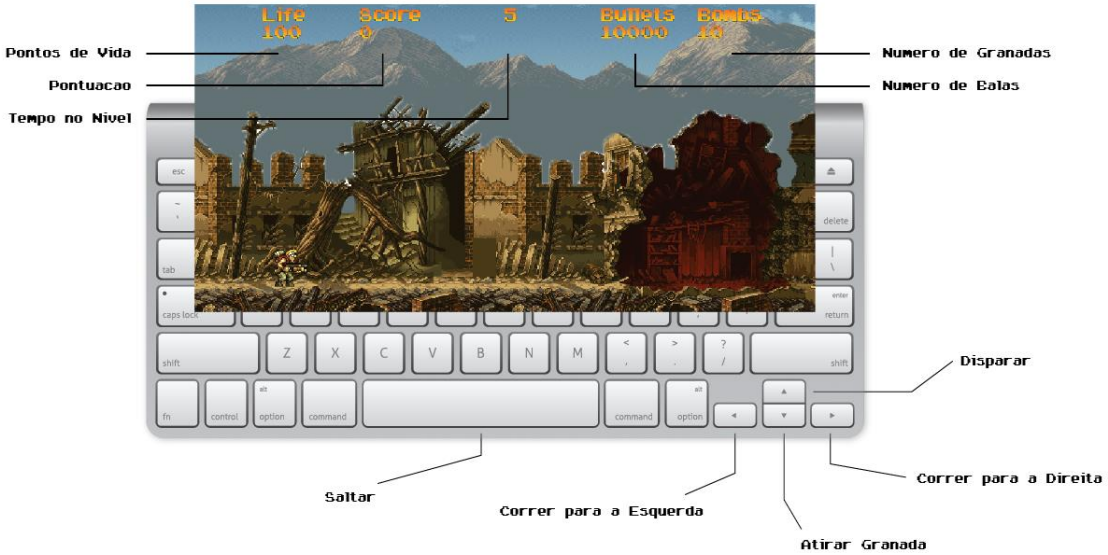

Figura 1: Explicação gráfica

3. Projeto

2.1. Descrição

O projeto consiste numa versão original do jogo “Metal Slug”, este tem um tema retro, em género pixel art. Os gráficos e música ajudam o jogador a emergir nesse tema.

Neste o jogador toma os comandos do *rambo*, tendo que passar por vários soldados inimigos para chegar ao seu objetivo, deste modo vencerá.

Caso seja atingido demasiadas vezes, morrerá e perderá o jogo.

Figura 2: Jogo original Metal Slug

2.2. Estrutura

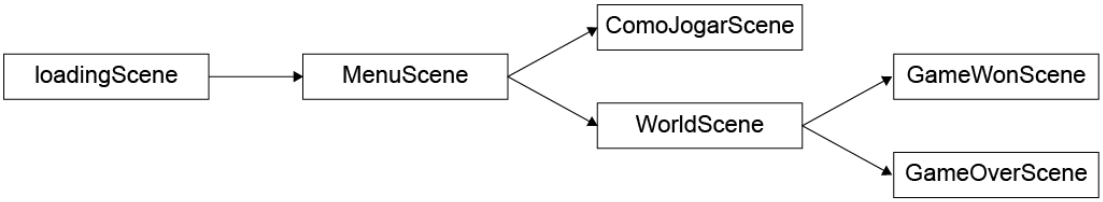

Figura 3: Estrutura do Projeto

2.4. Funcionalidades

Animações

As animações do jogador, explosão e do soldado foram feitas através de um ficheiro json. Duas das animações do jogador são as seguintes:

```
this.anims.create({
    key: "run",
    frames: this.anims.generateFrameNames("playersheet", {
        start: 1,
        end: 10,
        prefix: "ramboRun",
        suffix: ".png"
    }),
    frameRate: 15,
    repeat: -1
});

// --------- parado
this.anims.create({
    key: "stoped",
    frames: this.anims.generateFrameNames("playersheet", {
        start: 1,
        end: 4,
        prefix: "ramboStoped",
        suffix: ".png"
    }),
    frameRate: 5,
    repeat: -1
});
```

Figura 4: Exemplo de animações.

Font

A font é em bitmap.

```
// -- fonts
this.load.bitmapFont('fontPixel', 'assets/font/arcadepix.png', 'assets/font/arcadepix.fnt');
```

Figura 5: Load da bitmapFont.

```
// -- descricao
this.add.bitmapText(200, 430, 'fontPixel', 'Leandro Martins Magalhaes - 2018 - Copyrights Reserved', 12)
    .setDepth(1)
    .setScrollFactor(0);
```

Figura 6: Inserir bitmapFont.

Soldados

O modo como é feito o spawn dos soldados. Existe um máximo de três soldados na janela, de modo a controlar a performance do browser.

```
// -- metodo de spawn de soldados
spawnSoldados(time) {
    if (time > this.spawnTimeOffset) {
        var soldado = this.soldadosEnimigos.create(
            this.player.x + 530, // 800 5*
            this.player.y - 50,
            "soldadosheet"
        );

        soldado.body.gravity.y = 300;
        soldado.body.gravity.x = 0;
        //soldado.allowDrag = false;
        soldado.setScale(1.1);
        soldado.vida = 100;     // vida total do soldado
        soldado.shootOffset = 0;
        soldado.stage = 0; // 1 = correr | 2 = assustado | 3 = retirar arma | 4 = disparar
        soldado.ataque = false; // se esta em modo ataque ou não
        soldado.disparar = false; // se pode ou não disparar

        this.spawnTimeOffset = time + 2000;
    }
}
```

Figura 7: Spawn dos soldados inimigos.

Os soldados possuem fases (stages), em que dependendo de qual fase está faz algo diferente, por exemplo, na fase 1 o soldado corre numa direção até que fique perto do soldado, altura em que passa para a fase 2, em que mostra a animação de assustado, assim que a animação finalizar, inicia a animação de retirar a arma, e de seguida, quando esta tiver finalizado, inicia a animação de posição de disparo.

```
// -- Stage 2: soldado assusta-se
if (soldados[i].stage == 1 && soldados[i].ataque) {
  soldados[i].anims.remove("soldadocorrer");
  soldados[i].anims.play("soldadoscared", true);
  soldados[i].stage = 2;
}

// -- Stage 3: soldado retira a arma
if (soldados[i].stage == 2 && !soldados[i].anims.isPlaying) {
  soldados[i].anims.play("soldadosacaarma", true);
  soldados[i].stage = 3;
}

// -- Stage 4: poem-se em posição de disparo
if (soldados[i].stage == 3 && !soldados[i].anims.isPlaying) {
  soldados[i].anims.play("soldadodisparar", true);
  soldados[i].stage = 4;
}
```

Figura 8: Diferentes fases dos soldados.

Para alem disso, os soldados também possuem estados, de ataque e disparar. O estado de ataque só é ativado quando o soldado se encontra a pelo menos 300 pixéis do jogador. O estado de disparar só é ativado quando o jogador se encontra a pelo menos 200 pixéis.

```
// -- se o soldado estiver proximo do jogador fica em modo ataque e para de andar
if (
  (soldados[i].x - this.player.x <= 300 && direcao == -1) ||
  (this.player.x - soldados[i].x <= 300 && direcao == 1)
) {
  soldados[i].ataque = true;
  soldados[i].setVelocityX(0);
}

// -- se o jogador estiver longe do soldado, o soldado vem na sua direção
if (
  (soldados[i].x - this.player.x >= 350 && direcao == -1) ||
  (this.player.x - soldados[i].x >= 350 && direcao == 1)
) {
  soldados[i].ataque = false;   // não esta em modo ataque
  // -- Inicio: Condição para quando o soldado fez spawn
  if (!soldados[i].anims.isPlaying && soldados[i].stage == 0) {
    soldados[i].setVelocityX(80 * direcao);
    soldados[i].anims.play("soldadocorrer", true);
    soldados[i].stage = 1;
  }

  // -- Atingido : Condição para quando o soldado foi atingido à distancia
  if(soldados[i].anims.isPlaying && soldados[i].stage == 1) {
    soldados[i].setVelocityX(80 * direcao);
  }
  // -- Standard: Condição para quando o player se afasta do soldado,
  // espera que as animações cheguem ao fim e so depois pode voltar a se aproximar do jogador
  if(!soldados[i].anims.isPlaying && soldados[i].stage == 4){
    soldados[i].setVelocityX(80 * direcao);
    soldados[i].anims.play("soldadocorrer", true);
    soldados[i].stage = 1;
  }
}
```

Figura 9: Diferentes estados dos soldados.

Soldado Atingido

Quando o soldado é atingido por uma bala do jogador e perde 20 pontos de vida. Se já não tiver mais vida o soldado morre.

```
// -- quando soldado é atingido por 1 bala
soldadoAtingido(soldado, bala) {
    soldado.vida -= 20; // retira 20 pontos de vida ao soldado
    bala.destroy(); // elimina a bala
    soldado.setVelocityX(0); // para não ser afetado pela gravidade da bala

    // -- se o soldado perder a vida toda é eliminado
    if (soldado.vida <= 0) {
        soldado.destroy();
        this.somSoldadoMorre.play();

        this.player.score += 110;
        this.textScore.setText("Score\n" + this.player.score);
    }
}
```

Figura 10: Quando um soldado é atingido por uma bala.

Soldado Atingido por Granada

Quando o soldado é atingido por uma granada do jogador e perde 20 pontos de vida. Se já não tiver mais vida o soldado morre.

```
// -- quando soldado é atingido por 1 granada
soldadoExplodido(soldado, explosao) {
    soldado.vida -= 100; // retira 20 pontos de vida ao soldado
    soldado.setVelocityX(0); // para não ser afetado pela gravidade da bala

    // -- se o soldado perder a vida toda é eliminado
    if (soldado.vida <= 0) {
        soldado.destroy();
        this.somSoldadoArde.play();

        this.player.score += 90;
        this.textScore.setText("Score\n" + this.player.score);
    }
}
```

Figura 11: Quando o soldado é explodido por uma granada.

Jogador Atingido

Quando o jogador é atingido por uma bala inimiga, o jogador perde 20 pontos de vida. Se já não tiver mais vida, perde.

```
// -- quando o jogador é atingido por uma bala
playerAtingido(player, bala) {
    player.vida -= 20; // retira 20 pontos de vida ao jogador
    bala.destroy();
    this.cameras.main.flash(300);
    this.textVida.setText("Life\n" + this.player.vida);

    // -- se o jogador ficar sem vida morre
    if (player.vida <= 0) {
        this.physics.pause();

        if (this.musica.isPlaying) this.musica.stop();

        this.scene.start('GameOverScene');
    }
}
```

Figura 12: Quando o jogador é atingido por uma bala inimiga.

Jogador dispara bala

Disparo de uma bala pelo jogador.

```
fireBullet(time) {
    // so dispara uma nova bala algum tempo depois da ultima bala
    if (time > this.shootTimeOffset && this.numeroBalas > 0) {
        var bala;

        // dependendo do lado para o qual o jogador está virado, posiciona a bala na arma
        if (this.facingRight)
            bala = this.playerBullets.create(
                this.player.x + this.player.width / 2,
                this.player.y,
                "bullet"
            );
        else
            bala = this.playerBullets.create(
                this.player.x - this.player.width / 2,
                this.player.y,
                "bullet"
            );

        // aplica velocidade à bala, dependedo da direção do jogador
        bala.setVelocity(this.bulletSpeed * this.bulletDir, 0);

        this.playSomShoot(); // reproduz som do disparo

        this.shootTimeOffset = time + 100; // tempo para a proxima bala poder ser disparada

        this.numeroBalas -= 1;
        this.textNumBalas.setText("Bullets\n" + this.numeroBalas);

    }
}
```

Figura 13: Disparo de uma bala pelo jogador.

2.5. Resultado Final

Figura 14: Menu Inicial

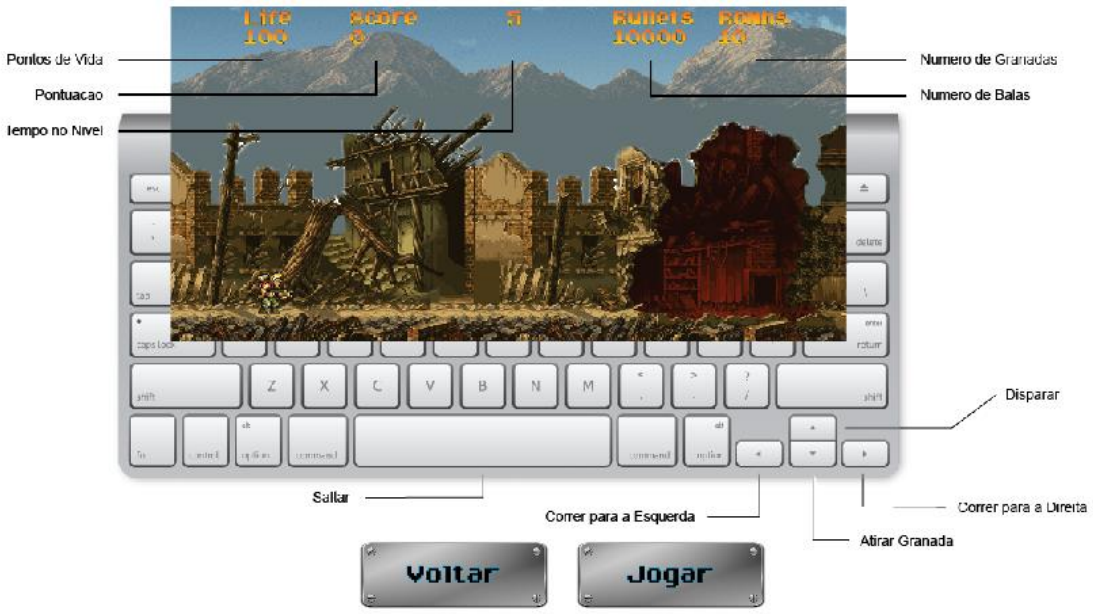

Figura 15: Janela de instruções

Figura 16: Janela de jogo.

Figura 17: Janela de Vitória.

2.6. Aspetos a Melhorar

Com o projeto finalizado, apenas melhoraria o seguinte:

- Mais animações para o soldado;
- Mais itens na janela (balas/granadas coletáveis, pontos, …);
- Um segundo nível;

- Possibilidade de retirar som ou dar muda ás músicas.

4. Referencia a material não original utilizado

Em termos de spritesheets, embora algumas tenham sido retiradas em pesquisas do google, a base foi retirada do site https://www.spriters-resource.com/neo_geo_ngcd/, aqui tem as spritesheets do jogo *Metal Slug* para as mais variadas coisas, o problema é que para alem de não estarem alinhadas, algumas não possuem transparência.

Deste modo, tentei conjugar as melhores spritesheets e editei no Photoshop e depois meti a posição de cada frame da animação num ficheiro json.

O mesmo aconteceu para os sons, mas tendo em conta que não tinham nomes e eram centenas, tive que pesquisar sons individualmente.

5. Conclusão

Dado por concluído o projeto, vejo que os objetivos principais foram concluídos, ficando apenas alguns detalhes secundários deixados de lado, ou menos trabalhados devido a falta de tempo (devidamente referenciados na secção de aspetos a melhorar).

Este projeto foi extremamente útil para a aprendizagem de phaser, uma vez que nas aulas, pouco aprendemos. E no projeto tivemos que pesquisar imenso para conseguir juntar as peças e desenvolver o jogo.

O que não foi fácil tendo em conta que o phaser 3 tem poucos meses e a comunidade online é muito baixa e os exemplos apesar de ajudarem imenso não contem tudo. E a documentação é extremamente fraca.

No entanto, este projeto elevou imenso o meu conhecimento sobre phaser, assim como sobre a programação de ambientes gráficos.