Trabalho Prático - Canyon Bomber

Pedro O.S. Vaz de Melo

October 30, 2023

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 ao jogo *Canyon Bomber*. O objetivo do jogo é criar uma versão para dois jogadores, onde o vencedor será determinado pelo número de pontos conquistados. Os pontos serão obtidos quando os jogadores acertarem alvos localizados na parte inferior da tela, usando um veículo controlado por eles. Além disso, os alunos devem incorporar a regra de que se um jogador disparar um total de três tiros sem acertar qualquer alvo, ele perderá o jogo.

2 Critérios de Avaliação

Este jogo pode ser tão complexo quanto você deseja, mas há uma versão básica que lhe garante os 20 pontos do trabalho prático. Abaixo as funcionalidades que devem ser implementadas na versão básica do jogo:

- **Movimento da nave.** A nave deve se mover de um canto lateral da tela para o outro canto. Depois de atravessar a tela completamente, ela faz o caminho de volta. Esse padrão deve se repetir até o jogo terminar. *(1 ponto)*
- **Tiro.** O jogador deve ter acesso a uma tecla que dispara o tiro da nave que ele controla. O tiro deve sair da nave no momento que a tecla foi apertada. Se a tecla é apertada novamente enquanto o tiro anterior ainda estiver na tela, esse some da tela e um novo tiro é disparado (como no jogo original). *(2 pontos)*
- **Velocidade do tiro.** A velocidade horizontal do tiro deve ser igual à velocidade horizontal da nave. A velocidade vertical do tiro deve sofrer um efeito de aceleração similar ao efeito da gravidade. Em outras palavras, o movimento do tiro deve seguir o padrão de um movimento de projétil. *(2 pontos)*
- **Dois jogadores.** O jogo deve conter duas naves, cada uma delas controlada por um jogador. As naves devem ter sempre as mesmas velocidades e devem sempre percorrer a mesma distância de um lado ao outro da tela. Além disso, elas sempre devem se mover em direções opostas. Inicialmente, uma nave fica acima da outra. No entanto, depois de cruzar toda a tela, as naves poderão trocar de posição vertical. Considere que a probabilidade delas trocarem de posição vertical é de 50%. *(3 pontos)*
- **Diferentes tipos de naves.** Cada vez que uma nave atravessar completamente a tela, ela poderá mudar o seu tipo. Um tipo, na forma mais simples, pode ser simplesmente a sua velocidade. Sempre que uma nave mudar de tipo, a outra também muda e sempre muda para o mesmo tipo da sua rival. *(2 pontos)*
- **Alvos.** Os alvos dos jogadores devem estar situados na parte inferior da tela e dispostos em um formato de grid (matriz). Alvos situados na mesma linha devem ter a mesma cor e valer a mesma quantidade de pontos. Alvos situados em linhas inferiores devem valer mais pontos que alvos situados em linhas superiores. O seu jogo deve usar um grid com pelo menos 8 linhas e 20 colunas. *(2 pontos)*
- **Colisão.** Quando um tiro passar sobre uma região de um alvo ativo, esse alvo deve sumir da tela e o seu valor em pontos deve ser somado à pontuação do jogador que o acertou. Depois de acertar um alvo, o tiro continua ativo e apto a acertar outros alvos que estejam em sua trajetória. *(2 pontos)*

- **Vidas.** Cada jogador pode errar no máximo 3 tiros. Um tiro é considerado errado quando não acerta nenhum alvo. O número de tiros que o jogador ainda pode errar deve ser exibido na tela. *(2 pontos)*

- **Fim de jogo.** Quando não houver mais alvos na tela ou quando um jogador errar 3 tiros, o jogo deve terminar. *(1 ponto)*

- **Pontuação.** O jogo deve exibir a pontuação dos jogadores. *(1 ponto)*

  item **Exibição do histórico das partidas.** O histórico das partidas (número de vitórias de cada jogador)realizadas entre os jogadores deve ser armazenado em um arquivo. No final do jogo, esse histórico deve ser exibido na tela, incluindo também o resultado da última partida que jogaram. *(2 pontos)*

- **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. Implementar diferentes tipos de tiros
4. Implementar tipos de naves diferentes além do que é pedido no enunciado;
5. Colocar sons e músicas;
6. Implementar animações para o movimento das colisões;
7. Implementar fases;
8. Implementar a opção de mais de dois jogadores.
9. Criar *addons* e *power-ups* que podem, por exemplo, dar ações especiais para o jogador;
10. Implementar mais de um tipo alvo;
11. Implementar obstáculos nos cenários;
12. Implementar menus e diferentes níveis de dificuldade;
13. **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. Desenhar o cenário.
2. Criar a estrutura da nave.
3. Desenhar uma nave na tela.
4. Implementar o movimento dessa nave.
5. Implementar o desenho e o movimento da segunda nave.
6. Implementar os diferentes tipos de naves e fazer com que as naves troquem de tipo aleatoriamente ao atravessar a tela.
7. Implementar a troca aleatória de posição vertical das naves.
8. Criar a estrutura para o tiro.
9. Implementar o disparo do tiro: desenho, disparo a partir da tecla e movimento.
10. Implementar a estrutura para o alvo.
11. Criar a matriz de alvos e inicializar os seus valores de acordo com o que é pedido neste enunciado.
12. Desenhar os alvos na tela.
13. Implementar a colisão do tiro com o alvo.
14. Implementar o sistema de pontuação.
15. Exibir a pontuação na tela.
16. Implementar o sistema de vidas (os 3 tiros que cada jogador pode errar).
17. Exibir as vidas na tela.
18. Implementar o fim de jogo.
19. Exibir uma tela de fim de jogo na tela.
20. Implementar sistema de registro do histórico de partidas.
21. Exibir o histórico das partidas na tela de fim de jogo.
22. Escrever a documentação.