

|  |
| --- |
| Técnicas Avançadas de Programação 3D |
|  |
| Trabalho Individual  Relatório Do Trabalho Prático  Da autoria de:  José Carlos Ferreira 16609 |



Introdução

Para este trabalho prático, desenvolvido no âmbito da unidade curricular de Técnicas Avançadas de Programação 3D, foi entregue um enunciado explicando as características que os alunos teriam de desenvolver para conclusão do mesmo, sendo acompanhado de um vídeo elaborado pelo docente da disciplina Moisés Moreira para auxiliar a compreensão do mesmo, com isto concluísse que as características do projeto envolvia a criação de um só shader (Surface Shader) presente num material de uma esfera no mundo 3D no Unity, no qual o tema seria a criação da terra vista do espaço com as seguintes características:

* Existir uma mecânica que simule o efeito do sol (Ponto de luz), sendo que a parte iluminada terá uma textura representativa do planeta terra iluminado pelo sol, e na parte “sombreada” uma textura que represente o planeta terra na ausência de luz (noite);
* Existência de um *slider* que intensifique e diminua os “pontos iluminados” na textura que representa o planeta terra na ausência de luz;
* Existência de um *toggle* que quando ativado faça desaparecer as zonas aquáticas/oceanos da textura representativa do planeta terra iluminado;
* Existência de um *toggle* que quando ativado faça aparecer uma nova textura na zona aquáticas/oceanos da textura representativa do planeta terra iluminado;
* Existência de um *toggle* que quando ativado ative uma animação que simule a movimentação das zonas aquáticas/oceânicas na textura criada na característica anterior;
* Existência de um *toggle* que quando ativado represente uma textura do *Pickle Rick* no interior da esfera;

Para Realizar este projetos vamos construir um *Surface Shader* , *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.

Desenvolvimento

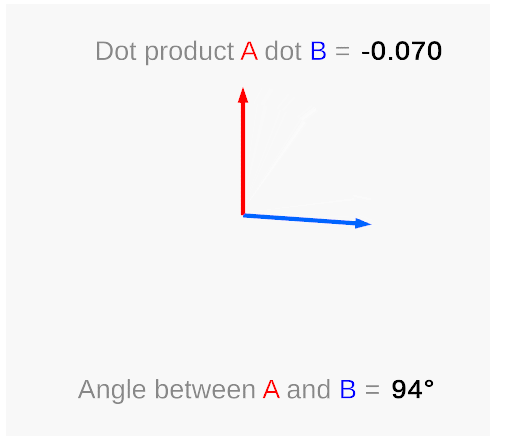
## Efeito de Dia e Noite

Para construirmos o efeito de um ponto de luz num objeto, neste caso sol e planeta terra, precisamos primeiramente de entender como são construídas as sombras/iluminação num objeto e como elas são desenvolvidas, para isso podemos utilizar um dos modelos mais comuns *Blinn-Phong*.

*Blinn-Phong* para a sua implementação utiliza vários recursos como o vetor do ponto de iluminação, normal da superfície do objeto …, contudo para a simulação simples de uma área sombreado precisamos apenas do vetor do ponto de iluminação e a normal da superfície do objeto;

Como é referido no vídeo que o docente disponibilizou, o ponto de iluminação que iremos utilizar no projeto será a *Direction Light* do nosso senário, para obtermos a mesma utilizamos a constante nomeada de \_WorldSpaceLightPos0 e para obtermos a normal na superfície do objeto utilizamos o valor o.Normal contudo por via das dúvidas poderá não estar normalizada por isso se recomenda a normalizar com o método normalize();

“\_WorldSpaceLightPos0: float4 Directional lights: (world space direction, 0). Other lights: (world space position, 1).”

Obtendo os 2 vetores podemos calcular relacionamentos entre eles; O método Dot Product, Dot(A,B) retorna o produto escalar entre 2 vetores, vetor A e vetor B, neste caso sendo A o vetor da luz direcional e B o vetor da normal da superfície do objeto; O seu resultado será comprimido entre os valores [1 ; -1], sendo 1 ambos os vetores tem exatamente a mesma direção, e -1 os vetores são exatamente opostos;

Uma imagem com capacete

Descrição gerada automaticamenteCom isto obtemos um ponto intermédio que será o valor 0 que representará o *break point* entre o lado iluminado e o lado sombreado, com isso podemos então distinguir que o intervalo positivo fica com a textura do planeta terra iluminado (pois é a área iluminada pelo ponto de luz), e no intervalo negativo implementamos a textura com o planeta terra noturno.

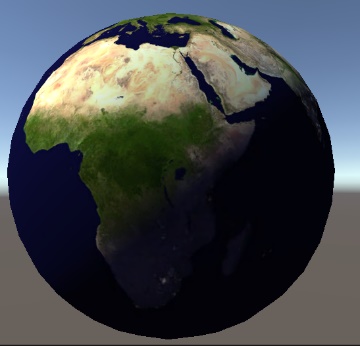
### Melhoria

Como melhoria desta mecânica decidi implementar uma mecânica controlada por um *slider* [0; 1] que acrescenta um fade entre a parte iluminada e sombreada do objeto, para tornar o efeito mais realista;

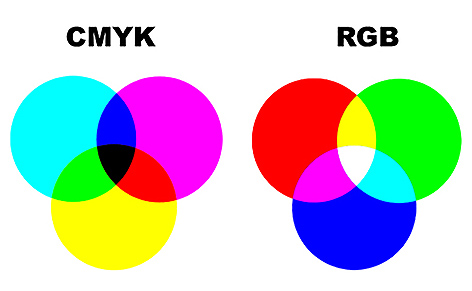
Para obtermos este efeito necessitamos de “misturar “ambas as texturas de acordo com a distância entre a parte completamente sombreada e a completamente iluminada, para isso foi preciso implementar a seguinte mecânica:

Entre a condição de completamente iluminada e completamente sombreada criamos um intervalo, em que de forma simétrica de acordo com a posição do pixel atribuímos uma percentagem A à textura iluminada e outra percentagem B à textura sombreada, sendo que A+B tem valor de 1 (100%).

Uma imagem com texto

Descrição gerada automaticamente

## 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.