|  |
| --- |
| Introdução A Programação 3D |
|  |
| Relatório Tanques – 3ªFase |
|  |
|  |
|  |
|  |

Bruno Couto A10664

Vítor Gomes A10658

|  |
| --- |
|  |

20/12/16

Índice

[Introdução 2](#_Toc470083129)

[Terreno 3](#_Toc470083130)

[Câmaras 4](#_Toc470083131)

[Câmara livre 4](#_Toc470083132)

[Câmara *Surface Follow* 4](#_Toc470083133)

[Câmara *Third Person* 5](#_Toc470083134)

[Iluminação 6](#_Toc470083135)

[Tanques 7](#_Toc470083136)

[Sistema de Colisões 9](#_Toc470083137)

[Sistema de Partículas 11](#_Toc470083138)

[Inteligência Artificial (*Boids*) 12](#_Toc470083139)

[Conclusão 14](#_Toc470083140)

[Bibliografia 15](#_Toc470083141)

# Introdução

Foi-nos proposto, para a disciplina de Introdução à Programação 3D, a criação de um jogo 3D usando a framework do Monogame. A proposta apresentada foi de desenvolver um jogo de batalha entre tanques, onde o jogador controla um dos tanques e tem como objectivo destruir o adversário.

O trabalho foi dividido em 3 fases. Na primeira foi pedido o render do terreno a partir de um mapa de alturas, e o controlo de pelo menos uma câmara com *surface follow*.

Na segunda, além do conteúdo anterior, a iluminação do jogo e também o render e controlo dos tanques e a respectiva interacção correcta dos mesmos com o terreno foi pedido.

Na 3ª fase e última, tendo em conta o conteúdo já desenvolvido anteriormente, é necessário várias câmaras diferentes, um sistema de detecção de colisões, o mecanismo de disparo de balas, o sistema de partículas de pó e finalmente o movimento autónomo dos tanques inimigos.

# Terreno

Para criar o terreno, foi-nos proporcionado um *height map* que indica os valores das alturas nos diferentes pontos no terreno.

Criamos um *array* que guarda a informação da cor dos pixéis da textura do *height map.* Depois geramos os vértices do terreno de acordo com os valores dentro dessa textura.

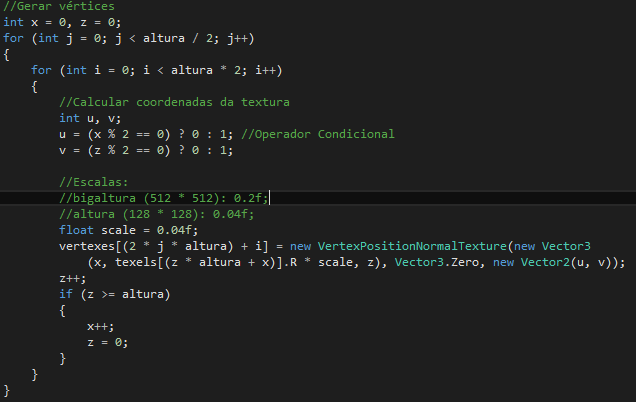


Imagem 1 Código para gerar os vértices do terreno

Depois aplicamos uma textura ao terreno que é repetida por cada *quad* presente no terreno. O mapa é criado através de um for que desenha cada faixa do terreno.

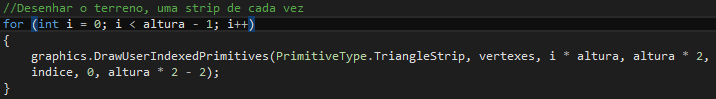


Imagem 2 Código para desenhar o terreno

# Câmaras

Como nos foi pedido no enunciado desenvolvemos três câmaras diferentes, todas elas tem o mesmo código base e controlo, apenas mudam em ligeiros aspectos, sendo eles dependentes do propósito da câmara. Existem três câmaras diferentes, sendo elas:

## Câmara livre

Para controlar esta câmara e as restantes usamos as teclas 8, 4, 5, 6 do *Numpad* – Frente, esquerda, baixo e direita, respectivamente. Também se pode fazer a câmara subir com a tecla 7 e descer com a tecla 1.Para controlar a direcção usamos o rato.

## Câmara *Surface Follow*

Para a câmara fazer *surface follow* criamos uma função que calcula a altura do terreno no local onde a câmara se encontra. Depois é só aplicar o resultado da função á câmara, deixando as coordenadas X e Z como se encontram e apenas alterando o valor Y da câmara.

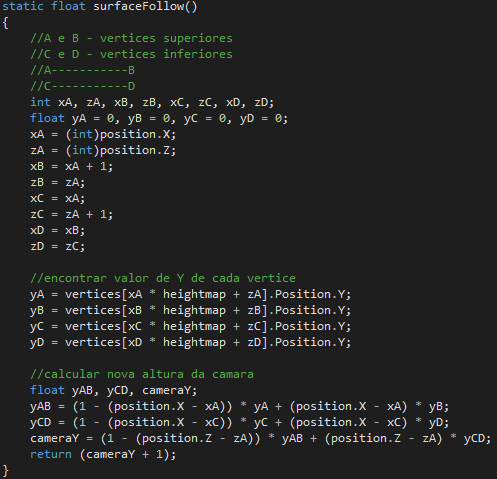


Imagem 3 Código do cálculo da altura do terreno numa posição

## Câmara *Third Person*

Esta câmara apenas segue o tanque numa perspectiva de terceira pessoa, sendo assim desnecessário controlo sobre a mesma.

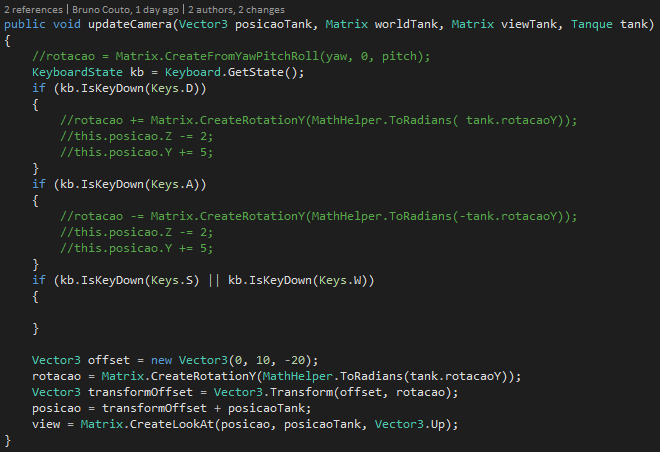


Imagem 4 Código do update da câmara

# Iluminação

Foi-nos pedido nos objectivos do projecto, aplicarmos a iluminação ao terreno, e para isso ser possível tivemos de calcular e implementar as normais do terreno, que foram feitas através de cada vértice de cada triângulo.

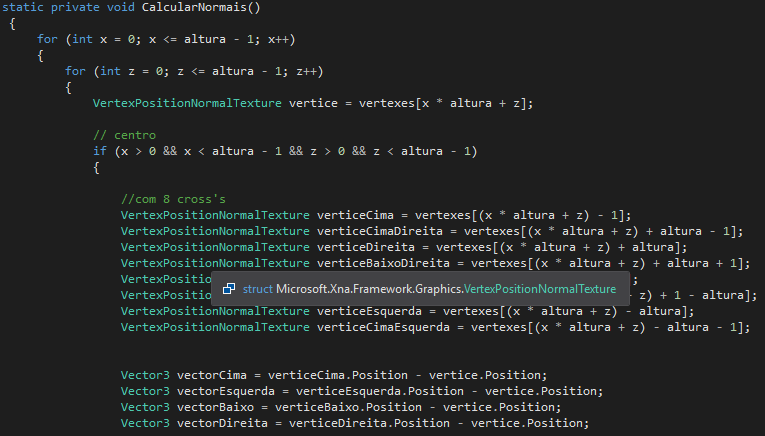


Imagem 5 Código do cálculo das normais

Depois de calculado as normais, implementámos o código da iluminação no projecto com os valores demonstrados abaixo.

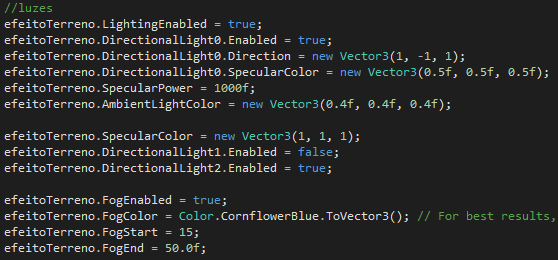


Imagem 6 Código dos efeitos da iluminação

# Tanques

Para desenharmos o tanque fizemos a classe Tank, no seu construtor temos parâmetros como *graphicsdevice* e *content manager*, mas também uma posição. Também temos a leitura dos bones do modelo 3D e os seus valores iniciais.

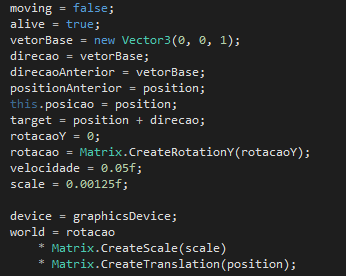


Imagem 7 Código de valores inicia no construtor

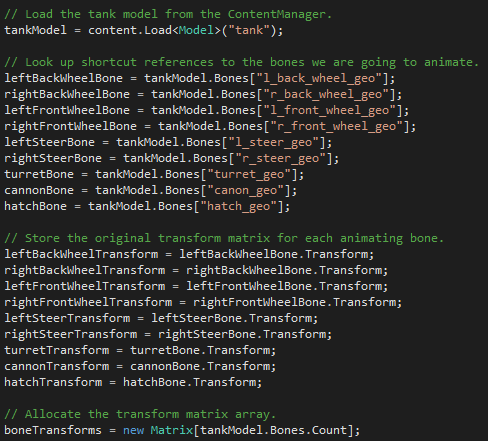


Imagem 8 Código da leitura dos bones

O tanque é movimentado pelas teclas W, A, S, D para o primeiro jogador e I, J, K, L para o segundo, também é possível controlar o canhão do primeiro com as teclas das setas.

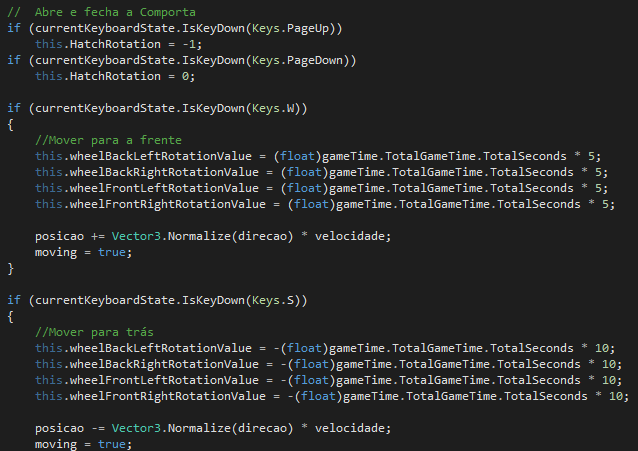


Imagem 9 Código do controlo do tanque

Para o comportamento correcto do tanque dependendo do terreno em que se encontra, usamos o *surface follow* da câmara e também calculamos a normal do tanque na posição que se encontra do mapa.

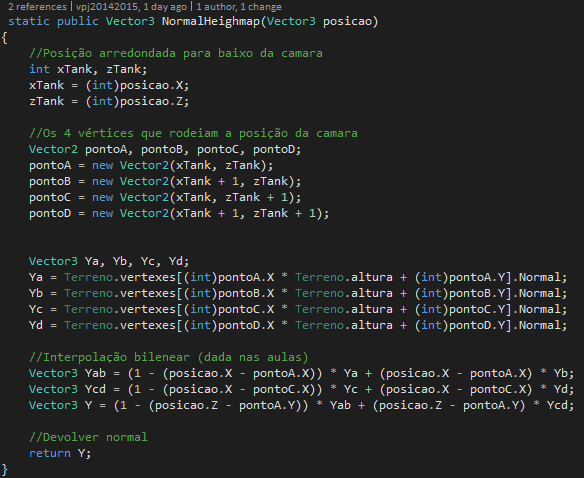


Imagem 10 Código do cálculo da normal do tanque

# Sistema de Colisões

O código de colisões do jogo tem duas partes, a dos tanques e a das balas. Ambos posicionados no update da main class, claro podia ser uma classe única dedicada ao funcionamento deste sistema mas neste caso não era necessário.

No código respectivo aos tanques se chegar a haver intersecção entre eles os tanques têm o comportamento de parar completamente.

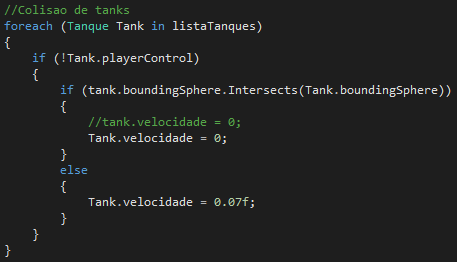


Imagem 11 Código de colisão entre tanques

Em relação às balas, temos duas classes, uma para as balas e outra para as listas das balas activas e desactivadas e uma função que faz o funcionamento pretendido das colisões, ou seja, se uma bala atingir um tanque o mesmo é “destruído” e pára o seu funcionamento normal.

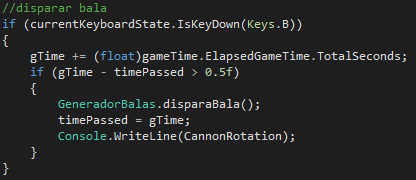


Imagem 12 Código de disparo da bala

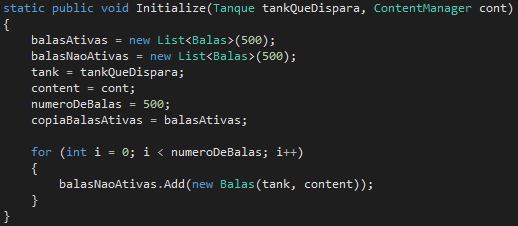


Imagem 13 Construtor da classe GeradorBalas

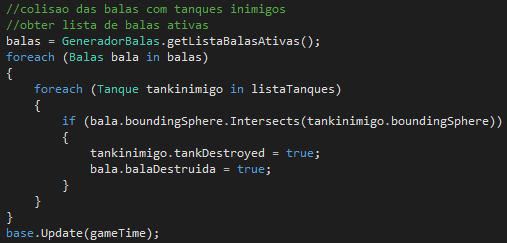


Imagem 14 Código das colisões das balas

# Sistema de Partículas

Para o sistema de partículas criamos duas classes, SistemaP e particulasTanque. Uma com a lista e quantidade de partículas e outra para o controlo e funcionamento das mesmas.

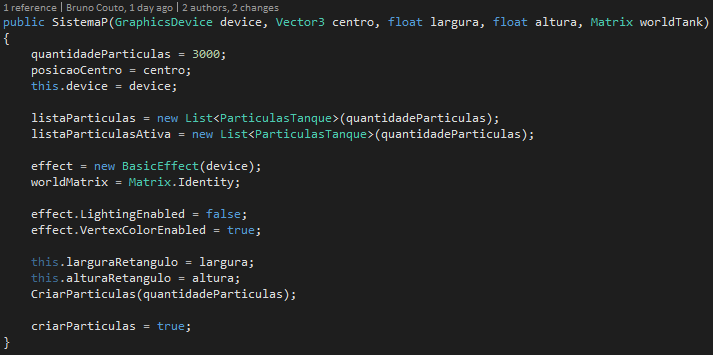


Imagem 15 Construtor do sistema de partículas

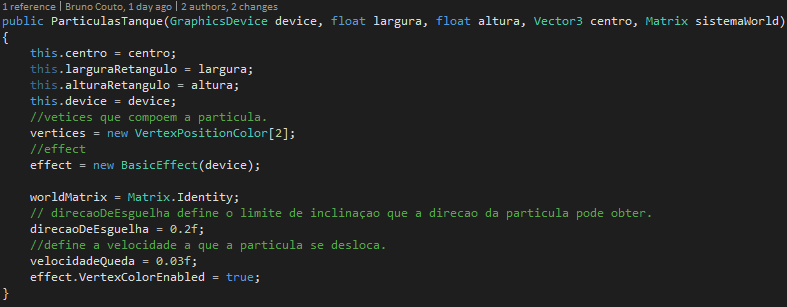


Imagem 16 Construtor das partículas do tanque

# Inteligência Artificial (*Boids*)

Para os tanques inimigos terem um comportamento de perseguição ou de fuga baseámo-nos no trabalho de *boids* de *Craig Reynolds*. Um modelo computadorizado que retrata comportamentos do movimento animal como bandos de pássaros ou cardumes de peixes.

Os movimentos dos tanques inimigos dependem da posição e direcção do *player*, ou seja, em momentos e situações diferentes os inimigos vão ter comportamentos de perseguição ou fuga.

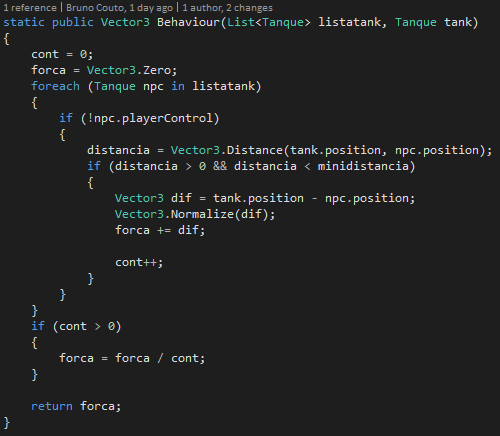


Imagem 17 Construtor da classe *Boid*

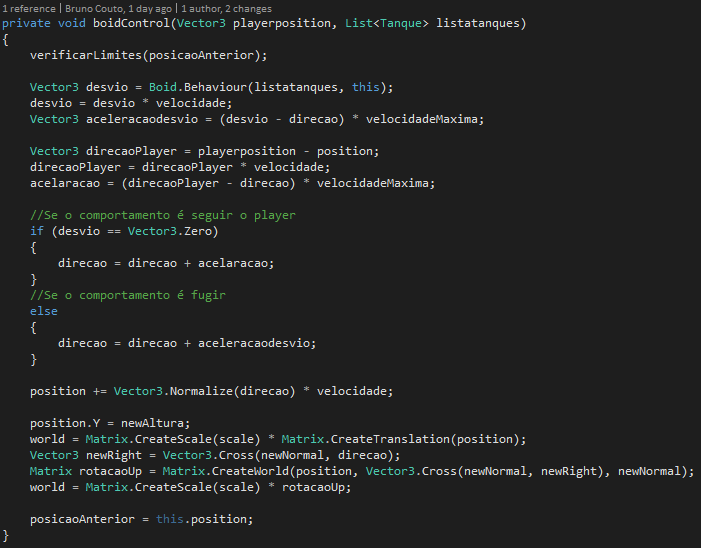


Imagem 18 Função que controla o comportamento dos tanques

# Conclusão

Na primeira fase do trabalho sucedemos em desenvolver competências básicas para a programação 3D, sendo elas, a criação de um algoritmo que seja capaz de gerar um terreno em 3D através da textura proporcionada. Também nos permitiu tomar conhecimento do funcionamento de uma câmara em 3D e o controlo da mesma.

Na segunda fase houve dificuldades sentidas na execução de cada um dos objectivos propostos, mais evidentes na resolução das normais do terreno, mas foram desenvolvidos com sucesso e permitiu-nos ganhar mais conhecimento em como criar um jogo em 3D em Monogame.

Nesta última fase efectuámos o código necessário para o funcionamento pretendido do jogo com sucesso, embora foram sentidas algumas dificuldades no âmbito do comportamento dos tanques inimigos, ou seja, os *boids*, mas com algumas pesquisas e estudo de uns artigos no assunto, as dúvidas foram esclarecidas.

Embora esta seja a última iteração do jogo, ao longo do tempo em que foi sendo desenvolvido, reparamos que para realmente ser um jogo era necessário continuar com o desenvolvimento para efectuar os devidos afinamentos necessários para um jogo limpo e optimizado, algo que foi possível perceber no desenvolvimento do mesmo.

# Bibliografia

<http://www.red3d.com/cwr/boids/>