

Trabalho Prático 1 - GORILLAS.BAS



Agora que você conhece um pouco mais sobre OpenGL, vamos implementar um jogo digital simples usando seus conhecimentos sobre ele e uma linguagem de programação à sua escolha¹ (C/C++, Java, etc.).

Jogos de artilharia datam de antes mesmo do surgimento dos computadores pessoais, na década de 70. A ideia básica é lançar ~~bananas explosivas~~ projéteis nos ~~gorilas~~ adversários para ~~explodi-los~~ explodi-los. Uma versão que ficou bastante famosa foi o jogo Gorillas (1981), que vinha "instalado" nos computadores pessoais da IBM e foi [escrito em qBasic](#).

Dois jogadores alternam o turno, sendo que em sua vez, cada um define o ângulo e a força de lançamento do projétil. Se o projétil atinge o adversário, o jogador que acertou é vitorioso.

Neste trabalho, vamos criar um jogo de artilharia com a mesma mecânica dos jogos clássicos, mas você está livre para definir o seu tema - portanto, seja ~~fanfarrão~~ criativo!

¹Se optar por outra linguagem diferente de C ou C++, converse como professor sobre isso **antes de começar** ;)

Instruções sobre o jogo

O jogo consiste em uma **câmera fixa** em um ambiente 2D e **personagens para os dois jogadores**. Existe um cenário (e.g., chão) onde os personagens ficam.

A cada turno, um jogador escolhe valores para ângulo e força. A interação para se escolher os valores deve ser **feita por meio do teclado** (mas não pelo terminal).

Assim que o projétil atinge o cenário (e.g., o chão), o turno passa para o outro jogador. Se o projétil atinge um jogador, ele perde o jogo e o outro ganha. Não é necessário provocar a destruição do cenário.

Finda uma partida, deve aparecer uma mensagem perguntando se o jogador deseja jogar novamente ou sair do jogo.

O uso de texturas nesse trabalho é obrigatório. Utilize-as tanto para dar vida ao ambiente 2D do campo de visão do jogador quanto estilizar o personagens e o cenário.

Além do controle do personagem via teclado, os seguintes comandos devem ser implementados:

- Ao clicar na tecla *p*, o jogo deve **pausar/continuar**;
- Clicando em *r*, o jogo deve ser **reiniciado**;
- E clicando em *esc*, o jogo deve ser **encerrado**.

Atenção: Uma mensagem de confirmação deve ser exibida para as ações de reiniciar e encerrar o jogo.

Deve ser implementada um HUD que mostre o ângulo e força antes do disparo e quem é o jogador que está jogando no turno corrente.

Para fim de colisão, os projéteis e os personagens podem ser considerados todos retangulares ou circulares.

A lógica do jogo implementada, o uso de textura, a interação do teclado, a HUD de ângulo e força e a criatividade na implementação dos itens mencionados equivalem a um total de **70%** da nota do trabalho. Para

conseguir mais pontos, você pode implementar algumas das funcionalidades adicionais no seu jogo. Essas funcionalidades serão avaliadas de acordo com a **difículdade de implementação**, o **efeito obtido** dentro do jogo e a **qualidade da implementação**.

Implementando-as, você pode obter até **125%** da pontuação do trabalho!

Exemplos de funcionalidades com suas respectivas pontuações **máximas** são mostrados a seguir:

- Relativas a **texturas**:

1. **Texturas animadas (até 10%)**: você pode criar animações para as texturas tanto para o cenário quanto para os TPs e personagem. **#diquentinha**: busque por *sprite sequence* na Internet

- Personagem morrendo (**3%**)
- Personagem atirando (**2%**)
- Projétil voando (**2%**)
- Projétil explodindo (**3%**)

- Relativas ao **cenário**:

1. Cenário **não plano (até 10%)**: em vez de um plano, o cenário pode ser montanhoso (com curvas) ou com platôs

- Neste caso, os projéteis devem colidir devidamente com o cenário
- Aleatório (+**4%**): cada partida é diferente da outra

2. Cenário **destrutível (até 10%)**

- Simples (**até 6%**): o cenário destrói, mas não "respeita a gravidade"
- Complexo (**até 12%**): o cenário e os jogadores se adaptam à destruição segundo a gravidade

- Relativas aos **personagens e projéteis**:

1. **Movimentação (até 3%)**: personagem pode andar uma distância \times para a direita ou para esquerda antes de atirar

2. **Vidas (4%)**: em vez de morrer no primeiro tiro, um jogador pode ter vidas

- A HUD deve mostrar quantas vidas cada jogador ainda tem

3. **Projétil orientado (5%)**: faça o projétil rotacionar na direção da sua velocidade

- Relativas ao **jogo**:

1. **Vento (até 12%)**: força eólica que altera a trajetória dos projéteis

- Deve mudar aleatoriamente a cada turno
- A HUD deve mostrar a direção e força do vento

2. **Inteligência artificial (até 6%)**: torne possível jogar contra o computador

- Outras funcionalidades:

1. **Modo *cheat* (6%)**: ativado/desativado pela tecla 'c', faz com que a trajetória do projétil seja mostrada antes dele ser atirado

2. **Manter razão de aspecto (4%)**: faça com que a razão de aspecto do jogo seja sempre mantida, independente das dimensões da janela, mas que o jogo ocupe a maior área possível da janela

3. **Telas (até 8%)**: faça um jogo completo, ou seja, implemente telas de *splash screen*, menu inicial, créditos, opções, *game over*, etc

4. **Sons (até 8%)**. Colocar efeitos sonoros e música de fundo no seu jogo

5. **Implementação criativa (?)**: qualquer implementação que não fuja muito do pedido, mas que traga elementos novos e interessantes para o seu jogo é bem-vinda!

Um lembrete importante

Preocupe-se **primeiro em implementar as funcionalidades básicas do trabalho!** Deixe o embelezamento do trabalho e a implementação das funcionalidades extras para somente quando você já possuir a base lógica do trabalho construída e funcionando.

Instruções gerais

O seu código deve estar comentado e, principalmente, **organizado**: ao construí-lo, pense que outra pessoa irá ler o código e você não estará lá

para explicar seu raciocínio, portanto, organize-o! Também não é necessário comentar o código inteiro, mas o faça quando sentir necessidade de uma explicação adicional à sua lógica.

Seu trabalho pode ser feito **individual ou em duplas** e produzido integralmente por você/dupla. Se imagens de terceiros forem usadas, coloque *links* para elas na documentação. A discussão e troca de ideias com os colegas é bem-vinda e estimulada, mas cada aluno/dupla deve ter seu próprio trabalho.

Trabalhos muito semelhantes receberão nota 0, independente de quem copiou quem. E claro, trabalhos semelhantes aos de outras pessoas ou retirados da Internet, também receberão nota 0.

Outros descuidos também o farão **perder pontos no trabalho**, como:

- Seu trabalho não executa: nota 0;
- Seu trabalho é uma cópia (como já mencionado): nota 0;
- Você não implementou os itens obrigatórios;
- Ausência de algum item obrigatório no que deve ser entregue (descritos a seguir);
- Baixa legibilidade/organização do código;
- Baixa qualidade da implementação;
- Entregar fora do prazo. Cada dia de atraso reduz o valor máximo de nota de acordo com a equação abaixo, de modo que x representa o número de dias de atraso e $f(x)$ equivale à penalidade percentual da nota:
 - Isso implica que 1 ou 2 dias de atraso são pouco penalizados
 - E após 5 dias de atraso, o trabalho vale 0
 - *Seeing is believing*: [https://www.google.com.br/search?q=y%3D\(2%5E\(x-2\)%2F0.16\)%2Cy%3D100](https://www.google.com.br/search?q=y%3D(2%5E(x-2)%2F0.16)%2Cy%3D100)

O que deve ser entregue

Você deve entregar um **arquivo .tar.gz, .7z ou .zip** via **Moodle** contendo os seguintes itens:

1. Pelo menos três *screenshots* de diferentes cenas de seu jogo;
2. Todo o programa fonte, com *makefiles* e bibliotecas necessárias para a compilação e execução do programa;
3. O arquivo executável do jogo;
4. Um arquivo **README** contendo (a) instruções para a compilação e execução, e (b) a lista de itens adicionais implementados em seu jogo.
5. (Opcional bônus, **5%**) Um *link* para um **vídeo curto** no YouTube, Vimeo, etc. mostrando seu jogo implementado!
 - Faça um **vídeo curto**!

Qualquer dúvida, entre em contato com o professor ou com o monitor. Ou então acrescente a sua interpretação no arquivo README e mãos à obra!