TP1 - CG

Jéssica Relva Josué Rocha Matheus Teles
Rhaynara Kristina

24 de Setembro de 2016

Sumário

1 Introdução

Este trabalho tem por objetivo apresentar o jogo 'BERZERK - CIVIL WAR' desenvolvido para a disciplina de Computação gráfica, lecionada pelo professor André Rodrigues da Cruz. 'BERZERK - CIVIL WAR' é uma releitura do jogo 'Berzerk', lançado para a plataforma Atari 2600 pela empresa Stern Electronics of Chicago em Agosto de 1982.

1.1 Objetivo do Jogo

O objetivo do jogo é chegar até a base do Império Robótico, passando por 5 bases inimigas, cada uma em uma região diferente do planeta, repletas de Robotroopers (tropa de base do Império Robótico), utilizando de reflexos rápidos

e de um arsenal poderoso para derrotar os seus inimigos que farão de tudo para se manterem no poder.

2 Como Jogar

2.1 Menus

Para se pular a animação inicial, basta pressionar a tecla 'Enter'.

Nos menus, pode-se selecionar as opções disponíveis (Jogar, Opções e Sair) com o mouse.

Opcionalmente, pode-se entrar no menu de Opções apertando a tecla 'O'. Para iniciar o jogo Pode-se pressionar a tecla 'Enter'. Para sair do jogo pode-se pressionar a tecla 'Esc'. Para retornar de menus, pode-se pressionar a tecla 'Backspace'.

Para alternar entre modo de visualização em janela ou em tela cheia, pode-se pressionar a tecla 'F'.

2.2 Jogo

Durante o jogo são utilizadas as seguintes teclas:

'W', 'A', 'S' e 'D' para movimentar o personagem nas direções Cima, Esquerda, Baixo e Direita, respectivamente.

Botão Esquerdo do mouse para atirar o primeiro tipo de arma.

Botão Direito do mouse para atirar o segundo tipo de arma.

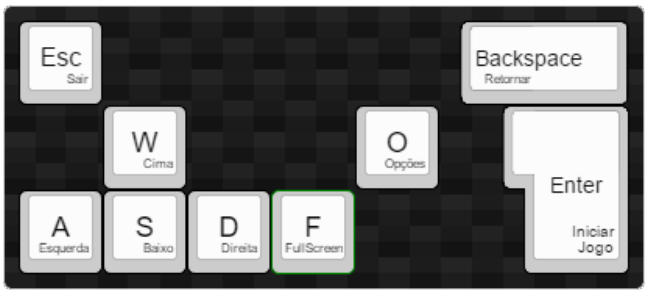

Figura 1: Teclas e Suas Funções

3 O jogo

3.1 Personagens

3.1.1 Heróis

Raven é uma cidadã berzerkiana que vive em uma pequena e escassa região de seu planeta BERZERK. Cresceu em uma família humilde e usava o pouco

que ganhava consertando robôs de classes mais baixas. Mal sabia Raven que quando se tornasse adulta, seu planeta seria tomado pelos robôs, e que sua vida mudaria completamente. O destino a uniu a um grupo de berzerkianos rebeldes, que já estavam cansados de serem dominados pelo poderoso Dark, um robô com tecnologia extremamente avançada, que fora programado desde sua construção para ser capaz de pensar e agir como um berzerkiano. Com uma sabedoria sobre tecnologia que ela mesma desconhecia, mal sabia que seria a única capaz de desativar Dark e restaurar o equilíbrio entre robôs e berzerquianos.

- Raven tem as seguintes características:
  - Dimensão = 25px X 25px;
  - Velocidade = Controlada pelo jogador;
  - HP = 100;
  - Cor = Laranja;

3.1.2 Inimigos

Em 'BERZERK - CIVIL WAR' existem 5 tipos de inimigos distintos que tentarão impedir o jogador ao longo dos níveis. Cada inimigo possui características diferentes, listadas a seguir:

- Inimigo 1
  - Dimensão = 25px X 25px;
  - Velocidade = 0.25px/frame;
  - HP = 40;
  - Formato = Quadrado;
  - Cor = Amarelo;
  - Raio de Visão = 500px;
- Inimigo 2
  - Dimensão = 15px X 15px
  - Velocidade = 0.30px/frame;
  - HP = 30;
  - Formato = Triangular;
  - Cor = Azul;
  - Raio de Visão = 500px;
- Inimigo 3
  - Dimensão = 15px X 15px
  - Velocidade = 0.20px/frame;

  - HP = 30;
  - Formato = Hexágono;
  - Cor = Rosa;
  - Raio de Visão = 300px;

- Inimigo 4
  - Dimensão = 30px X 30px
  - Velocidade = 0.25px/frame;
  - HP = 70;
  - Formato = Octógono;
  - Cor = Azul-Marinho;
  - Raio de Visão = 200px;

- Inimigo 5
  - Dimensão = 70px X 70px
  - Velocidade = 0.20px/frame;
  - HP = 300;
  - Formato = Circular;
  - Cor = Vermelho;
  - Raio de Visão = 300px;
  - Atravessa Paredes;

3.2 Recursos

3.2.1 Medkits

O jogo dispõe de kits médicos ("Medkits") para restaurar o hp do jogador. Estes "medkits"são encontrados entre as fases do jogo na forma de triângulos vermelhos. Para coletá-los, basta se movimentar até eles.

3.2.2 Pontuação

Cada inimigo derrotado contribui para a pontuação final. Inimigos diferentes têm valores de pontuação distintos quando derrotados.

3.3 As fases

O jogo é composto de 5 fases com layouts distintos, cada um contendo a própria disposição de inimigos e itens coletáveis definidos previamente. Seguem abaixo imagens do layout dos níveis.

- Fase 1 - Naboo

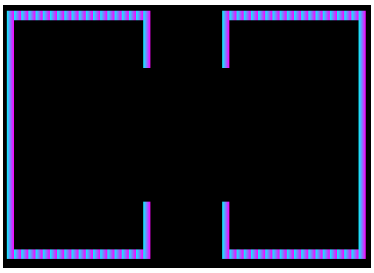

Figura 2: 1º nível

- Fase 2 - Kamino

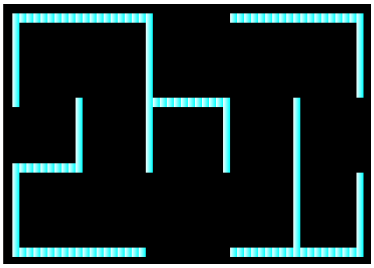

Figura 3: 2º nível

- Fase 3 - Saleucami

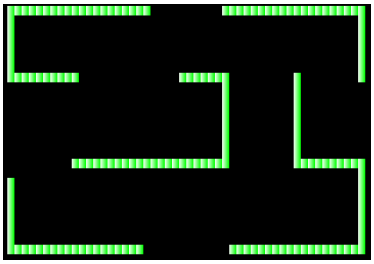

Figura 4: 3º nível

- Fase 4 - Alderaan

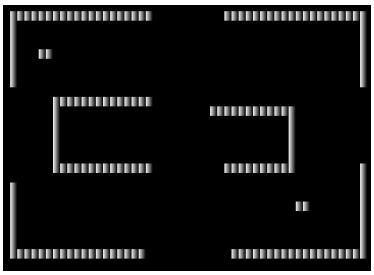

Figura 5: 4º nível

- Fase 5 - Hoth

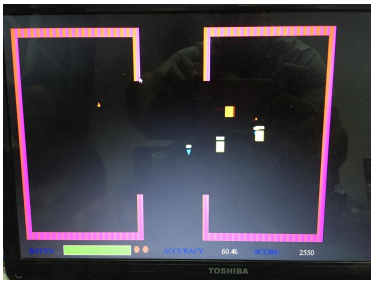

Figura 6:

4 Estruturas de Dados do Projeto

No projeto, são utilizadas diversas estruturas de dados para representar os atores do jogo e as funções que controlam o ambiente da aplicação. As classes e métodos mais significativos serão descritas abaixo.

(i) Estruturas Coord, Dimensao e Cor:

Com o objetivo de facilitar a representação de certos tipos de variáveis, optou-se por criar estruturas de dados para simbolizar as coordenadas do plano 2D (Estrutura Coord), as dimensões de um personagem (Estrutura Dimensao) e as cores em geral (Estrutura Cor).

```
1     typedef struct Coord {
2         float x, y;
3         Coord() {};
4         Coord(float x1, float y1) {
5             x = x1;
6             y = y1;
```

```
        }
    } Coord;

    typedef struct Dimensao {
        float largura, altura;
        Dimensao() {};
        Dimensao(float altura, float largura) {
            this->largura = largura;
            this->altura = altura;
        }
    } Dimensao;

    typedef struct Cor {
        float r, g,b;
        Cor() {};
        Cor(float r, float g, float b) {
            this->r = r;
            this->g = g;
            this->b = b;
        }
    } Cor;
```

(ii) Classe Personagem:

A classe Personagem é uma superclasse (Herdada por Player e Enemy) que incorpora as variáveis-chave para a representação do jogador e dos inimigos na tela. São elas:

- Coord c - Representa o ponto de referência do personagem.
- Dimensao d - Armazena a largura e a altura do personagem.
- bool isActive - Variável booleana para representar se um personagem está ou não ativo na cena atual.
- float Velocidade - Constante que modifica a velocidade de movimento do personagem.
- float hp - Representa a quantidade de pontos de vida do personagem.
- Cor cor - Armazena a cor do personagem.
- void Display() - Função responsável por desenhar o personagem na tela.

```
    class Personagem{
    public:
        Coord c;
        Dimensao d;
        bool isActive;
        float velocidade, hp, hpOriginal;
        Cor cor;
        Personagem(Coord c,Dimensao d,int velocidade,int hp,
            Cor cor);
        Coord Centro();
        void Display();
    }
```

(iii) Classe Projétil:

A classe Projétil é a classe que representa os projéteis atirados tanto pelo jogador quanto pelos inimigos ao longo do jogo.

- Coord c - Representa o ponto de referência do projétil.
- Coord vetor - Representa o vetor unitário que rege o movimento do projétil
- Dimensao d - Armazena a largura e a altura do projétil.
- bool isActive - Variável booleana para representar se um projétil está ou não ativo na cena atual.
- float velocity - Constante que modifica a velocidade de movimento do projétil.
- int alcance - Representa o alcance máximo do projétil.
- float dano - Representa a quantidade de pontos de vida retirados de um personagem atingido pelo projétil.
- Cor cor - Armazena a cor do projétil.

```
1   class Projectile{
2       public:
3       Coord c;
4       Coord vetor;
5       Dimensao d;
6       Cor cor;
7       float velocity;
8       int alcance, dano;
9       bool isActive;
10
11      Projectile(Coord r,Coord vect,Dimensao d,int alcanc,
            int dano,float velocity,bool isActive,Cor cor);
12      void Deactivate();
13      void Display();
14  }
```

(iii) Classe Coletável

A classe Coletável é a classe que representa os itens espalhados pela tela que podem ser coletados pelo jogador.

- Coord c - Representa o ponto de referência do coletável.
- Dimensao d - Armazena a largura e a altura do coletável.
- bool active - Variável booleana para representar se um coletável está ou não ativo na cena atual.
- Cor cor - Armazena a cor do coletável.

```
1       class Coletavel {
2           public:
3           Coord c;
```

```
4               Cor cor;
5               Dimensao d;
6               bool active;
7               Coletavel();
8               Coletavel(Coord c, Cor cor, Dimensao d, bool
                    status);
9               void Display();
10          }
```

(iv) Classe Tela A classe tela é a classe que contém a maioria das operações das regras do jogo. Esta classe é responsável por receber as entradas do programa, converter as entradas em operações na cena atual e checar colisões entre as entidades.

- int contadorTempo - Contador para limitar a velocidade com que os inimigos podem atirar.
- int faseAtual - Flag para identificar qual fase se está jogando no momento.
- Coord mouse - Variável para armazenar as coordenadas do mouse.
- bool keyBuffer[5] - Vetor de flags para verificar se as teclas 'W', 'A', 'S', 'D' e o clique do mouse foram pressionados.
- int shoot - Variável para controlar o tiro do jogador.
- bool fullscreen - Variável que verifica o estado da tela (Fullscreen ou Janela).
- Score score - Variável que armazena a pontuação do jogador.
- std::vector <Projectile*> vetorProjeteisAmigos - Vetor que armazena todos os projéteis gerados pelo jogador.
- std::vector <Projectile*> vetorProjeteisInimigos - Vetor que armazena todos os projéteis gerados pelos inimigos.
- std::vector <Enemy*> enemyBuffer - Vetor que armazena todos os inimigos na tela.
- std::vector <Hiscore*> pontuacoes - Vetor que armazena as maiores pontuações.
- Player player - Referência para o jogador.
- MainMenu menu - Referência para o menu principal.

```
1           class Tela {
2               public:
3               //variaveis de controle
4               int contadorTempo = 0;
5               int faseAtual;
6               Coord mouse;
7               bool keyBuffer[5]; //Respectivamente up, down,
                    left, right, click
8               int shoot;
```

```
9             bool fullscreen;
10
11            //Pontuacao
12            Score score;
13
14            //Personagens
15            std::vector <Projectile*> vetorProjeteisAmigos;
16            std::vector <Projectile*> vetorProjeteisInimigos;
17            std::vector <Enemy*> enemyBuffer;
18            std::vector <EfeitosVisuais*> vetorEfeitosVisuais;
19            std::vector <Hiscore*> pontuacoes;
20            Player player;
21
22            //Animacoes
23            AnimacaoInicial animacaoInicial;
24
25            std::vector <Level*> fases;
26            MainMenu menu;
27
28            glutWindow janela;
29
30            Tela();
31            //Metodos[...]
32        }
```

(v) Classe Menu Principal

- As variáveis booleanas emJogo, emTela, emMenu, emMenuPrincipal e opcao representam qual estado do menu principal está representado num dado tempo, e são utilizadas para controlar as funções de desenho.
- As variáveis booleanas iniciarMenuButton, opcaoMenuButton, desenharMenuButton e sairButton checam se os botões Iniciar, Opção e Sair foram pressionados.
- A variável clickAtivo verifica se o botão do mouse foi pressionado, e a variável Coord mouse guarda a coordenada onde foi clicado.
- As funções DesenhaTextoMenu(char * string, Coord c), DesenhaInstrucoes(char * string, Coord c) e DesenhaInstrucoesBloco(char * string, Coord c) são responsáveis por desenhar os menus.

```
1         class MainMenu {
2         public:
3             // VARIAVEIS DE MENU
4             bool emJogo;
5             bool emTela;
6             bool emMenu;
7             bool emMenuPrincipal;
8             bool inicia;
9             bool opcao;
10
11            //FLAGS DE FUNCOES
12            bool iniciarMenuButton;
```

```
13              bool opcaoMenuButton;
14              bool desenharMenuButton;
15              bool sairButton;
16
17              //MOUSE
18              bool clickAtivo;
19              Coord mouse;
20
21              MainMenu();
22              void Run();
23              void DesenhaTextoMenu(char * string, Coord c);
24              void DesenhaInstrucoes(char * string, Coord c);
25              void DesenhaInstrucoesBloco(char * string, Coord c
                    );
26              void OpcaoMenu();
27              void IniciarJogo();
28              void Display();
29              void OpcaoClick(int op);
30              void PosicaoMenu();
31          }
```

(vi) Classe Nível

A classe nível é responsável por agrupar todas as estruturas que compõe uma fase do jogo, importadas do arquivo de texto.

- int contParedes - Variável responsável por armazenar a quantidade de paredes lidas.
- std::vector <Enemy*> vetorInimigos - Lista de inimigos que estarão na cena.
- std::vector <Coord*> vetorParede - Lista que armazena a referência de todas as paredes da cena.
- std::vector <Coletavel*> vetorColetaveis - Lista que armazena a os coletáveis disponíveis na cena.
- int matTela[26][50] - Matriz que representa a tela (800x600) dividida em blocos.
- void LerArquivo(std::string url) - Função responsável por ler o arquivo .txt e preencher as variáveis da classe.
- void DesenhaFase() - Função responsável por plotar na tela todas as entidades representadas pela classe Level na tela.

```
1           class Level
2           {
3               public:
4               int contParedes;
5
6               //Vetores de inimigos e paredes
7               std::vector <Enemy*> vetorInimigos;
8               std::vector <Coord*> vetorParede;
9               std::vector <Coletavel*> vetorColetaveis;
10              int matTela[26][50];
```

```
11              Level();
12
13              Level(int tipo);
14              void DesenhaFase();
15              void LerArquivo(std::string url);
16              void MapeiaFase();
17          }
```

Lista de Figuras

Referências

[1] 'Berzerk - Coins Detected in Pocket!' disponível em: http://thedoteaters.com/?bitstory=berzerk

[2] 'Movement using OpenGL/Glut' disponível em: https://unpolishedcreations.com/2014/09/08/movement-using-openglglut-2/