

|  |
| --- |
| Técnicas Avançadas de Programação 3D |
|  |
| Trabalho de Grupo  Relatório Do Trabalho Prático  Da autoria de:  José Carlos Ferreira 16609  Nuno José da Silva Antunes 15352 |



Índice

[Introdução 3](#_Toc105540671)

[Desenvolvimento 4](#_Toc105540672)

[Shader Enemy 4](#_Toc105540673)

[Shader FireMag 5](#_Toc105540674)

[Scared Shader 6](#_Toc105540675)

[Desabilitar Oceanos 7](#_Toc105540676)

[Pequena Melhoria 7](#_Toc105540677)

[Nova Textura de Oceano 7](#_Toc105540678)

[Animação da Textura 7](#_Toc105540679)

[Textura no Interior da Esfera 8](#_Toc105540680)

[Melhoria Nuvens 9](#_Toc105540681)

# Introdução

Para este trabalho prático, desenvolvido no âmbito da unidade curricular de Técnicas Avançadas de Programação 3D, foi definido pelo professor da UC que os alunos deveriam de desenvolver um jogo com shaders desenvolvidos por eles, com isto concluísse que a característica do projeto envolva a criação de um shader (Surface Shader/Unlit Shader) ou mais por aluno.

Para realizar este projetos vamos construir vários *Surface Shaders* , *Surface Shaders* é uma abordagem mais simplificada na criação e elaboração de código que torna muito mais fácil escrever *shaders* do que usar programas vertex/pixel s*hader* de baixo nível; Com o uso da função surf void surf (Input IN, inout SurfaceOutput o) faz uso dos input implementados no shader (por exemplo uma textura, valores float, colorpicker, entre outros) e relaciona com o output do shader realizando assim a implementação do shader no mundo.

Construímos também alguns *Unlit Shaders, Unlit Shaders* são shaders que permitem criar materiais que não são afetados pela iluminação, mas inclui opções para tipo de superfície, cor emissiva etc. Com o uso da opção Pass faz uso das suas structs appdata e v2f, e também da função v2f vert(appdata v) onde podemos alterar o objeto como exemplo aplicar rotações, por fim também é utilizada a função fixed4 frag(v2f i) : SV\_Target que é executada sempre no fim, nesta função é onde aplicamos texturas, cores, etc.

# Desenvolvimento

## Shader Enemy

Este shader foi criado para inimigo, o objetivo deste shader é apresentar o inimigo com a sua textura, mas quando colide com a bala o inimigo começa a dissolver-se.

Para criarmos o dissolve usamos um noise texture, começamos por fazer um clip a noise texture menos o valor que pretendemos dissolver. Usamos o clip para remover os valores negativos.

Por fim verificamos onde a textura de noise menos o valor de dissolve menos 0.05 se esse valor for maior que 0 começamos a realizar o dissolve, o 0.05 é essencial pois cria um gap entre o dissolve e a texture transmitindo a ideia de que o zombie está a ser queimado.

Uma imagem com céu, água

Descrição gerada automaticamenteUma imagem com céu, água, exterior

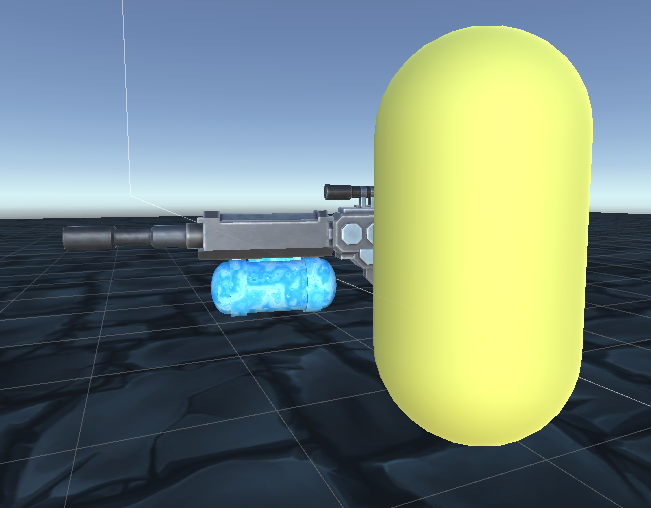
Descrição gerada automaticamente

## Shader FireMag

O shader FireMag serve para mostrar que tipo de arma o jogador vai ter e que tipo de balas ira disparar. A textura é definida aleatória, ou pode ser de gelo ou fogo, para criarmos algo mais atrativo decidimos aplicar um movimento dando a ideia que o carregar está carregado com um líquido. Para o movimento foi construída com o auxílio de uma função de onda senoidal como parâmetro de entrada o \_Time para o parâmetro de entrada se alterar consoante o tempo, e utilizamos a textura noise para obter valores e utilizar como coordenadas na nova textura de água ou fogo.

Fazemos recurso da função senoide para alterar os valores provenientes da textura *noise* e implementar como coordenadas uv “aleatórias” na nova textura de água ou fogo, porém apesar de existir uma animação a mesma é muito repetitiva e não fica percetível nenhuma deslocação ou fluidez da água ou fogo.

Uma imagem com interior, eletrodoméstico

Descrição gerada automaticamente

## Scared Shader

O scared shader é um post processing shader que coloca zonas do ecrã preto no ecrã conforme o utilizar vai falhando tiros, pois ao falhar esses tiros aumento o nível de scaredLevel o que faz com que o raio de visão diminua e o ecrã fique mais preto.

Começamos por criar duas funções bool isInside(float2 centerXY, float rad, float2 xy) e a função float PercentageOffFade(float minRadius, float maxRadius) a função isInsisde serve para verificar se o circulo está dentro do ecrã, a função PercentageOffFade serve para calcular conforme o mínimo e o máximo do raio, a percentagem que Fade que devera ter.

Para desenhar foi simples começamos transformamos as coordenadas UVs para pixéis de ecrã, de seguida calculamos o centro do ecrã, uma vez com o tamanho do ecrã começamos a desenhar. Para desenhar criamos uma condição onde verificamos se não está dentro ecrã através da função isInside, se não estiver calculamos a distancia daquele ponto especifico para o centro atráves da rais quadrada para isso utilizamos o sqrt. Colocamos a distancia para o centro entre 0 e 1, com a função PercentageOfFade calculamos o valor de fade e por fim aplicamos a cor preta menos a percentagem criando assim o fade.



## Triplanar Shader

posteriormente antes de passarem para coordenadas de uv da nova textura de água, dando assim.

Conclusão

A elaboração deste relatório teve como objetivo expor o trabalho desenvolvido em contexto da unidade curricular Técnicas Avançadas de Programação 3D.

Neste projeto tivemos a oportunidade de pôr em prática grande parte dos conhecimentos adquiridos na unidade curricular. Após este desafio ficamos mais capacitados para quando formos abordados para a criação de novos shaders.

Enriquecemos conhecimentos da utilização da ferramenta do Unity através das pesquisas e aprofundamos sua aplicação em casos de forma mais assertiva, fruto da criação deste projeto.

Conseguimos aprender funcionalidades que irão servir como base num futuro projeto.

Deixamos um agradecimentos ao docente Moisés Moreira, pela disponibilização da documentação e apoio na criação deste projeto.