

|  |
| --- |
| 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](#_Toc105462485)

[Desenvolvimento 4](#_Toc105462486)

[Shader Enemy 4](#_Toc105462487)

[Melhoria 4](#_Toc105462488)

[Intensidade de Pontos Iluminados 5](#_Toc105462489)

[Desabilitar Oceanos 5](#_Toc105462490)

[Pequena Melhoria 5](#_Toc105462491)

[Nova Textura de Oceano 6](#_Toc105462492)

[Animação da Textura 6](#_Toc105462493)

[Textura no Interior da Esfera 7](#_Toc105462494)

[Melhoria Nuvens 7](#_Toc105462495)

# 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

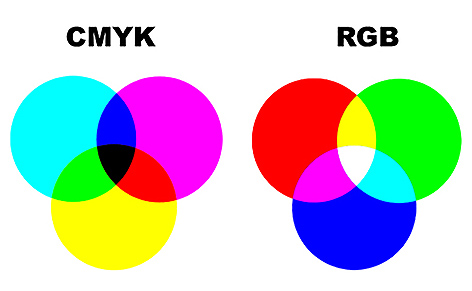
Descrição gerada automaticamenteUma imagem com céu, água, exterior

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 gele ou de fogo.

## Intensidade de Pontos Iluminados

Para intensificarmos os pontos luminosos na parte sombreada da terra precisamos primeiramente de perceber como a cor amarela é formada; De acordo com a composição das cores RGB e ao contrário das cores CMYK o amarelo é construído a partir de 2 cores primárias, sendo elas a cor vermelha e a cor verde (como observamos na imagem), com isso sabemos que caso os canais de cor vermelho e verde sejam maiores que o azul, o resultado será sempre semelhante com a cor amarelo, sendo assim como os pontos luminosos na figura são representados pela cor amarela podemos intensificar os valores dos canais vermelho e verde para obtermos uma cor mais vibrante de acordo com um *slider* presente no shader.

## Desabilitar Oceanos

Em semelhança à mecânica implementada anteriormente, para desabilitar os oceanos na textura iluminada analisamos os canais de cor da textura, sabendo que o oceano é nomeadamente azul concluímos que caso o canal de cor azul seja maior que os restantes 2 canais (vermelho e verde) a cor resultante será o azul, com isso podemos filtra essa informação dos canais de cor, e caso a condição seja verdadeira simplesmente descartamos o pixel.

### Pequena Melhoria

E para descartamos a possibilidade de descartarmos cores próximas do branco (polos gelados) acrescentamos à condição que os canais vermelho e verde tem de conter valores inferiores a 0,6 por exemplo, pois o azul do oceano é um azul escuro logo não corremos o risco de termos azul claro.

## Uma imagem com azul, ovo, colorido, bola Descrição gerada automaticamenteNova Textura de Oceano

Para ser implementada uma nova textura na zona oceânica do lado iluminado do planeta é apenas necessário colocar uma condição que empesa o *discard* do pixel da mecânica anteriormente representada, e coloque uma nova textura no seu lugar, neste caso a nova textura de água/oceano.

### Animação da Textura

De acordo com o vídeo que o docente entregou conseguimos perceber que a nova textura do oceano está sendo influenciada por uma outra textura *noise* que está a auxiliar a sua animação com a intenção de a tornar mais credível e natural nos seus movimentos, o que por um lado não é o mais recomendado porque um oceano visto do espaço deveria de conter movimentos muitos mais lentos e não deveria ser tão percetível a sua “*pixalização*”.

Para a conclusão desta mecânica construi 3 animações, todas elas foram construídas 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 utilizo a textura *noise* para obter valores e utilizar como coordenadas na nova textura de água.

**A primeira animação** faz recurso da função senoide para alterar os valores provenientes da textura *noise* e implementar como coordenadas uv “aleatórias” na nova textura de água, porém apesar de existir uma animação a mesma é muito repetitiva e não fica percetível nenhuma deslocação ou fluidez da água (mais semelhante com a do exemplo do docente).

**A segunda animação** faz recurso da função senoide para alterar os valores nas uv coordenadas da textura *noise* o que implementa uma fluidez da água na esfera porem apesar disso a textura fica monótona.

**A terceira animação** faz uso da lógica das 2 animações anteriores, implementa a onda senoide para influenciar as coordenadas de uv da textura *noise* e depois volta a influenciar o seu resultado posteriormente antes de passarem para coordenadas de uv da nova textura de água, dando assim fluidez e animação à textura, sendo a terceira a melhor opção para uma textura mais credível de água utilizando os recursos fornecidos.

## Textura no Interior da Esfera

Caso o shader representado neste relatório fosse da categoria *unlite shader* e podessemos implementar a mecânica “ *pass { }*” o menor número possível de *pass* para implementar este projeto seria 2 pass, pois para a implementação da nova textura no interior da esfera utilizaremos parâmetros diferentes, como é o caso do *cull front* este parâmetro faz com que a textura não seja renderizada nas superfícies direcionadas para a camara mas permite que sejam vistas nos planos interiores, com isso quando se desabilitar os oceanos vai ser possível ver no interior a textura do *Pickle Rick*, e se não fizéssemos uso de 2 *pass* tanto as texturas anteriores e a do Pickle Rick seriam visíveis no interior da esfera, mas como estamos a utilizar um *suface shader*, a implementação dos *pass* é diferente do *unlit shader,* mas o conceito é o mesmo.

Mas contudo conseguimos perceber que o pickle rick contem uma mapa de alpha na sua composição, por isso para ser possível a implementação dos canais de alpha que permitem transparência de texturas temos de acrescentar o parâmetro fullforwardshadows alpha:blend junto com os restantes #pragma.

## Uma imagem com diferente, capacete Descrição gerada automaticamenteMelhoria Nuvens

Nuvens foi uma melhoria que implementei para dar um tom mais realista na simulação de planeta terra, para a sua construção utilizei um dos canais de cores da textura *noise*, em que o canal representa os valores de *alpha* e *albedo*, sendo que caso o valor seja 0 o pixel seria transparente e na eventualidade de ser valor 1 o pixel seria opaco de cor branca, juntando um vetor de movimento ás coordenadas de uv da textura e um intervalo que intensifique e diminua o valor do pixel com o auxilio de uma onda senoide conseguimos criar nuvens a rodar em torno a nossa esfera com surgimento e desaparecimento de novas nuvens.

Conclusão

Após este desafio fiquei mais capacitado para quando formos abordados para a criação de novos shaders.

Enriqueci 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.

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

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