ARUÃ DE MELLO SOUSA
DANIEL GRACIA DOS SANTOS
KIM ARCHANJO TOSTES
RAFAEL BARBOZA JANUZI

Desenvolvimento de um Jogo em 3D Baseado em um Jogo da Plataforma Odyssey II

Batalha Medieval - Relatório Final

Professora:

Dra. Ana Luísa Dine Martins Lemos

São José dos Campos, SP

Introdução

Batalha Medieval é um jogo de artilharia. Cada lado do jogo há um canhão, um operador e uma muralha. O objetivo é utilizar o canhão para destruir o inimigo antes que ele te destrua. Você também pode atrasar o outro canhão e seu operador temporariamente se acertá-lo.

Segue abaixo a descrição do jogo original:

Este jogo ocorre em um tempo em que cavaleiros eram corajosos e desintegrar castelos eram um dos esportes mais populares.

Neste artilharia, dois jogadores devem tentar destruir o castelo do outro utilizando suas catapultas. Pressionando qualquer direção no joystick, o jogador lança uma pedra no oponente. Quanto mais você pressionar o botão, mais longe a pedra vai.

Juntamente acertando os castelos, pedras podem também acertar a catapulta inimiga ou o soldado que está operando. Em qualquer um desses casos, a catapulta e o soldado irão para fora da tela para receber os reparos necessários ou primeiros socorros.

Tiros mal calculados podem também acertar castelo do próprio jogador, dando pontos ao oponente. O vencedor é quem tiver mais pontos no total de dez batalhas.

Jogadores ganham três pontos para cada pedra acertada em um castelo, sete pontos se acertar o soldado inimigo e dez pontos se acertar a catapulta do inimigo. No final de cada batalha, os pontos ganhos pelo vencedor são multiplicados pelo número da batalha (de um a dez). O perdedor não ganha pontos.

O jogo possui três variações de acordo com a tensão da catapulta. Quanto mais alto o número da variação, mas rápido os tiros das catapultas.

Algumas imagens do jogo são mostradas abaixo:

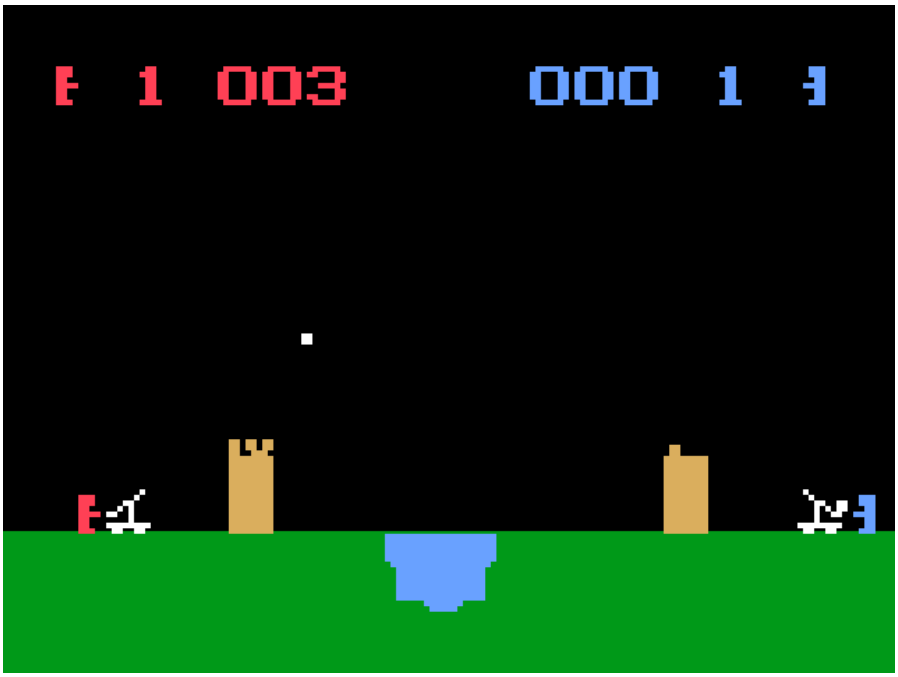

Figura 1 - Tela do jogo durante uma partida

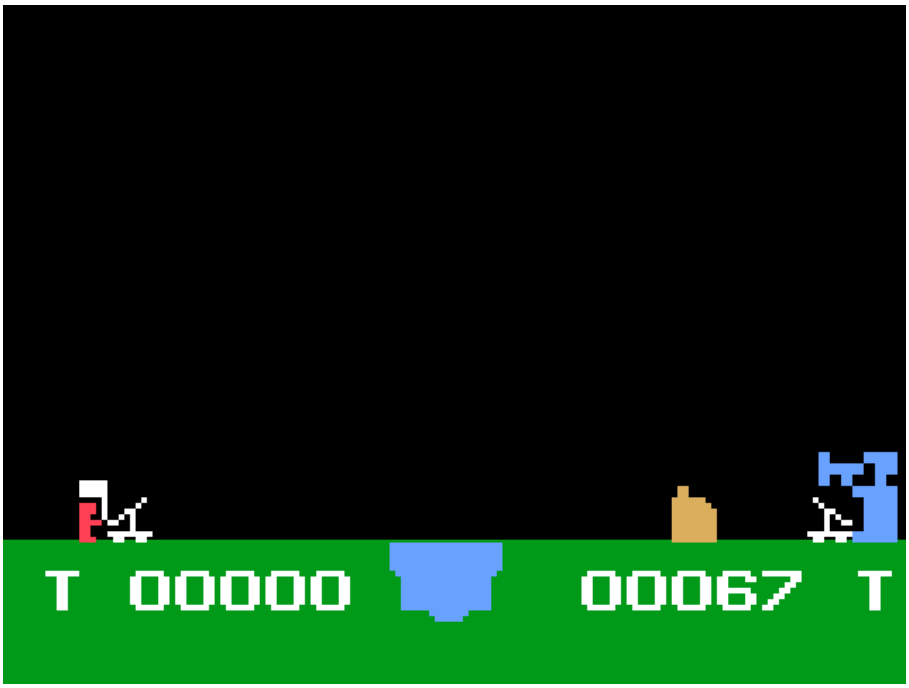

Figura 2 - Tela de final de jogo. (Vitória do jogador azul)

Este trabalho irá utilizar conceitos de Computação Gráfica, uma disciplina do curso Ciência da Computação, e possui como objetivo a recriar o jogo Batalha Medieval original do console

Magnavox Odyssey² em um ambiente inteiramente tridimensional, utilizando como base a interface de programação OpenGL para manipulação de gráficos em 2D e 3D.

Descrição do Jogo

O jogo consiste no conflito entre duas catapultas posicionadas nos cantos esquerdo e direito da tela. Cada catapulta fica detrás de um muro de proteção e é comandada por um atirador. Cada jogador controla a sua própria catapulta, e tem como objetivo destruir uma seção do muro que protege a catapulta do adversário, abrindo uma brecha em suas defesas.

Seções do muro podem ser derrubadas atingindo-as diversas vezes com os projéteis lançados pelas catapultas. A cada projétil que acerta uma seção, faz com que ele diminua de tamanho tornando-a um alvo menor e consequentemente mais difícil de acertar.

Além do muro do adversário, é possível acertar também o atirador do adversário. Quando o atirador é atingido o adversário perde tempo para que outro atirador chegue até a catapulta. Um jogador pode acidentalmente acertar o próprio muro se lançar projéteis com pouca força.

Cada alvo acertado rende ao atacante uma determinada quantia de pontos: 5 e 10 pontos para muro e atirador, respectivamente. Comparado com o jogo original, foi removida a possibilidade de acertar também a catapulta por se mostrar redundante com a presença do ataque ao atirador.

Metodologia de Implementação

Primeiramente, é importante salientar que, para centralizar o desenvolvimento, foi utilizado um repositório de código no serviço BitBucket utilizando a ferramenta Git. Foram sentidas algumas dificuldades de aprendizado, mas eventualmente a ferramenta colaborou para o bom desenvolvimento do trabalho.

O programa foi modularizado o máximo que fosse possível e viável: funções para desenho de objetos e personagens, cenários, letreiros, lógica de jogo, carregamento de textura,

iluminação e loop principal do jogo são controlados cada um em um arquivo diferente, e compilados juntos utilizando uma ferramenta de build automatizada. No arquivo principal com a função *main*, o contexto OpenGL é configurado utilizando a biblioteca GLUT, e de lá é executado o loop principal da aplicação.

Nenhuma biblioteca externa foi utilizada, porém, o programa requer a compilação utilizando o padrão GNU99, disponível apenas em compiladores GCC, para a execução de algumas macros facilitadoras.

Descrição da implementação

Figuras

Todas as figuras foram criadas utilizando primitivas do OpenGL e do GLUT, combinadas com operações de transformação e escala. Elas são criadas dentro de uma matriz própria na pilha, e depois se necessário quando invocadas, são escalonadas e transferidas para posições corretas.

As figuras são a catapulta (adaptada para canhão), os bonecos dos jogadores, a bandeira, a trombeta, e a muralha.

A muralha é gerada consultando uma matriz definindo os blocos a serem renderizados, para facilitar o desenho de muralhas destruídas. Apenas uma seção da muralha é desenhada por chamada de função. O desenho da muralha completa é feito na função de desenho do arquivo principal.

Os bonecos, originalmente feitos utilizando somente primitivas OpenGL e cores básicas, foram refeitos utilizando texturas e possuem aparência similar ao personagem principal utilizado no jogo *Minecraft*. Eles possuem 4 estados: jogando, vitoriosos, derrotados ou mortos/incapacitados. Sua pose e seu estado de animação são definidas por argumentos passados como parâmetro para a função de desenho.

No arquivo de figuras também são definidas funções para renderização de letreiros, utilizando a função glutStrokeCharacter. Elas são utilizadas para desenhar o placar de cada jogador e as telas iniciais e finais do jogo.

Cenário

O cenário é composto pelo chão gramado, pelo rio, e pelas árvores. O chão e o rio são texturizados e definidos como primitivas simples escalonadas. As árvores são geradas como

primitivas simples e renderizadas em posições aleatórias no campo de jogo. A função de randomização das posições é executada no começo do jogo, e é configurada para não colocar árvores no rio, nem do lado de dentro das muralhas.

Existe apenas uma função de desenho de cenário que executa a renderização de todas as partes. Ela é chamada logo no começo da função de desenho principal.

Iluminação

A iluminação é gerada e configurada no arquivo de configuração. É utilizado o modelo de Gouraud para a renderização de superfícies, e normais são aplicadas nos desenhos quando necessário. É utilizada iluminação ambiente, especular e difusa. A função de configuração é executada no começo do jogo.

Texturas

As texturas são carregadas utilizando funções passadas em aula para carregamento de arquivos BMP. Elas são configuradas para repetir se necessário, e são geradas como mipmaps, para melhor qualidade visual. Por motivos de performance, a texturização é ativada apenas quando necessário via glEnable/glDisable. São definidas múltiplas texturas para a muralha, e texturas para os bonecos, grama e rio. Uma constante de compilação é definida para tratar o caso de compilação em Windows ou Linux, mudando os caminhos indicando a localização das texturas.

Lógica

A lógica de jogo, isto é, as regras, o sistema de colisão, e etc, são implementadas por jogador. Funções para iniciar e terminar a medição de força do tiro são exportadas no cabeçalho, assim como funções que informam todos os dados de um jogador (estado das muralhas, estado do jogador, quantidade de força do tiro, pontuação, posição atual do projétil, etc). Para evitar duplicação de código, são utilizadas várias macros de compilação com versões “genéricas” dos códigos de tratamento para um jogador qualquer.

A colisão é feita considerando algumas equações matemáticas. Para cada jogador, é verificado a posição da bala de canhão, e calculado se ela está dentro de uma região correspondente a muralha do oponente, a muralha do próprio jogador ou diretamente na área onde o boneco do jogador oponente está. Tais regiões são derivadas de equações de parábola

para as dimensões X e Z. As regiões de colisão da muralha foram divididas previamente e são codificadas diretamente na fonte. Elas foram calculadas resolvendo-se equações envolvendo o comprimento da parábola (i.e muralha).

Função principal

No arquivo principal todas as figuras são dispostas na tela de acordo com as informações de cada jogador, o input é tratado, as funções de configuração do programa são executadas, e etc.

Um desafio encontrado foi permitir o processamento de pressionamento de múltiplos botões via GLUT. Por padrão, a biblioteca não é preparada para esse comportamento. Foi necessário criar um buffer das teclas pressionadas e desativar a repetição de teclas que é ativada por padrão (isto é, pressionar e segurar uma tecla é entendido pelo GLUT como múltiplos pressionamentos). Quando uma é pressionada, o valor correspondente no buffer é marcado como verdadeiro, e quando solta, é marcado como falso. Os estados do buffer são tratados em uma função a parte, chamada em toda execução da função principal de desenho.

O estado do jogo é definido verificando se um dos jogadores venceu o jogo, de acordo com o que é informado pelas funções de lógica. Uma tela inicial com um letreiro foi feita pedindo o pressionamento de Enter para iniciar o jogo de fato.

Para animação dos personagens, é chamada uma função timer que incrementa periódicamente um contador que é fornecido para as funções de desenho correspondentes. O desenho das muralhas e dos projéteis é feito utilizando como parâmetros as informações fornecidas pelas funções de lógica.

A câmera é configurada também no arquivo principal. É utilizada uma projeção perspectiva. Para melhor visualizar o campo de jogo, são definidos comandos de rotação da tela.

Imagens

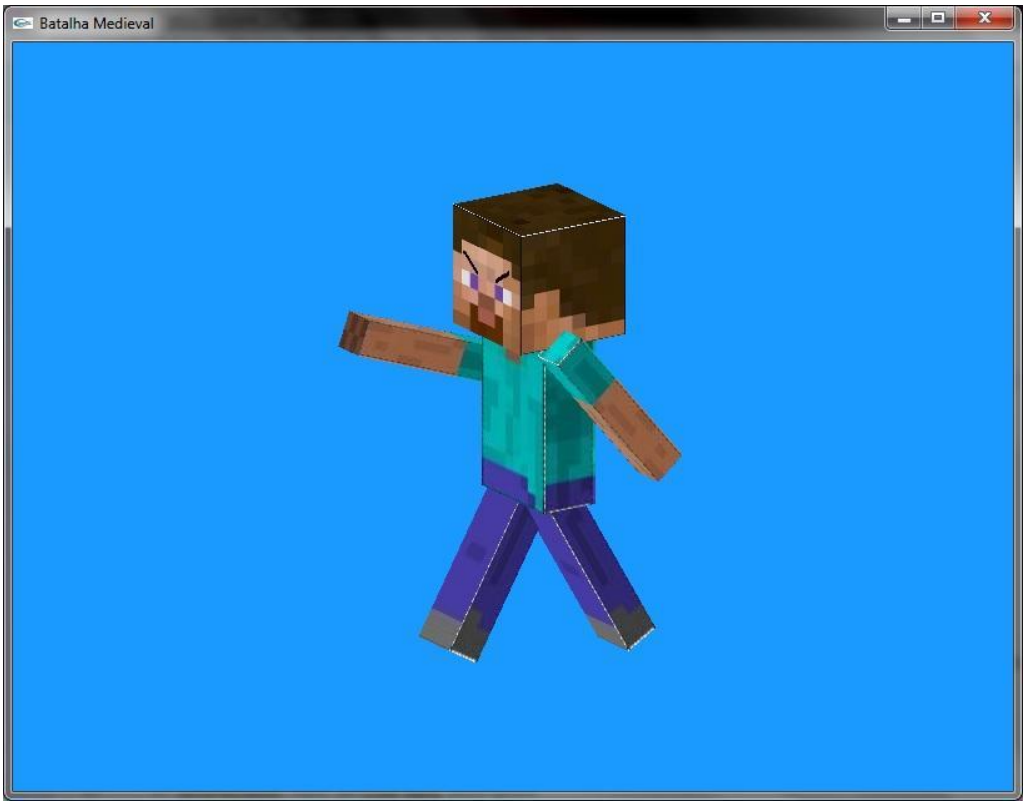

Figura 3 - Soldado texturizado

Documentação do Código-Fonte

Para a documentar o código-fonte desenvolvido utilizamos a ferramenta *Doxygen* (doxygen.org) para geracao automática de documentação que é apresentada em formato *HTML*, que pode ser acessada pelo arquivo Docs\html\index.html em anexo à esse relatório. Em conjunto com o *Doxygen* utilizamos a ferramenta *Graphviz* (graphviz.org) para a geração automática de grafos de chamadas entre as funções desenvolvidas, esses grafos são apresentados na documentação em *HTML* na descrição de cada função.

Apêndice - Demonstração de partida

O jogo começa temos uma pequena animação descrevendo o cenário como tela inicial, e clicando em ENTER iniciamos a partida.

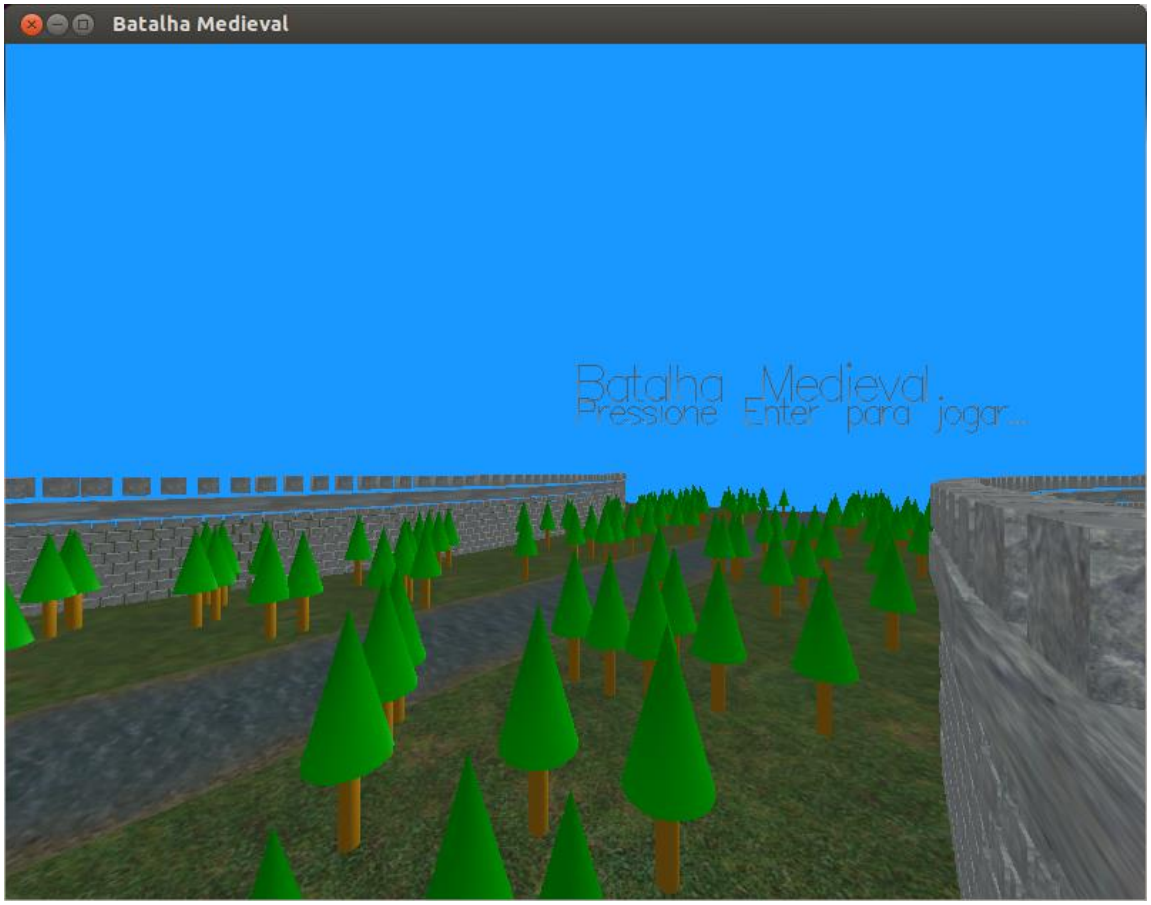

Figura 4 - Tela inicial

O jogo inicia-se então. Pontuações zeradas, cada muralha intacta. Com os comandos S, X e Z o jogador 2 move seu canhão para a esquerda, direita e atira, respectivamente.
Com os comandos J, N e M o jogador 1 move seu canhão para a esquerda, direita e atira, respectivamente. Assim a batalha começa.

Figura 5 - Cenário com texturas e placar

Figura 6 - Muralhas destruídas

Depois de algumas rodadas o jogador 1 acerta o soldado do jogador 2, o levando a uma morte precoce. Mas a batalha não pode parar, então outro soldado é convocado e a batalha continua. Em qualquer um desses casos, os soldados param temporariamente para receber os primeiros socorros.

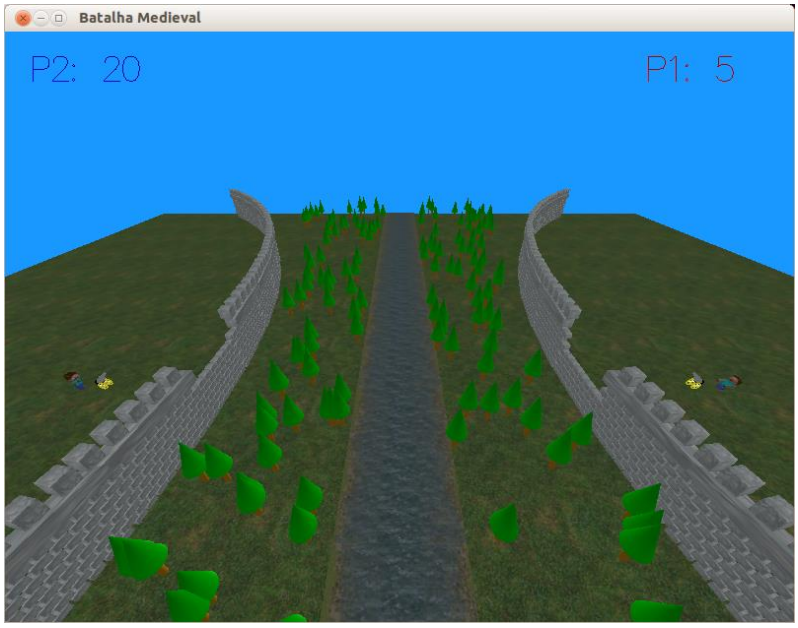

Figura 7 - Projétil acertando o jogador 2

Por fim o jogador 2 vence! Vitória

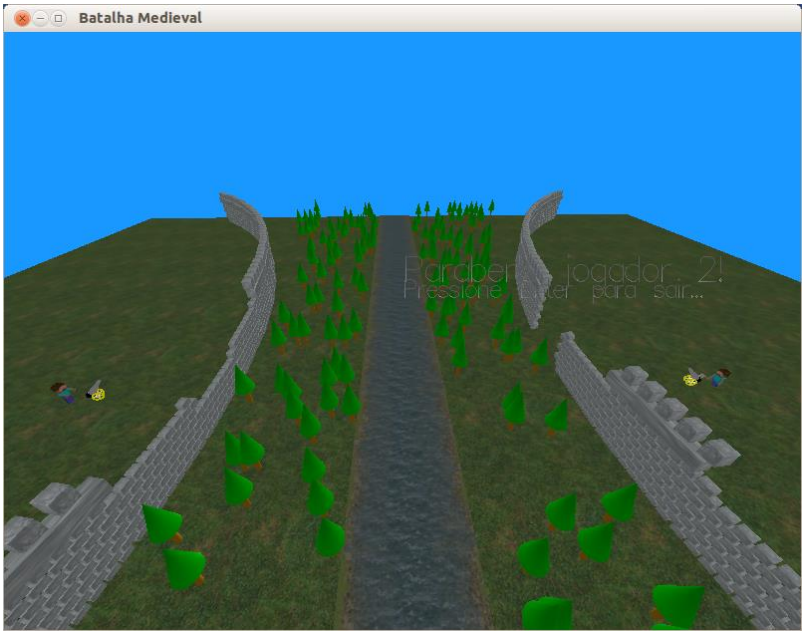

Figura 8 - Vitória do jogador 2

Conclusão

Por fim, podemos dizer que o novo jogo implementado é um remake que desafia a capacidade dos alunos em construir jogos 3D. A indústria dos vídeo-games e jogos eletrônicos é referente ao setor criativo de planejamento e produção de consoles, suportes e jogos eletrônicos. No Brasil, além da vinda de empresas multinacionais, há o fortalecimento de empresas nacionais que vendem jogos feitos no Brasil em nosso mercado interno e em outros países, também cresceu o número de cursos de desenvolvimento de games em todo o país.