| | |
|---|---|
| | Centro Tecnológico<br>Departamento de Informática |
| Disciplina: Computação Gráfica | Código: INF09282 e INF09284 |
| Prof. Thiago Oliveira dos Santos | |

# Trabalho 3D

## 1 Introdução

Esse trabalho tem como objetivo fixar as técnicas de computação gráfica 3D adaptando o trabalho anterior, T2D, para 3 coordenadas.

O jogo será composto por dois jogadores e uma arena com alguns elementos (obstáculos) que interagirão com o jogador. Os personagens dos jogadores serão controlados pelo teclado, sendo que o primeiro jogador poderá usar também o mouse, e poderão atirar. O objetivo do jogador é matar o jogador inimigo. O trabalho deverá ser implementado em C++ (ou C) usando as bibliotecas gráficas OpenGL e GLUT (freeglut).

## 2 Especificação das Funcionalidades

Ao rodar, o programa deverá ler e interpretar os elementos da arena do arquivo do tipo SVG informado pela linha de comando. O jogo será apresentado em uma janela de 800x500, metade, 400x500, para a visão de um jogador e metade, 400x500, para a do outro. A arena será composta por uma série de elementos (ver Figura 1): uma arena representada sempre por um círculo azul; um círculo verde representando sempre o personagem do jogador 1 (controlado pelo mouse e teclado); um círculo vermelho representando sempre o personagem do jogador 2 (controlado teclado apenas) e círculos pretos representando obstáculos. Um arquivo SVG será fornecido como exemplo juntamente com a descrição do trabalho, porém é responsabilidade do aluno testar outros arquivos com configurações (posições e tamanhos) diferentes para os elementos. A leitura do SVG poderá ser feita utilizando-se um parser para XML. Sugiro utilizar a Tinyxml que é simples e pode ser enviada juntamente com o código para ser compilada (não é permitido usar bibliotecas pré-compiladas). Use o Inkscape para visualizar a imagem da arena.

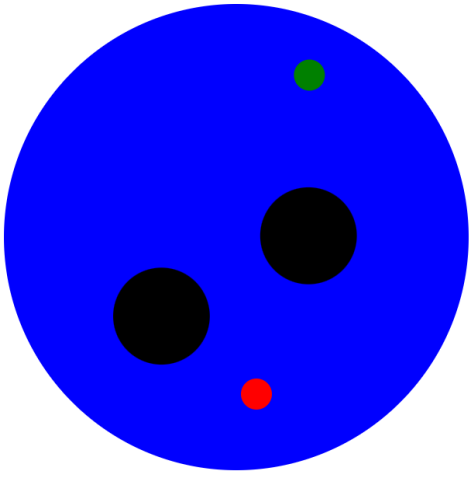

Figura 1: Arquivo SVG representando a arena em azul, obstáculos em preto, jogador 1 em verde e jogador 2 em vermelho.

*Arena*

Assim como no trabalho curto 4, o programa deverá criar uma arena virtual, porém desta vez em 3D. O plano x e y terá informações idênticas às lidas do arquivo "svg" (assim como no trabalho anterior). A altura da arena, z, deverá ser 4 vezes a altura do jogador (a ser definido adiante) e a altura dos obstáculos será a mesma dos jogadores.

*Jogador*

Cada jogador deverá ter os mesmos componentes do trabalho anterior (pernas, braço, arma, etc.), porém agora em 3D. A perna deverá se mover à medida que o jogador anda, simulando assim, o movimento de perna de um humano normal. A perna deve implementar pelo menos a junta do quadril e do joelho. Utilize a criatividade para construir o jogador! Ele continuará sendo delimitado pelo círculo definido no "svg" (no plano x-y), ou seja, o jogador deverá caber no círculo definido no SVG e terá sua altura proporcional ao tamanho do círculo (a altura vai depender do modelo escolhido como jogador). Perceba que o círculo é "virtual", ele serve apenas para calcular a colisão e não deve ser mostrado na tela.

*Atirar*

O jogador poderá atirar assim como no trabalho anterior, porém agora no espaço 3D, ou seja, o braço pode fazer o movimento de um cone imaginário e a bala deve seguir na direção em que a arma estiver apontando. Os controles da arma (braço) seguem o do trabalho anterior, ou seja, movimento do mouse de um lado para o outro e para cima e para baixo devem mexer o braço do jogador 1 respectivamente para essas direções, enquanto para o braço do jogador 2, além dos movimentos definidos no trabalho anterior, as teclas "8" e "2" permitirão movimentá-lo para cima e para baixo respectivamente. O braço poderá ir de -45° a +45°, de um lado para o outro, e de -45° a 45°, de baixo para cima. Os tiros seguem os mesmos comandos do trabalho anterior.

*Pulo*

O jogador deverá ser capaz de pular ao pressionar as teclas "x" e "." respectivamente para o jogador 1 e jogador 2. Ele não deverá pular se já estiver executando um pulo e poderá apenas mover-se para frente e para trás quando estiver pulando. O pulo deverá ser de mais ou menos 2 vezes a altura do personagem e se o botão de pular for liberado antes de atingir a altura máxima, o jogador deverá começar a descer. Isso permitirá controlar a altura do pulo com o botão. O pulo deverá permitir subir nos obstáculos. Uma vez em cima do obstáculo, o jogador deverá se movimentar normalmente, pular ou cair ao sair dele. O pulo deve durar aproximadamente 4 segundos, quando feito na altura máxima, 2 para subir e 2 para descer. Os valores de tempo dos pulos foram estimados aqui, se precisarmos ajustar, podemos conversar sobre os melhores valores.

*Colisão*

Os personagens não deverão: sair da arena, invadir obstáculos ou ocupar um mesmo espaço do outro jogador. A colisão deverá ser calculada considerando sempre o círculo da cabeça e a altura do personagem ou obstáculo. A colisão deverá considerar o círculo no plano x-y e a altura do jogador em z, ou seja, deverá considerar o cilindro envolvendo o jogador. Caso o jogador pule e caia na cabeça de um oponente, esse deverá ser tratado como um obstáculo (ou seja, o jogador poderá andar na cabeça do oponente).

*Jogo em geral*

O jogo em geral deverá seguir as funcionalidades do T2D, e.g., ganhar, perder, mostrar mensagens, reiniciar, etc.

*Aparência do Jogo*

Deverão ser utilizados conceitos de iluminação e textura. O jogo deverá conter pelo menos um modelo de luz na arena (pontual ou direcional). Além disso, o jogo deverá ter um modo noturno (fazer a troca de modos com a tecla "n") e que todas as luzes da arena são apagadas e a lanterna (representada por uma iluminação spot) fixada na arma de cada jogador e apontando na mesma direção dela é acesa. As paredes, o chão e o teto da arena deverão ser texturizados, assim como o corpo do jogador. As paredes e o chão deverão enfatizar o efeito da lanterna. O aluno está livre para escolher as texturas e utilizar luzes adicionais. Use a criatividade!

*Câmeras*

As visões das câmeras deverão ser implementadas para cada jogador, ou seja, é necessário dividir o viewport! O jogo deverá implementar 2 tipos de visões que poderão ser trocadas (afetando a visão dos dois jogadores) com os botões numéricos do teclado ("v" e "b"). A tecla "v" (opção padrão) deverá acionar uma câmera perspectiva posicionada em cima da arma e olhando na direção dela (up apontando para o teto). Com essa visão, deve ser possível ver parte do "cano" da arma (ou o que for usado para

atirar). A tecla "b" deverá acionar uma câmera perspectiva posicionada inicialmente atrás do jogador (a uma distância grande suficiente para ver todo o jogador por uma terceira pessoa) e a uma altura superior à do jogador, e olhando para o centro do jogador (up apontando para o teto). Essa última câmera poderá ser rotacionada em torno do jogador quando pressionado o botão direito do mouse (-180° a +180° de um lado para o outro e -90° a +90° de cima para baixo). As teclas "+" e "-" permitirão alterar o zoom dessa câmera.

*Visão do Jogador*

Implementar uma visão permanente do olho do jogador. Ela deverá usar uma câmera perspectiva posicionada no olho do jogador e olhando para frente (*up* apontando para o teto). Utilizar uma janela com 200 pixels a mais em *y* para mostrar a visão do jogador constantemente durante o jogo (isto é, o viewport inicial de 400x500 para cada jogador será de 400 por 700 se essa funcionalidade for implementada). É necessário dividir o viewport novamente!

*Bônus 1*

Mapa de posição, sua e dos seus adversários. Utilizar uma câmera ortogonal para desenhar um minimapa da arena descrevendo a sua posição (verde), a posição do adversário (vermelho), e dos obstáculos (preto). O chão desse mapa deve ser transparente para não ofuscar a visão original do jogo (Ou seja, utilizar apenas linha para representar os círculos da pista). Utilizar o mesmo conceito da impressão de texto no canto da tela. O mapa deve ficar fixo no canto inferior direito e ocupar 1/4 da largura da janela (ou seja, 1/16 da área).

*Bônus 2*

Utilizar modelos avançados de jogador e suas partes (ver exemplos na Figura 2). O aluno está livre para utilizar modelos 3D e suas partes feitos no Blender ou baixados da internet, ou editado com ambos. Um exemplo de site com modelos é o do mixamo (https://www.mixamo.com/). A qualidade dos modelos será julgada caso a caso. Atenção, modelos muito pesados podem deixar o jogo muito lento e isso não é desejável. Escolha de forma a ter um jogo fluido. Para quem for usar modelos avançados, será permitido usar a arma acoplada ao ombro para não ter que movimentar o braço do modelo. Neste caso, as funcionalidades referentes ao braço descritas nos itens anteriores deverão ser implementadas para a arma (ou seja, controle com o mouse, atirar, etc).

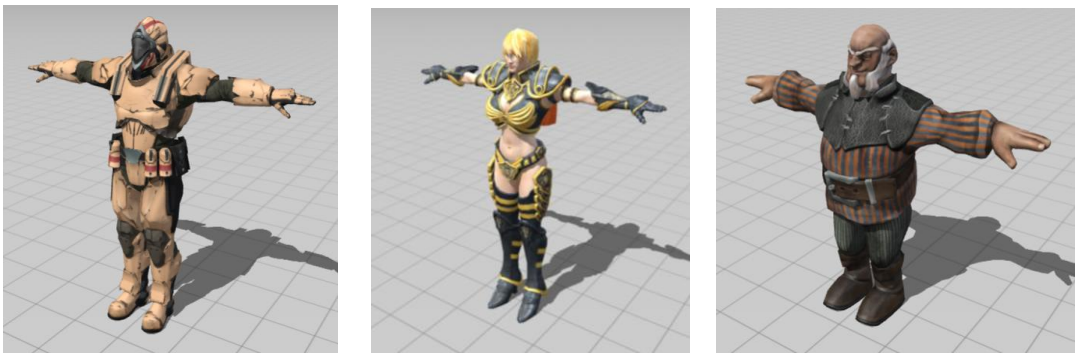

Figura 2: Exemplos de modelos avançados.

OBSERVAÇÔES: O aluno poderá incluir (e deverá, se for a única maneira de mostrar uma funcionalidade) parâmetros e teclas adicionais para facilitar a apresentação do trabalho. Por exemplo, o aluno pode criar uma tecla para habilitar e desabilitar uma determinada funcionalidade, para mostrar que ela funciona. As funcionalidades só serão pontuadas se elas forem vistas durante a apresentação, isto é, falar que colocou a luz não basta, é necessário mostrar o seu efeito e explicar coerentemente. Esperar 10 minutos para ver o um personagem ser atingindo não é factível. Planeje a sua apresentação. O aluno deverá utilizar os mesmos conceitos já exigidos no trabalho anterior. Um arquivo exemplo será distribuído juntamente com essa especificação. Inclua um README.txt explicando os atalhos e funcionalidades adicionais.

# 3 Regras Gerais

O trabalho poderá ser feito em dupla, exceto para os alunos das pós-graduação. Trabalhos identificados como fraudulentos serão punidos com nota zero e os envolvidos poderão ter que responder a instâncias superiores da universidade. Casos típicos de fraude incluem, mas não se restringem à cópia de trabalhos, ou parte deles, assim como trabalhos feitos, no todo ou em partes, por terceiros (sendo estes pessoas ou sistemas de inteligência artificial).

## 3.1 Entrega do Trabalho

O código deverá ser entregue pelo Google Classroom dentro do prazo definido. Trabalhos entregues após a data estabelecida não serão avaliados.

A entrega do trabalho deverá seguir estritamente as regras a seguir. O não cumprimento **inviabilizará a correção do trabalho**.

- Arquivo zippado (com o nome dos autores, ex. FulanoDaSilvaGeronimoFeliz.zip) contendo todos os arquivos necessários para a compilação do trabalho;
- **Não enviar** arquivos já compilados, inclusive bibliotecas!
- O arquivo zip deverá necessariamente conter um *makefile* que implemente as seguintes diretivas "make clean" para limpar arquivos já compilados, "make all" para compilar e gerar o executável. O executável deverá ser chamado *trabalhocg*.

Lembre-se que a localização do arquivo da arena será passada via linha de comando e, portanto, não se deve assumir que haverá um arquivo desses na pasta do executável. Seja cuidadoso ao testar o seu programa, isto é, não teste com o arquivo no diretório do programa, pois você pode esquecer de testá-lo em outro lugar posteriormente.

# 4 Pontuação

O trabalho será pontuado conforme a tabela dada na última folha desse documento. Bugs serão descontados caso a caso. Observe que existem duas funções bônus no trabalho, ou seja, 2 pontos extras. Os pontos dessas questões bônus serão utilizados para completar a nota desse trabalho, da prova, ou dos trabalhos curtos que não tenham atingido a nota máxima 10.

## 4.1 Apresentação do Trabalho

O trabalho terá 15 minutos para ser apresentado para a turma. A apresentação será feita para a turma no laboratório. As apresentações ocorrerão no horário da aula e em uma data posterior à de entrega, conforme cronograma das aulas. Durante o tempo de apresentação, o aluno deverá mostrar e testar todas as funcionalidades requeridas do trabalho. O trabalho (arquivos) a ser utilizado na apresentação deverá ser o mesmo enviado para o professor, e será fornecido pelo professor na hora da apresentação. A ordem de apresentação será sorteada durante a aula, portanto, todos os alunos devem estar preparados para apresentar o trabalho durante o período de apresentações. Os alunos devem estar preparados para responder possíveis perguntas sobre o trabalho. Prepare-se para fazer a apresentação dentro do seu tempo. **Pontos só serão dados para funcionalidades apresentadas**, isto é, a audiência deverá ser capaz de ver e perceber o resultado produzido pela funcionalidade implementada no jogo. Cabe aos alunos, portanto, criar atalhos (para habilitar e desabilitar funcionalidades, por exemplo, movimento do oponente) no trabalho para facilitar a apresentação das funcionalidades.

# 5 Erratas

Qualquer alteração nas regras do trabalho será comunicada em sala e no portal do aluno. É de responsabilidade do aluno frequentar as aulas e manter-se atualizado.

| | |
|---|---|
|  | Centro Tecnológico<br>Departamento de Informática |
| Disciplina: Computação Gráfica | Código: INF09282 e INF09284 |
| Prof. Thiago Oliveira dos Santos | |

**Tabela de pontos**

| | |
|---|---|
| Nome do aluno: | |
| Nome do aluno: | |

| Itens | Sub-Itens | Feito | Observações | Pontos | Nota |
|---|---|---|---|---|---|
| Base do jogo | Jogo (morrer, ganhar, mensagem, etc.) | | | 1,0 | |
| Jogador 3D | Movimento das pernas | | | 1,0 | |
| | Braço e tiro | | | 1,0 | |
| | Pulo e obstáculos | | | 1,0 | |
| Aparência do jogo | Iluminação 1 | | | 1,0 | |
| | Lanterna | | | 1,0 | |
| | Textura | | | 1,0 | |
| Câmeras | Câmera 1 | | | 1,0 | |
| | Câmera 2 | | | 1,0 | |
| | Visão do Jogador | | | 1,0 | |
| Bônus 1 | Minimapa | | | 1,0 | |
| Bônus 2 | Modelos avançados | | | 1,0 | |