**ICT – Unifesp**

**Computação Gráfica**

**Projeto Final, Entrega Final**

**Data**: 29/06/2018

**Equipe**:

Guilherme Felipe Reis Duarte RA: 120805

**Professora**: Ana Luisa Lemos

**1 Introdução**

Este relatório apresenta a implementação do *remake* do jogo Tank Wars de Kenneth Morse (1992). O relatório descreverá o jogo, a metodologia de implementação e fará uma breve descrição do programa final.

* 1. **Glossário**

Para evitar confusão de termos, definimos um glossário a seguir (que eu mesmo me confundi do relatório passado para esse; os termos abaixo são finais).

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

**Rodada:** Utilizaremos este termo como uma tradução para *GAME* do Tank Wars. Ou seja, uma rodada 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.

**2 Descrição do jogo original**

**2.1 Introdução do jogo**

**Tank Wars** é um antigo jogo de artilharia 2D, escrito por Kenneth Morse de 1990 (primeira versão) a 1992 (versão 3.2, a mais recente encontrada). O jogo foi feito para DOS e atualmente só é possível jogá-lo utilizando um emulador como o DOSBox (<https://www.dosbox.com/>).

O jogo consiste em um cenário com 2 a 10 jogadores, cada um controlando um tanque de guerra. O objetivo principal é destruir todos os outros tanques inimigos e ser o único sobrevivente. Os tiros não são realizados simultaneamente: como em um jogo de tabuleiro, cada tanque tem a sua própria vez de mirar e atirar. É possível inclusive que um tanque seja destruído antes mesmo de chegar a sua vez de jogar! Por esse motivo, o jogo é realizado em várias rodadas (por exemplo 20); o tanque que atira primeiro varia entre uma rodada e outra.

À medida que um jogador destrói um (ou mais) tanques, ele ganha pontos e uma quantidade equivalente em dólares, os quais podem ser utilizados para comprar munições mais poderosas entre uma rodada e outra.

O número de rodadas é estabelecido antes do início do jogo. Ao final de todas as rodadas, vence o jogador que vencer obtiver o maior número de vitórias (não necessariamente a maior pontuação) isto é, quem sobreviveu uma maior quantidade de rodadas. Em caso de empate em número de vitórias, vence o jogador com a maior pontuação.

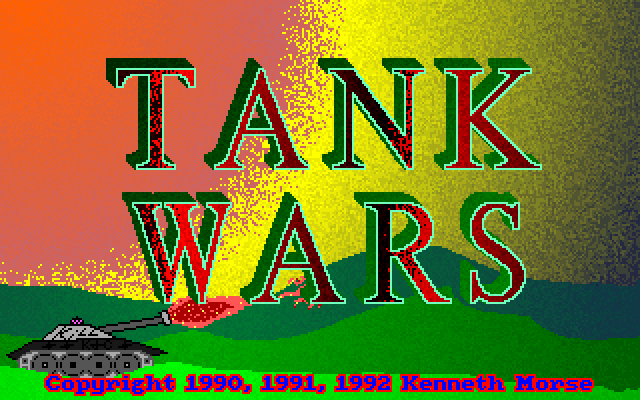


Figura 1 Tela inicial do jogo Tank Wars. Fonte: <https://archive.org/details/TankWars_274>.

**2.2 Características de Tank Wars**

**2.2.1 Básico**

No início do jogo, todos os tanques possuem “100 homens”. Quando chega a sua vez, o jogador escolhe a munição desejada, o ângulo do canhão e a potência do tiro.

Após escolher os parâmetros, o jogador pressiona espaço para atirar. O tiro segue então uma trajetória parabólica, sofrendo efeito da gravidade, bem como do vento, se houver..

A potência máxima que um tanque consegue aplicar em um tiro é 10 vezes a quantidade de homens que possui, ou seja, 1000 no início do jogo. Essa potência é reduzida à medida que o tanque sofre danos. Tanques podem sofrem danos de duas formas:

* Quando atingidos pelo raio de explosão de um tiro; e
* Quando caem por consequência de a terra sob eles ter sido desintegrada por explosões ou por munições “terremoto” como descrito abaixo.

À medida que um tanque sofre dano, ele perde homens. Quando a quantidade de homens cai para zero, o tanque é destruído e removido do cenário.

**2.2.2 Cenário**

No jogo original, os tanques são inseridos em um ambiente retangular, com colinas geradas aleatoriamente. Quanto mais jogadores houver, mais próximos os tanques estarão uns dos outros, aumentando o caos da rodada.

O cenário é delimitado por “paredes”. As paredes podem ter diferentes comportamentos, conforme as opções escolhidas no início do jogo:

* “Sem paredes”: os tiros atravessam a parede e deixam o cenário. Dependendo de sua trajetória ou do vento, podem eventualmente retornar ao cenário. A fronteira inferior comporta-se com se possuísse terra: tiros explodem quando atingem ali.
* “Warp”: quando atravessam as paredes laterais, o tiro é transportado da fronteira esquerda para a direita ou vice versa. As paredes superior e inferior comportam-se como se não houvesse paredes.
* “Elástica”, “Aceleradora” ou “Grudenta”: tiros que atingem a parede são rebatidos. A parede elástica não altera a velocidade do tiro; paredes aceleradoras aceleram o tiro em 50% e grudentas reduzem a velocidade em 50%.

**2.2.3 Munições e Defesas**

O jogo original conta com diferentes tipos de munição. Classificamos os tipos de itens que um jogador pode comprar da forma abaixo:

* Munições convencionais: explodem quando atingem um obstáculo. Variam em força, desde uma pequena granada (só destroem um tanque inimigo se atingir em cheio) até uma bomba de “5 Megatons” (possui um raio de explosão que atinge uma porção considerável do cenário).
* Munições “terremoto”: Removem uma grande porção de terra abaixo dos tanques; ao final do seu efeito, os resquícios de terra e os tanques que estavam sobre elas caem, sofrendo dano (exceto se possuir *amortecedor de inércia*, descrito abaixo). Costuma ser uma forma eficiente de destruir vários tanques de uma vez!
* Munições de terra: Ao invés de explodir, estas munições produzem terra. Normalmente são usados como defesa.
* Munições especiais: Elaboramos este grupo para descrever o canhão laser e o MIRV (*Multiple Independently Targetable Reentry Vehicle*).
  + Laser: um tiro que segue uma linha reta, não é afetado por vento e atravessa a terra.
  + MIRV: Consiste em um tiro que se subdivide em 5, em uma configuração pré-determinada.
* Acessórios de defesa: São itens que protegem o tanque contra danos e , consequentemente, aumentam as chances de o tanque sobreviver. No jogo original, há 3 defesas possíveis:
  + Amortecedor de inércia: Protege o tanque contra quedas. O amortecedor é gasto à medida que o tanque cai; dessa forma, ele só consegue proteger o tanque até uma certa altura máxima.
  + Repulsor: Repulsa quais quer tiros que cheguem a um certo raio de distância do tanque. A força do repulsor, contudo, é limitada: tiros mais rápidos conseguem penetrar o repulsor. O laser também não é afetado pelo repulsor. O repulsor é perdido quando o tanque sofre dano causado por explosões (mesmo não sendo destruído).
  + Escudo: Protege o tanque de explosões. Em 100%, consegue proteger o tanque contra danos de até 200 homens (um chute; ainda não temos certeza).

**2.2.4 Fotos da tela do jogo**

São apresentados a seguir algumas fotos da tela do jogo. As fotos são todas da janela do emulador DOSBox.



Figura 2 – Início de uma partida de Tank Wars, com 10 jogadores.

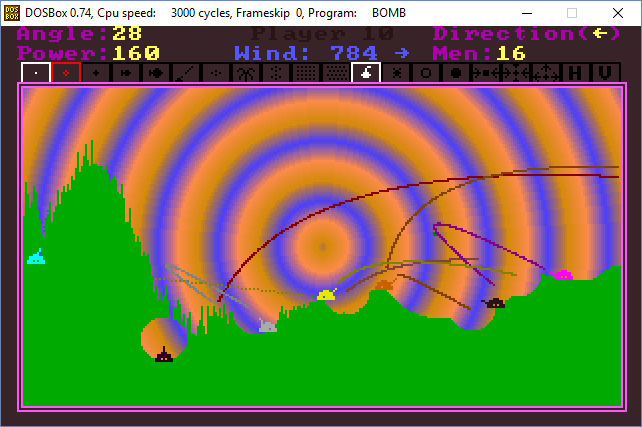


Figura 3 – Partida após alguns tiros. Tanques que são danificados têm a intensidade de sua cor diminuída até ficarem negros.



Figura 4 – Tela de resultado parcial.

**3 Descrição do Tank Wars 3D**

**3.1 Características**

Para o projeto final, pretendeu-se implementar uma versão 3D do Tank Wars original de Kenneth Morse. A versão 3D possui as características a seguir:

* Modelos 3D dos tanques, do terreno, dos tiros e das explosões;
* Utiliza o recurso de iluminação para evidenciar a tridimensionalidade dos modelos;
* Foram utilizadas duas fontes de iluminação diferentes: uma para representar o sol e iluminar o cenário inteiro; outra que é criada durante a ocorrência de explosões, de forma a representar o efeito de iluminação causado por elas;
* O cenário 3D é posicionada em uma viewport que ocupa apenas parte da tela, de forma a deixar uma porção superior da janela disponível para exibir informações sobre o jogador atual;
* Da mesma forma que no jogo original, a cada rodada, um terreno com elevações aleatórias é criado; os jogadores são posicionados em ordem aleatória ao longo do terreno;
* Ao abrir o jogo, ele apresenta ao usuário um menu que permite que se escolha o número de jogadores; na tela seguinte, outro menu permite nomear cada jogador.

**3.2 Especificações do jogo**

Será abaixo apresentada as especificações do jogo que haviam sido introduzidos na Entrega 2 do Projeto Final. Junto a cada especificação, será apresentada uma nota se a especificação foi implementada ou não.

* **Jogadores – implementado**.
  + 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 –** implementado parcialmente:
  + 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. – **implementado**.
* **Munições**: Serão implementadas apenas um subconjunto das armas possíveis que existem no jogo original. Se o tempo permitir, tentaremos inserir outros tipos de munição. – **implementado**.
  + 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”**: - **implementado**.
  + Explosão, com raio aleatório escolhido dentro os raios das munições citadas acima;
* **Paredes**: simplificadas para dois tipos: - **não implementado.**
  + 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**: - **não implementado.**
  + 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 –** implementado parcialmente**.**
  + Maior número de vitórias de partidas;
  + Em caso de empate, maior número de pontos.

**2.4 Comentários sobre especificações não implementadas**

**Cenário**: não foi implementada a eliminação de terra por ocasião de uma explosão. O terreno sempre permanece intacto após cada explosão.

O motivo é que a implementação do terreno, como será descrita posteriormente, foi trabalhosa. Implementar a perda de terreno será tão trabalhosa quanto gerar o terreno em si: será necessário manipular a matriz que mapeia as alturas do terreno, bem como criar descontinuidades de vetor normal no local da explosão. Essa implementação possivelmente necessitará de uma estrutura de dados ligeiramente diferente da que foi adotada no programa atual.

**Paredes**: O jogo só possui o modo “sem paredes”, pois não houve tempo suficiente para implementar as paredes elásticas.

**Vento**: Embora uma função que cria vento no jogo tenha sido criada, ela sempre cria vento = 0 (sem vento). Embora de implementação simples, decidiu-se por não implementar o vento nesse momento. A razão é que, como não foi implementada a destruição de terreno com explosões, poderia acontecer situações em que um terreno gerado seja tão inclinado que o vento empurraria tiros de forma que outros jogadores nunca atingiriam um jogador no fundo daquela colina inclinada.

**Critério de vitória**: No estágio atual, o jogo corretamente identifica o fim da rodada, ou seja, quando só há 1 ou 0 jogadores presentes no cenário. No caso de 1 jogador ter sobrevivido, o jogo também dá a vitória àquele jogador.

Contudo, não foi implementada ainda a tela de resultado parcial, nem o sistema de pontuação. O motivo foi tão somente falta de tempo: a implementação do jogo consumiu muito mais tempo do que o esperado inicialmente.

**4 Estratégia de Implementação**

O jogo foi implementado na linguagem C++, uma linguagem orientada a objetos. Isso permitiu que se definissem os diversos objetos que descrevem as entidades presentes no jogo.

O código fonte foi divido em vários arquivos, cada um implementando uma parte diferente do jogo:

**4.1 Estados do jogo**

Para implementar as funcionalidades desejadas, implementaram-se “estados do jogo”, também chamadas de “tela atual”. Esses estados foram apresentados na Figura 5.

Figura 5 – Fluxograma que apresenta os estados do jogo Tank Wars 3D.

**Tela Inicial**

Contém o menu principal.

**Renomear Jogadores**

Contém o menu de renomear jogadores.

**Rodada**

Cria e exibe o cenário na tela

**Resultado Parcial**

Cria e exibe o cenário na tela

**Compras**

Exibe menu de compras para cada jogador

* **Tela Inicial**: é o estado em que o jogo se encontra assim que ele é carregado. Neste estado, ele exibe o menu principal, onde se escolhem o número de rodadas e o número jogadores (Figura 6).
* **Renomear Jogadores**: neste estado, o jogo apresenta um menu no qual permite renomear cada participante do jogo.
* **Rodada**: este é o estado que apresenta o jogo propriamente dito. A cada rodada, um novo cenário é criado – terreno aleatório e disposição dos jogadores.
* **Resultado Parcial**: Ao final da rodada, o jogo transiciona para a tela de resultado parcial. Ela exibe uma tela com a pontuação e o número de vitórias de cada jogador. Quando o jogo atinge o limite de número de rodadas, ele se encerra e volta para a Tela Inicial. Caso contrário, o jogo passa para o estado Compras.   
  Obs: a tela Resultado Parcial não foi implementada no Projeto Final.
* **Compras**: Para cada jogador, é apresentada uma tela com as armas que ele consegue comprar com os dólares que possui no momento. Ao sair do menu de compras, o jogo inicia uma nova rodada.

Nota: A tela de compras também não foi implementada no Projeto Final.

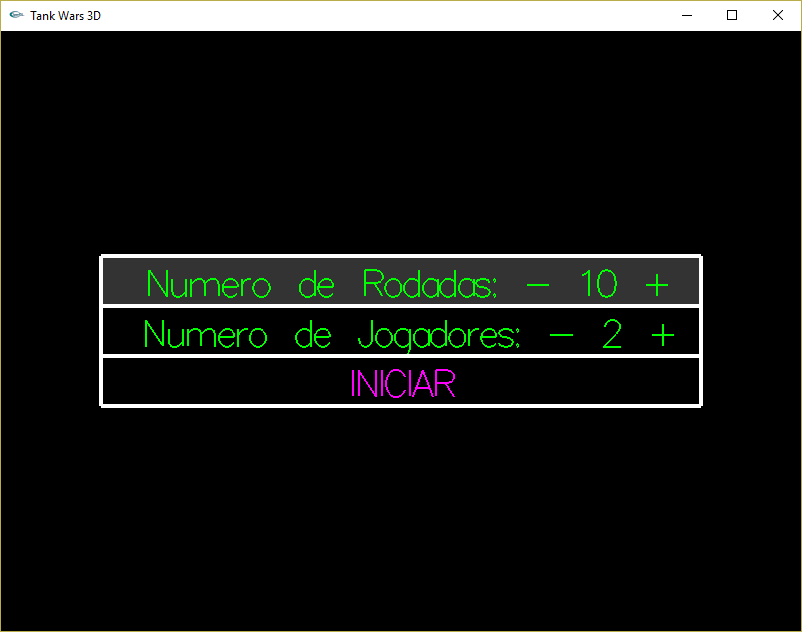


Figura 6 – Menu principal. É apresentada quando o jogo se encontra na Tela Inicial.



Figura 7 – Screenshot do Menu Renomear Jogadores.



Figura 8 – Screenshot do início de uma rodada, com todos os jogadores já posicionados.

**4.2 Objetos**

Os principais objetos implementados são os seguintes:

* **Mundo**: É o principal objeto do jogo. Ele armazena todas as configurações do jogo, tais como número de jogadores, número de rodadas totais, número da rodada atual, etc. É também o objeto Mundo que trata as interações com o usuário via teclado.
* **Jogador**: É a classe que representa um jogador do jogo. Armazena todas as informações relacionadas ao jogador: Cor, nome, armas que possui, pontuação, número de vitórias, etc.
* **Cenario**: É a classe que representa o cenário de cada rodada. Ao ser criado um objeto Cenario, são gerados o Terreno (objeto à parte) e uma lista contendo os jogadores participantes em ordem aleatória. A partir do momento que a rodada inicia e o cenário é criado, é o objeto Cenario que passa a ser responsável pela interação com o usuário. Recebe os comandos do jogador via teclado; cria o objeto Projetil (descrito mais embaixo) quando o jogador aperta *Espaço* (para atirar), executa as animações de voo do projétil, explosões e morte de jogadores, se for o caso, e por fim, verifica se a rodada encerrou após cada tiro.
* **Terreno**: É a classe que implemente a estrutura de dados para formar o relevo exibido na tela. Os detalhes de como ele é determinado foi apresentado no Anexo A.
* **Municao**: É um objeto abstrato que representa uma arma qualquer do jogo. Embora no Projeto Final tenha sido implementada somente munições explosivas, a ideia seria futuramente implementar outros tipos de munição no jogo. Independente do tipo, este objeto força com que todas as munições possuam a mesma interface, para serem usados nos outros pontos do jogo.
  + **MunicaoExplosiva**: Representa a classe de munições explosivas. Por enquanto, é a única classe de munições que o jogo possui. Cada munição explosiva é uma classe por si só:
    - Incinerador
    - IncineradorM2
    - Bomba20Kilotons
    - Bomba5Megatons
* **Projetil**: É o objeto que representa um projétil em voo. Este é o objeto responsável por realizar o cálculo da trajetória de voo e detectar se o projétil atingiu um obstáculo. Quando o projétil encontra um obstáculo, ele chama a função detonar() da Munição que representa.
* **Explosao**: Representa uma explosão na tela. Atualmente, todas as explosões são exibidas da mesma forma, diferindo-se apenas no raio de explosão. Esta classe é responsável tanto pela animação e desenho da explosão, quanto por calcular os efeitos da explosão sobre os jogadores próximos ao seu epicentro.
* **Menu**: É a classe que representa os menus exibidos pelo jogo. Ele engloba um conjunto de objetos “OpcaoMenu” (opções de menu) e, quando solicitado, os exibe na tela.
* **OpcaoMenu**: É a classe que representa cada entrada no menu. OpcaoMenu é apenas uma classe abstrata: ela contém apenas a interface que Menu necessita para funcionar. Cada tipo de menu implementa a interface OpcaoMenu de acordo com o tipo de opção:
  + **Botao**: Implementa uma OpcaoMenu como um *botão*: quando o usuário seleciona o botão e aperta *Enter*, o objeto executará uma função pré-definida. Os botões utilizados aqui fazem o jogo transicionar entre os estados descritos na seção 4.1.
  + **OpcaoAlterarValorNumerico**: Implementa uma OpcaoMenu como uma seleção que permite alterar o valor de uma variável usando as setas esquerda e direita. Exemplos deste objeto são exibidos na Figura 6.
  + **OpcaoEditarNome**: Como sugere o nome, é a OpcaoMenu que permite ao usuário alterar os nomes de cada participante. A Figura 7 exibe o menu de renomear jogadores, onde este objeto é utilizado.

**5 Comparação com o Tank Wars Original**

Serão apresentados agora alguns screenshots comparando o Tank Wars original com o jogo implementado neste Projeto.

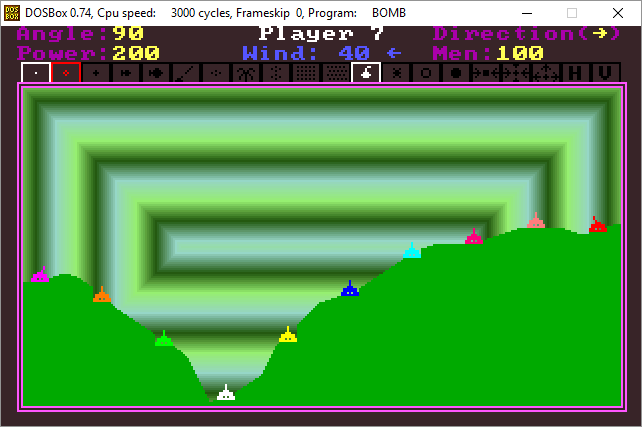


Figura 9 – Tank Wars original: início do jogo

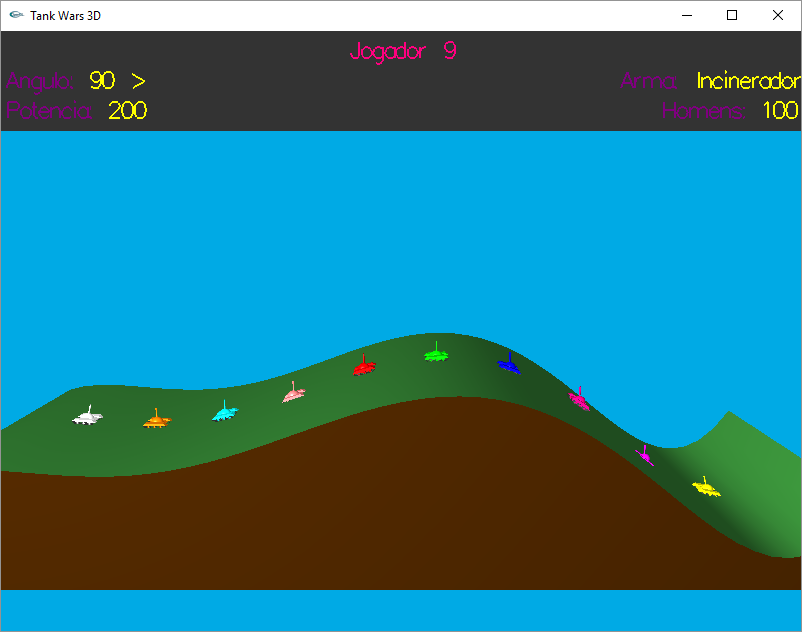


Figura 10 – Tank Wars 3D: início do jogo

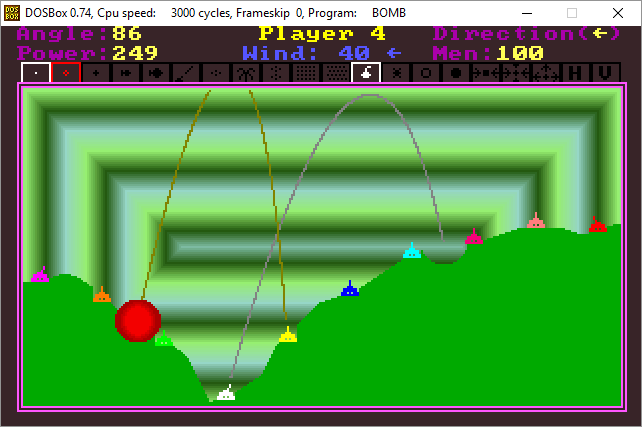


Figura 11 – Tank Wars original: explosão de um incinerador.



Figura 12 – Tank Wars 3D: explosão de um incinerador.



Figura 13 – Tank Wars original: explosão de uma bomba de 5 megatons.

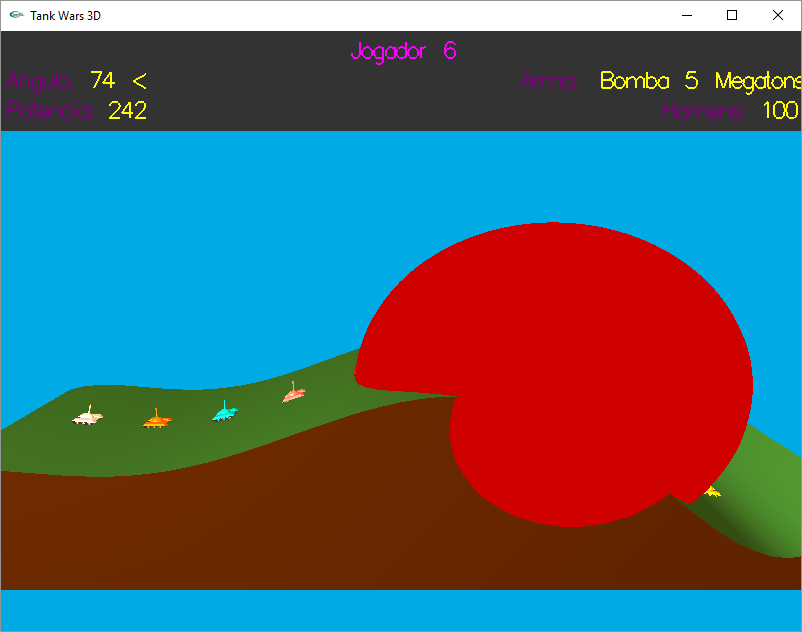


Figura 14 – Tank Wars 3D: explosão de uma bomba de 5 megatons.

**6 Conclusão**

Neste Projeto Final, foi escolhido implementar uma versão 3D do jogo Tank Wars, de Kenneth Morse. Durante o projeto, observou-se que, apesar de ser apenas um jogo bidimensional, o Tank Wars é um jogo consideravelmente complexo. Ele envolve:

* Cálculos físicos da balística de projéteis;
* Modelamento de dano causado por explosões;
* Geração procedural de terrenos.

Para implementar uma versão 3D desse jogo, além das características acima, outros detalhes precisaram ser considerados:

* Calcular os vetores normais ao terreno após ele ter sido calculado proceduralmente (detalhado no Anexo A);
* Posicionamento e definição das cores das fontes de luz;
* Desenhar modelos 3D dos tanques, projéteis, terreno e explosões.

A implementação do jogo, portanto, foi desafiador. Avaliou-se que a implementação foi feita com êxito. Embora as telas de compras e de resultado parcial não tenham sido implementadas por falta de tempo hábil, a parte principal do jogo, que é a parte em que os tanques atiram uns contra os outros, foi implementada e está funcionando perfeitamente, como demonstrado nas Figuras 9 a 14.

Com isso, o restante do jogo poderá ser implementado aos poucos, como projeto pessoal deste desenvolvedor.

**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.

**Anexo A – Coordenadas do Mundo e Desenho do Relevo**

Será descrito aqui a metodologia de gerar **proceduralmente** o relevo de cada cenário criado.

**6.2.1 Limites das Coordenadas do Mundo**

Determinou-se que o relevo se estenderá pelos limites:

* 0 ≤ x ≤ 100;
* -20 ≤ y ≤ 20;
* 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 2,0 unidades.

Com isso, tem-se que o número de pontos em cada eixo x e y são:

Eixo x: (100 / 2,0) + 1 = 51 pontos

Eixo y: (40 / 2,0) + 1 = 21 pontos

Portanto, a matriz **malha**, que representa o relevo, possui dimensões 51 x 21 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)*.

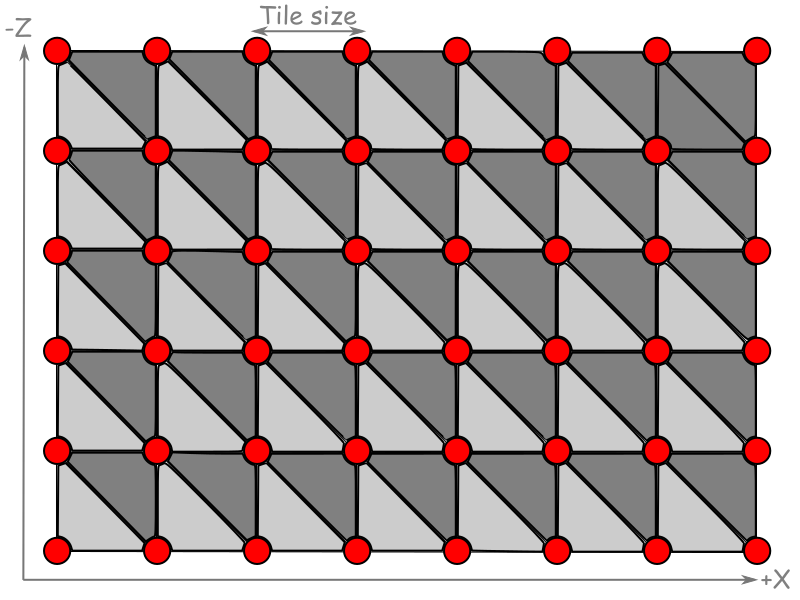


Figura 15 Ilustração do desenho da malha do terreno (TORAL, 2018).

**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:

**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.