Trabalho Prático - Final Fantasy

Pedro O.S. Vaz de Melo

July 8, 2021

1 Descrição do Problema

O objetivo deste trabalho é fazer com que o aluno utilize as técnicas de programação aprendidas na disciplina para desenvolver um jogo eletrônico gráfico semelhante às versões clássicas (antigas) da franquia *Final Fantasy*. O jogador comanda um grupo de personagens nos jogos Final Fantasy enquanto eles progridem pela história ao explorar o mundo e derrotar oponentes. Inimigos são tipicamente encontrados randomicamente através da exploração. Os jogadores emitem ordens como "Lutar", "Magia" e "Item" para personagens individuais enquanto combatem através de uma interface de menu. As batalhas são organizadas em rodadas, com os protagonistas e os antagonistas em lados diferentes do campo de combate.

Você pode jogar o *Final Fantasy VI* através deste link: `https://www.retrogames.cz/play_900-SNES.php?language=EN`. Um vídeo com todo o *gameplay* do jogo pode ser visto através deste link: `https://www.youtube.com/watch?v=XF2cski7Q7M`.

Nesse trabalho, você vai implementar, de forma bem rudimentar, duas funcionalidades desse jogo: um *modo de exploração* e um *modo de batalha*. No *modo de exploração*, você deve criar um cenário estático para o personagem navegar. O personagem inicia a exploração no canto inferior esquerdo do cenário e deve chegar até uma posição alvo que fica no canto oposto, ou seja, no canto superior direito do cenário. Ao chegar no alvo, que pode ser desenhado como uma caverna (ver exemplo disponibilizado pelo professor), o jogo termina com sucesso e a pontuação do jogador é exibida. Se a pontuação for a maior pontuação obtida até o momento, uma mensagem indicando "novo recorde" deve ser exibida e a pontuação deve ser registrada em arquivo.

O jogador obtém pontos sempre que ele derrotar inimigos em batalhas. Antes de começar o jogo, você deve sortear coordenadas (x, y) do cenário para esconder os inimigos. Você deve esconder pelo menos 18 inimigos. Sempre que o personagem estiver próximo (exemplo: distância entre as coordenadas do inimigo e do personagem é menor que 16 pixeis) a uma dessas coordenadas, você deve iniciar o *modo de batalha* do personagem contra esse inimigo.

No *modo de batalha*, o personagem e o seu inimigo intercalam as suas ações. O personagem deve ter pelo menos três ações: (1) *ataque*, (2) *especial* e (3) *fugir*. A primeira é um ataque simples, com dano previsível. A segunda é um ataque especial, que pode levar a danos imprevisíveis. Se preferir, você também pode implementar o *especial* como um ataque de dano maior que pode ser usado por um número limitado de vezes. Por fim, se o jogador optar pela opção *fugir*, a batalha é interrompida com uma probabilidade de $P_{fuga}\%$. Você pode escolher o valor de $P_{fuga}\%$, mas ele precisa ser maior que 10% e menor que 90%. Depois da ação do personagem, é a vez do inimigo, que deve realizar, no mínimo, um ataque simples. Esse ataque deduz os pontos de vida do personagem e deve ter um poder aleatório cada vez que ele for lançado.

A batalha termina caso o personagem ou o inimigo morram ou, em outras palavras, seus pontos de vida fiquem menores ou iguais a zero. Se o inimigo morrer ou o personagem conseguir fugir, o modo de navegação é reativado. No caso do inimigo morrer, ele não deverá ativar mais o *modo de batalha* no *modo de exploração*. No caso do personagem conseguir fugir, o inimigo ativará novamente o *modo de batalha* caso o personagem fique próximo a ele no *modo de exploração*[1]. Se o personagem morrer, o jogo termina. Por fim, é importante que os inimigos tenham diferentes níveis de poder e que você represente visualmente essas diferenças. No mínimo, inimigos devem ter diferentes valores de pontos de vida e também de dano. Veja o exemplo do jogo disponibilizado pelo professor.

2 Critérios de Avaliação

Este jogo pode ser tão complexo quanto você deseja. A versão disponibilizada pelo professor, por exemplo, contém diversas funcionalidades extras, que se implementadas, vão levar a um trabalho que supera os 20

[1] Para que seu jogo não ative o *modo de batalha* com o mesmo inimigo sempre o personagem fugir, faça com que o personagem se afaste do inimigo no *modo de exploração* sempre que ele fugir.

pontos.

Abaixo algumas funcionalidades que devem (ou podem) ser implementadas no *modo de exploração* do jogo e os seus respectivos valores em pontos:

1. **Controle preciso do movimento do personagem.** O personagem deve ser capaz de se movimentar para todas as direções e não deve ultrapassar os limites da tela. *(1 ponto)*
2. **Direção do movimento do personagem.** O desenho do personagem deve alterar de acordo com a direção do seu movimento. *(1 ponto)*
3. **Desenho e tratamento do alvo.** O cenário deve conter um alvo para o personagem alcançar que, quando alcançado, o jogo termina. *(2 pontos)*
4. **Disposição dos inimigos aleatoriamente no cenário.** Pelo menos 18 inimigos devem ser dispostos aleatoriamente no cenário. Sempre que o personagem estiver próximo a um deles, o *modo de batalha* deve ser iniciado. *(2 pontos)*
5. **Inimigos mortos não devem ser reativados.** Caso um inimigo seja derrotado no *modo de batalha*, ele não deve ativar uma batalha novamente no *modo de exploração*. *(1 ponto)*

Abaixo algumas funcionalidades que devem (ou podem) ser implementadas no *modo de batalha* do jogo e os seus respectivos valores em pontos:

1. **Menu de opções do jogador.** Exibição do menu com pelo menos as três opções descritas na seção anterior. O jogador deve ter acesso a um *cursor* que se movimenta pelas opções a partir do teclado (exemplo, usando as teclas ↑ e ↓), indicando qual opção ele estará escolhendo caso ele aperte a tecla `ENTER`. *(2 pontos)*
2. **Desenho do personagem.** Exibir o personagem com os seus respectivos pontos de vida restantes. Os pontos de vida podem ser representados por texto ou graficamente, como no exemplo disponibilizado pelo professor. *(1 ponto)*
3. **Desenho do inimigo.** Exibir o inimigo com os seus respectivos pontos de vida restantes. Os pontos de vida podem ser representados por texto ou graficamente, como no exemplo disponibilizado pelo professor. *(1 ponto)*
4. **Inimigos com diferentes níveis de poder.** Os inimigos devem ter diferentes níveis de poder, pelo menos na quantidade de dano que podem gerar e absorver (quantidade de pontos de vida). *(2 pontos)*
5. **Ataque do personagem.** Animação e efeitos do ataque do personagem. *(1 ponto)*
6. **Ataque do inimigo.** Animação e efeitos do ataque do inimigo. *(1 ponto)*
7. **Especial do personagem.** Animação e efeitos do ataque especial do personagem. *(1 ponto)*
8. **Fuga do personagem.** Fuga do personagem, que deverá ser bem sucedida com probabilidade P_{fuga}. Nesse caso, o inimigo ainda deve estar disponível para batalha no *modo de exploração*. *(1 ponto)*

Abaixo algumas funcionalidades gerais que devem (ou podem) ser implementadas no jogo e os seus respectivos valores em pontos:

1. **Pontuação.** Exibição e contabilização da pontuação do personagem. A cada inimigo derrotado, a pontuação deve aumentar. *(2 pontos)*
2. **Fim de jogo.** O jogo deve terminar quando o personagem morrer ou quando ele alcançar o alvo. Uma tela indicando o fim e o resultado do jogo (vitória ou derrota) deve ser exibida. *(1 ponto)*
3. **Recorde.** A maior pontuação registrada (recorde) deve ser armazenada em um arquivo. O valor do recorde deve ser exibido sempre que o personagem terminar o jogo de forma vitoriosa. Se além disso ele também bater o recorde, uma mensagem informativa deve ser apresentada para ele. *(3 pontos)*
4. **Documentação.** Deve conter o manual de uso, que descreve como operar o jogo, e detalhes da implementação, que descreve brevemente os trechos de código e as estruturas de dados desenvolvidas por você. Exemplos de documentação podem ser baixados na página da disciplina ou através do link `https://drive.google.com/open?id=1KPl5y2DVEZqTW-Rrhor5SFhGLffGOJOr`. *(2 pontos)*

2.1 Conhecimento do Código

Conhecimento do aluno sobre o código apresentado será verificado via prova oral, que será dada no formato de uma entrevista. Sua nota total será multiplicada pela sua nota da prova oral, que vale 1. Assim, se você tirar 0.5 na prova oral, sua nota será dividia por 2.

2.2 Pontos Extras

Além dos pontos acima, o professor pode atribuir até 10 pontos a mais caso o aluno implemente extras, tais como:

1. Usar imagens e animações do tipo *sprite*;
2. Gerar diferentes tipos de cenários;
3. Permitir diferentes tipos de ataques, que produzem efeitos diferentes;
4. Colocar sons e músicas;
5. Implementar animações para o movimento das ações;
6. Implementar fases;
7. Criar *addons* e *power-ups* que podem, por exemplo, restaurar os pontos de vida do jogador ou ataques mais poderosos;
8. Implementar modo com mais de dois jogadores;
9. Implementar opção de salvar o jogo;
10. Implementar mais de um personagem;
11. Implementar mais ações;
12. Permitir mais de um inimigo ao mesmo tempo no *modo de batalha*.
13. Implementar diferentes tipos de monstros, com ataques diferentes;
14. Implementar uma estória, com enredo e objetivos;
15. Implementar um cenário contínuo, ou seja, a tela se movimenta para os lados se os jogadores forem para esse lado;
16. **Qualquer outro extra que você ache interessante!**

IMPORTANTÍSSIMO: Pontos extras só serão dados aos alunos que obtiveram mais de 50% **dos pontos nas provas, ou seja, mais de** 36 **no somatório das três provas.**

3 Como eu faço?

Apesar da descrição fazer o trabalho parecer complicado, ele é bastante simples. Tudo que o aluno precisa saber para desenvolver este jogo são os conhecimentos adquiridos na disciplina e um pequeno entendimento de desenvolvimento de aplicações gráficas. Assim como são necessárias bibliotecas novas para a utilização de funções não nativas da linguagem C, como a math.h, uma biblioteca também é necessária para que se utilize funções gráficas. Para este trabalho, pede-se que se utilize a biblioteca Allegro5, que fornece inúmeras funções que podem ajudar no desenvolvimento deste trabalho. Os vídeos abaixo ensinam como instalar a biblioteca Allegro5 em um ambiente Windows com o MingW instalado:

`https://www.youtube.com/watch?v=AezxBP687n8`
`https://www.youtube.com/watch?v=cgqjzJzm00w`

4 Roteiro de Desenvolvimento Sugerido

Como o jogo é complexo, identificar a sequência de funcionalidades que devem ser desenvolvidas pode ser um problema. Assim, a seguir estão descritas etapas de desenvolvimento sugeridas, colocadas em ordem cronológica.

1. (*modo de exploração*) Implementar o desenho do cenário;
2. (*modo de exploração*) Implementar a estrutura com informações do personagem;
3. (*modo de exploração*) Implementar o desenho do personagem e o seu movimento pelo cenário;
4. (*modo de exploração*) Implementar o desenho do alvo e tela de fim de jogo;
5. (*modo de exploração*) Implementar a estrutura com informações dos inimigos;
6. (*modo de exploração*) Implementar a disposição aleatória dos inimigos no cenário;
7. (*modo de exploração*) Implementar a verificação da proximidade do personagem com os inimigos. Para testar se o seu método está funcionando, imprima "Inimigo encontrado!" (com `printf`) sempre que o personagem estiver próximo de um inimigo;
8. (*modo de batalha*) Implementar o desenho do cenário. Invocar este cenário sempre que um inimigo for encontrado no *modo de exploração*;
9. (*modo de batalha*) Implementar o desenho do personagem.
10. (*modo de batalha*) Implementar o desenho do inimigo encontrado no *modo de exploração*;
11. (*modo de batalha*) Implementar o menu de opções;
12. (*modo de batalha*) Implementar o ataque do personagem;
13. (*modo de batalha*) Implementar o ataque do inimigo;
14. (*modo de batalha*) Implementar o ataque especial do personagem;
15. (*modo de batalha*) Implementar a volta para o *modo de exploração* quando o personagem ou o inimigo morrerem;
16. (*modo de batalha*) Implementar a fuga do personagem;
17. Implementar o sistema de pontuação;
18. Implementar o armazenamento do recorde;
19. Divirta-se implementando funcionalidades extras;
20. Escrever a documentação.

Um vídeo tutorial ensinando a desenvolver alguns dos primeiros itens desse roteiro estará disponível na página da disciplina a partir do dia 23 de julho de 2021.