**ICT – Unifesp**

**Computação Gráfica**

**Projeto Final, Parte 3 e 4**

**Data**: 15/06/2018

**Equipe**:

Guilherme Felipe Reis Duarte RA: 120805

Glauber Prado RA: 55938

**Professora**: Ana Luisa Lemos

**1 Introdução**

Este relatório apresenta a situação atual da implementação do *remake* do jogo Tank Wars de Kenneth Morse (1992) e as dificuldades encontradas.

Este relatório incluirá informações que eram para serem entregues no dia 08/06/2018.

* 1. **Glossário**

Para evitar confusão de termos, definimos um glossário a seguir.

**Jogo**: Utilizaremos este termo para nos referir ao jogo como um todo.

**Partida:** Utilizaremos este termo como uma tradução para *GAME* do Tank Wars. Ou seja, uma partida é equivalente a uma batalha, que inicia com todos os tanques no cenário e termina quando apenas 1 tanque (ou nenhum) sobrevive. O tanque sobrevivente é o vencedor da partida e recebe +1 vitória e um bônus em pontos por ter vencido (a definir). O vencedor do jogo será o jogador que tiver o maior número de vitórias.

**Rodada**: A partir deste relatório, uma rodada se referirá à vez de um jogador atirar. Após atirar, a rodada muda para o próximo jogador.

**2 Especificações**

Relembrando as especificações:

* **Jogadores**
  + 2 a 10 jogadores.
  + Não haverá configuração de times (ao contrário do jogo original): é cada um por si!
  + Não haverá inteligência artificial, i.e., não será possível jogar contra o computador.
* **Cenário**:
  + Tridimensional; porém simplificado, de forma os tanques estarão todos dispostos em uma linha (coordenada lateral y = 0 para todos).
* **Controles**: Potência do tiro (velocidade inicial) e ângulo do canhão em relação à vertical.
* **Munições**: Serão implementadas apenas um subconjunto das armas possíveis que existem no jogo original. Se o tempo permitir, tentaremos inserir outros tipos de munição.
  + Incinerador: é o tiro mais comum;
  + Incinerador Mark II: possui o dobro do raio de explosão do incinerador comum;
  + Bomba de 20 kilotons: possui o dobro do raio do Incinerador Mark II;
  + Bomba de 5 Megatons: possui o dobro do raio da Bomba de 20 kilotons.
* **“Mortes”**:
  + Explosão, com raio aleatório escolhido dentro os raios das munições citadas acima;

Nota: A explosão do tipo “terra” foi removida.

* **Paredes**: simplificadas para dois tipos:
  + Sem paredes;
  + Paredes elásticas. Rebate tiros incidentes sem alterar o módulo de sua velocidade. Pretende-se implementar paredes transparentes, que brilham no local onde o tiro a atinge.
* **Vento**:
  + Vento aleatório somente na direção x (direção de disposição dos tanques).
  + O vento é constante em uma mesma rodada; muda apenas de uma rodada para outra, como no jogo original.
* **Critério de vitória do jogo**
  + Maior número de vitórias de partidas;
  + Em caso de empate, maior número de pontos.

**3 Descrição da Implementação até o momento**

**3.1 Loop Principal do Jogo**

O foco do trabalho consistiu em implementar o *loop* do jogo, ou seja, todas as transições de uma parte do jogo até outra.

O jogo possui 5 telas distintas:

* **Tela Inicial**: Consiste em um Menu em que o usuário escolhe o número de jogadores. Já foi implementado.
* **Tela Renomear Jogadores**: Consiste em um Menu no qual o usuário renomeia cada jogador que participará do jogo. Também já foi implementado.
* **Tela Rodada**: Apresenta o cenário tridimensional do jogo em si. Os tanques são posicionados sobre o relevo e atiram uns contra os outros. Quando a rodada termina, o jogo vai para a tela de resultado parcial. Ainda está em elaboração.
* **Tela Resultado Parcial**: Apresenta a pontuação dos jogadores até o momento. Ainda está em elaboração.
* **Tela de Compras**: Apresenta os itens que cada jogador pode comprar com os dólares que possui disponível no momento. Ainda não implementado.

**3.2 Objetos 2D e 3D**

Como descrito acima, apenas a Tela da Rodada é que não apresenta um Menu para interagir com o usuário. Dessa forma, bastante trabalho foi dedicado para desenhar os menus, como mostrado nas Figuras 1 e 2.

Após a elaboração dos menus, focou-se em resolver outra dificuldade: gerar o relevo sobre o qual os tanques serão posicionados. As Figuras 3 e 4 apresentam o resultado. Falta ainda inserir iluminação no relevo.

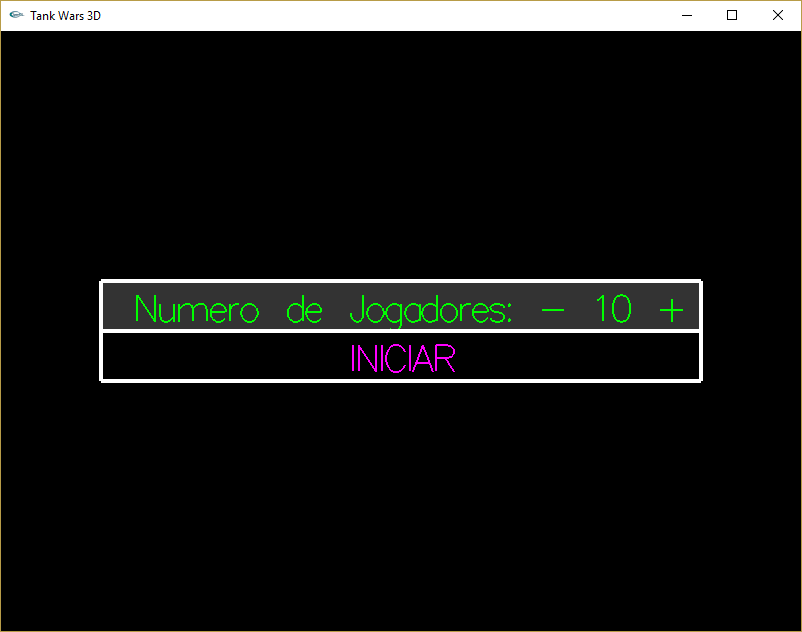


Figura 1 – Screenshots da Tela Inicial, que permite selecionar o número de jogadores.



Figura 2 – Screenshot da Tela Renomear Jogadores. É permitido aqui escrever o nome de cada jogador, ou simplesmente iniciar o jogo diretamente.

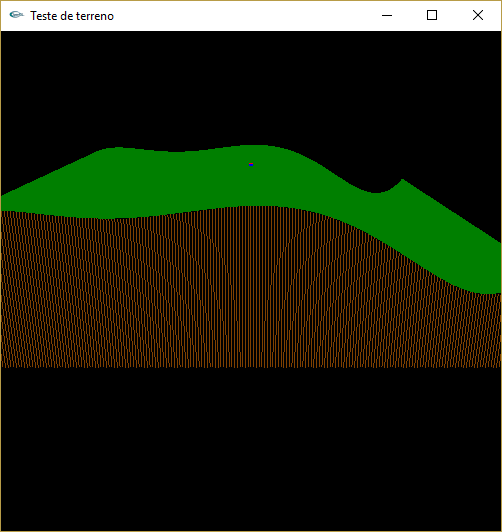


Figura 3 – Duas execuções diferentes do programa que gera o terreno, para demonstrar que cada execução gera um relevo diferente. Um pequeno cubo foi desenhado e transladado para ficar exatamente sobre um ponto (x, y, z) sobre o solo.

**4 Construção dos Objetos**

A hierarquia dos objetos construídos é apresentada abaixo:

* **Menu**: Representa um menu a ser apresentado na tela.
  + **OpcaoMenu**: Representa cada quadro do menu que pode ser selecionado, editado, clicado, etc. É uma interface que se desdobrou em vários objetos:
    - **Botao** – Um quadro que ao ser selecionado e o usuário clicar *Enter*, executa uma ação pré-determinada. É o caso de “Iniciar Jogo” ou “Voltar”.
    - **OpcaoAlterarValorNumerico**: Um quadro que permite ao usuário alterar o valor numérico de uma configuração. Por enquanto só é utilizado para mudar o número de jogadores.
    - **OpcaoEditarNome**: Um quadro que permite ao usuário editar um texto. Aqui, foi utilizado para alterar o nome de cada jogador, como apresentado na Figura 2.
* **Municao**: Uma interface comum para todos os tipos de munições do jogo. Cada tipo de munição (explosiva, de terra, etc.) deverá implementar esta interface. Porém, para este trabalho, apenas a munição explosiva será implementada.
  + **MunicaoExplosiva**: Uma classe abstrata que representa todas as munições explosivas, ou seja, que provocam danos ao explodir. Esta classe possui mais classes filhas, cada uma representando uma arma individual, com suas próprias características (tamanho da explosão e, eventualmente, um desenho diferente):
    - **Incinerador**
    - **Incinerador M2**
    - **Bomba20Kilotons**
    - **Bomba5Megatons**
* **Mundo**: Armazena todas as informações sobre o estado de execução do jogo. Exemplos: número de jogadores, número de jogos, Tela Ativa, Menu Ativo, etc. Armazena ainda os métodos que transicionam de um estado para outro do jogo, bem como o conjunto de todos os objetos Jogador.
  + **Jogador**: Armazena as principais informações sobre cada jogador: Sua cor, seu nome, armas que possui, etc.

Prevê-se ainda a implementação de um objeto chamado “Cenário”: ele armazenará as informações sobre uma rodada ativa: matriz que representa o relevo; a ordem em que aparecem os jogadores no cenário; quais jogadores estão vivos e quais não estão, etc.

**5 Funções utilizadas que não foram explanadas em aula**

**5.1 glutStrokeCharacter()**

A função *glutStrokeCharacter()* é uma função da própria biblioteca GLUT, que permite desenhar textos na tela. Os textos são desenhados utilizando linhas, o que permite que o texto possa ser transladado, rotacionado e sofrer operações de escala (KILGARD, 2018; LIGHTHOUSE3D.COM, 2018).

Contudo, a funcionalidade só apresenta dois tipos de fonte: “GLUT\_STROKE\_ROMAN” e “GLUT\_STROKE\_MONO\_ROMAN”. Não são fontes “bonitas” e elaboradas, porém, para fins deste trabalho, atenderam muito bem ao elaborar os menus.

**6 Dificuldades encontradas**

**6.1 Desenhar os Menus**

Desenhar os menus foi mais difícil do que se antecipou. Como o OpenGL não possui uma biblioteca que desenhe menus, tal implementação foi realizada do zero.

Demorou-se para decidir a melhor forma de implementar, até que se chegou na hierarquia de objetos descrita no item 4 – Construção dos Objetos.

Posteriormente, houve dificuldade em encontrar um método de escrever texto na tela. Após buscas na internet, chegou-se ao site do próprio OpenGL, na qual são descritas diversas bibliotecas existentes que desenham textos na tela (OPENGL.ORG, 2018). Uma das formas descritas foi o Stroke Character, método utilizado no jogo atual.

O método Stroke Character foi escolhido ao invés de selecionar bibliotecas externas pois:

* É implementado pelo próprio GLUT;
* Permite ser transladado e redimensionado.

O GLUT possui ainda um método de exibir caracteres chamado *glutBitmapCharacter()*, com um número maior de fontes e tamanhos de caracteres. A desvantagem desse método – e principal motivo pelo qual não foi utilizado – é que esse texto é desenhado diretamente em coordenadas da tela, não permitindo, portanto, que possa ser redimensionado ou transladado para um local em coordenadas do mundo.

Após determinada a forma de desenhas os caracteres, passou-se para o desenho dos quadros do menu em si. Foi bastante trabalhoso realizar o seu posicionamento correto na tela: os quadros, bem como seus textos, precisavam estar centralizados na tela, e cada texto precisava caber dentro de seu respectivo quadro.

Além dos desenhos de menu, foi ainda necessário implementar do zero o objeto OpcaoEditarNome, que consiste em um mini editor de textos.

**6.2 Desenho do Relevo**

A implementação do relevo também se mostrou trabalhoso. Desejava-se que o relevo fosse gerado aleatoriamente a cada rodada, ao invés de o relevo ser sempre igual entre uma rodada e outra. Em outra palavras, deseja-se gerar alturas de relevo **proceduralmente**.

**6.2.1 Limites das Coordenadas do Mundo**

Determinou-se que o relevo se extenderá pelos limites:

* 0 ≤ x ≤ 100
* 0 ≤ y ≤ 100
* z > 0.

**6.2.2 Representação do Relevo e dos Vetores Normais à Superfície**

Considerados os limites acima citados, o relevo foi representado como uma **matriz**, na qual cada elemento contém as coordenada (x, y, z) da superfície do relevo. Para elaborar a matriz, as coordenadas foram discretizadas em passos de 0,5 unidades.

Com isso, tem-se que coordenada x e a coordenada y ambos apresentam um total de (100 / 0,5) + 1 = 201 pontos em cada eixo.

Portanto, a matriz **malha**, que representa o relevo, possui dimensões 201 x 201 elementos. Cada índice (i, j) da matriz está associada às coordenadas do mundo pelas seguintes expressões:

Onde xmin e ymin correspondem ao valor da menor coordenada do mundo. Apesar de inicialmente serem considerados iguais a 0, esses valores poderão mudar futuramente, caso necessário.

Juntamente com a matriz de coordenadas, o jogo também manterá uma matriz de **vetores normais ao terreno**, com dimensões iguais à matriz malha. Cada elemento seu corresponderá ao vetor normal à superfície no ponto (i, j) da matriz do relevo.

**6.2.3 Desenho da Superfície**

Supondo que a malha do terreno esteja definida, o relevo é desenhado mediante a criação de diversos triângulos sobre as coordenadas existentes na matriz, conforme a figura a seguir. O efeito pode ser obtido utilizando o comando *glBegin(GL\_TRIANGLE\_STRIP)*.

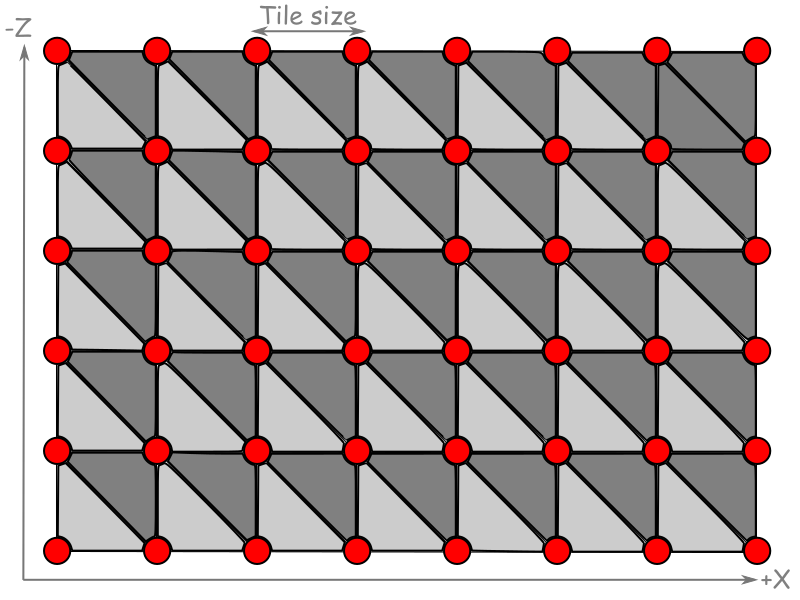


Figura 4 Ilustração do desenho da malha do terreno (TORAL, 2018).

**6.2.4 Determinar o Vetor Normal à Superfície do Terreno**

Esse passo é fundamental para que a iluminação da superfície possua o efeito desejado. Conforme explanado em sala, quando objetos são desenhados utilizando as primitivas do OpenGL tais como GL\_POLYGON ou GL\_TRIANGLES, a normal de cada vértice deve ser determinada manualmente.

Para resolver esse problema, seguiu-se a explanação do Apêndice E do livro eletrônico The Official Guide to Learning OpenGL, Version 1.1. (<http://www.glprogramming.com/red/appendixe.html>).

Uma superfície 3D suave e diferenciável pode ser escrita como uma função vetorial da forma:

Onde *X*, *Y* e *Z* são funções escalares diferenciáveis, e *s* e *t* são duas variáveis que descrevem duas direções distintas na superfície. Dessa forma, as derivadas abaixo:

Fornecem os vetores tangentes à superfície nas direções *s* e *t*, respectivamente. Portanto, dado um ponto (s0, t0), o vetor normal à superfície nesse ponto é dado pelo produto vetorial:

Em particular, a nossa matriz de relevo representa uma superfície em que *z*= *f*(*x*, *y*). Escrevendo essa superfície na forma vetorial, temos que:

Com isso, os vetores tangente nas direções x e y são dados por:

A multiplicação vetorial entre eles fornece a expressão para o vetor normal em qualquer ponto do terreno:

Portanto, após determinar a matriz de coordenadas (x, y, z) do terreno, calcula-se a matriz de vetores normais baseado na expressão acima. Para cada elemento i, j da matriz de vetores normais, as coordenadas do vetor normal no ponto correspondente são calculados utilizando uma aproximação da derivada em cada ponto:

**7 Resumo dos trabalhos realizados e próximos passos**

Tabela 1 – Tarefas realizadas e tarefas pendentes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tarefa** | **Início** | **Término** | **% completo** | **Responsável** |
| Definição das Estruturas de Dados | 25/05/2018 | 15/06/2018 | 90% | Guilherme |
| Elaboração do Loop Principal do Jogo | 25/05/2018 |  | 75% | Guilherme |
| Elaboração dos algoritmos da física do voo dos objetos | 25/05/2018 | 28/05/2018 | 95% | Guilherme |
| Elaboração dos Menus e de seus respectivos objetos 2D | 04/06/2018 | 14/06/2018 | 67% | Guilherme |
| Elaboração do relevo 3D | 14/06/2018 | 16/06/2018 | 80% | Guilherme |
| Implementação da jogabilidade: rodadas, tiros, etc. | 16/06/2018 |  | 0% | Guilherme |
| Desenhos 3D dos tanques | 16/06/2018 |  | 0% | Guilherme |
| Animação dos tiros | 17/06/2018 |  | 0% | Guilherme |
| Efeitos das Explosões | 18/06/2018 |  | 0% | Guilherme |

**Referências**

INTERNET ARCHIVE. **Tank Wars:** Kenneth Morse. Disponível em: <<https://archive.org/details/TankWars_274>>. Acesso em: 25 mai. 2018.

KILGARD, Mark. **Font Rendering**. Disponível em: <<https://www.opengl.org/resources/libraries/glut/spec3/node75.html>>. Acesso em 06 jun. 2018.

LIGHTHOUSE3D.COM. **Stroke Fonts.** Disponivel em: <<http://www.lighthouse3d.com/tutorials/glut-tutorial/stroke-fonts/>>. Acesso em: 05 jun. 2018.

MORSE, Kenneth. **Tank Wars:** Release 3.2. Manual do jogo, disponível apenas em arquivo eletrônico. 1992.

OPENGL.ORG. **Survey of OpenGL Font Technology**. Disponível em: <<https://www.opengl.org/archives/resources/features/fontsurvey/>>. Acesso em 05 jun 2018.

TORAL, Iago. **OpenGL Terrain Renderer**: rendering the terrain mesh. Disponível em: <<https://blogs.igalia.com/itoral/2016/10/13/opengl-terrain-renderer-rendering-the-terrain-mesh/>>. Acesso em: 15 jun. 2018.

THE OFFICIAL GUIDE TO LEARNING OPENGL, Version 1.1. Disponível em <<http://www.glprogramming.com/red/index.html>>. 20??. Acesso em: 15 jun. 2018.