Aluno: Renilton Ribeiro de Oliveira Matrícula: 11328395

# Atividade 4

## Introdução

Esse é um relatório sobre a quarta atividade de computação gráfica, implementação de diversos tipos de filtros de texturas disponíveis no three.js, com o objetivo de fazer uma análise comparativa entre eles, e familiarizar os alunos com o conceito de mapeamento de textura tradicionalmente utilizado em computação gráfica, essa disciplina é ministrada pelo professor Christian Pagot, na UFPB, no período 2020.2. Para esse trabalho foi utilizada a linguagem de programação Javascript juntamente com o framework disponibilizado pelo professor, o programa template que renderiza um cubo com textura, como ilustrado na Figura 1.

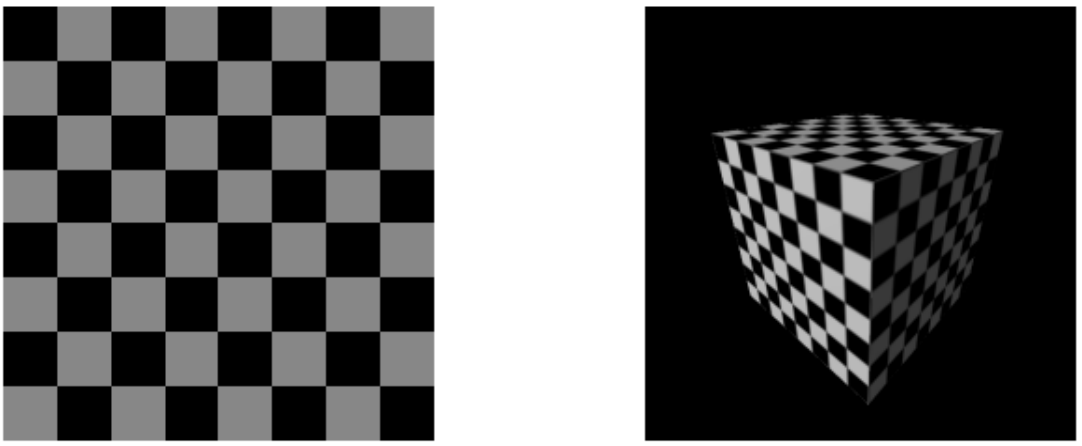

*fig 1. Template que renderiza um cubo com textura*

## Estratégia

Optei pela por dedicar tempo a entender a documentação do three.js responsável pela parte utilizada para manipular textura, escolhi usar checkerboard para melhor visualização dos resultados.

## Resultado

Para modificação dos filtros e da anisotropia para realização do trabalho foram modificados apenas os trechos no código correspondente a esse exemplo abaixo, onde a primeira linha se refere a propriedade magnification, a segunda a minification e a terceira linha ao fator anisotrópico aplicado a textura, mais a frente será mais detalhado.

```
texture6.magFilter = THREE.NearestFilter;
texture6.minFilter = THREE.LinearFilter;
texture6.anisotropy = 1;
```

Bom, comecei testando as diferenças entre os filtros de ampliação, NearestFilter retorna o valor do elemento de textura que está mais próximo (na distância de Manhattan) das coordenadas de textura especificadas. LinearFilter é o padrão e retorna a média ponderada dos quatro elementos de textura que estão mais próximos das coordenadas de textura especificadas e pode incluir itens embalados ou repetidos de outras partes de uma textura, dependendo dos valores de wrapS e wrapT , e do exato mapeamento.

Após comparar o resultado de ambos é notado que a texturização apresentada pelo a NearestFilter dá um pouco diferente tamanhos quadrados, uma vez que a textura e a grade da imagem não combinam perfeitamente.

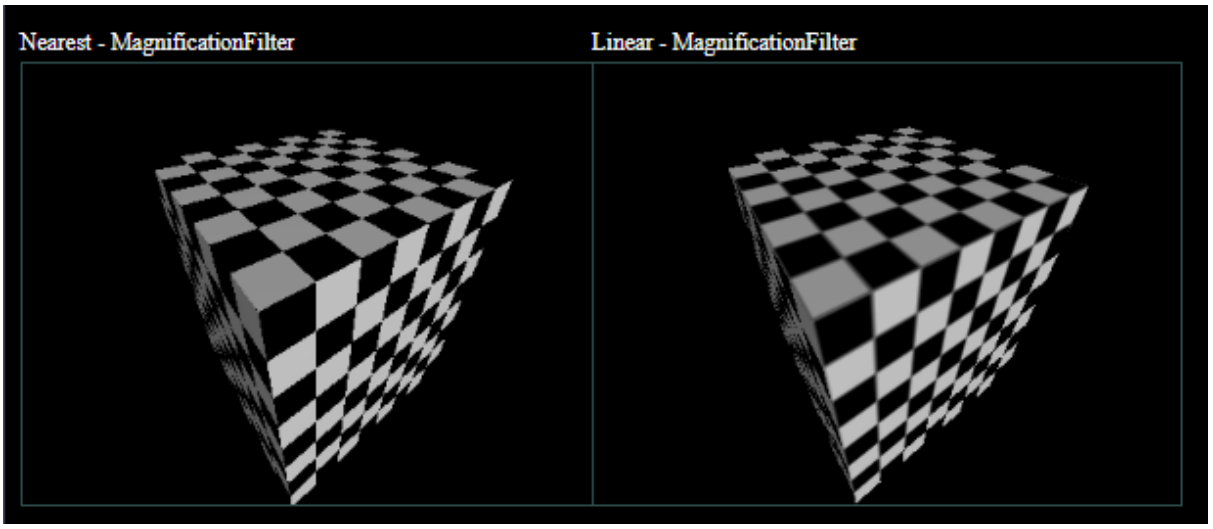


*figura 2. Usando os filtros de Magnification NearestFilter e LinearFilter.*

Bom, Filtro anisotrópico é um método de melhorar a qualidade de imagem das texturas em superfícies que estejam à distância e num ângulo pronunciado em relação ao ponto de vista. Reduzindo mais o detalhe numa direcção que noutra, estes efeitos podem ser reduzidos.

Técnicas antigas, tais como filtros bilineares e trilineares não levam em conta o ângulo da superfície do qual esta é vista, o que pode resultar em limites em serra ou texturas borradas.

Para a próxima parte vamos fazer comparações utilizando filtros de minificação, esse filtro é para uso com a propriedade minFilter de uma textura, eles definem a função de minimização de textura que é usada sempre que o pixel sendo texturizado é mapeado para uma área maior que um elemento de textura (texel), abaixo nas próximas duas imagens estamos utilizando os mesmo dois filtros da ampliação o NearestFilter e LinearFilter na minificação e aplicando diferença de anisotropia em ambos.

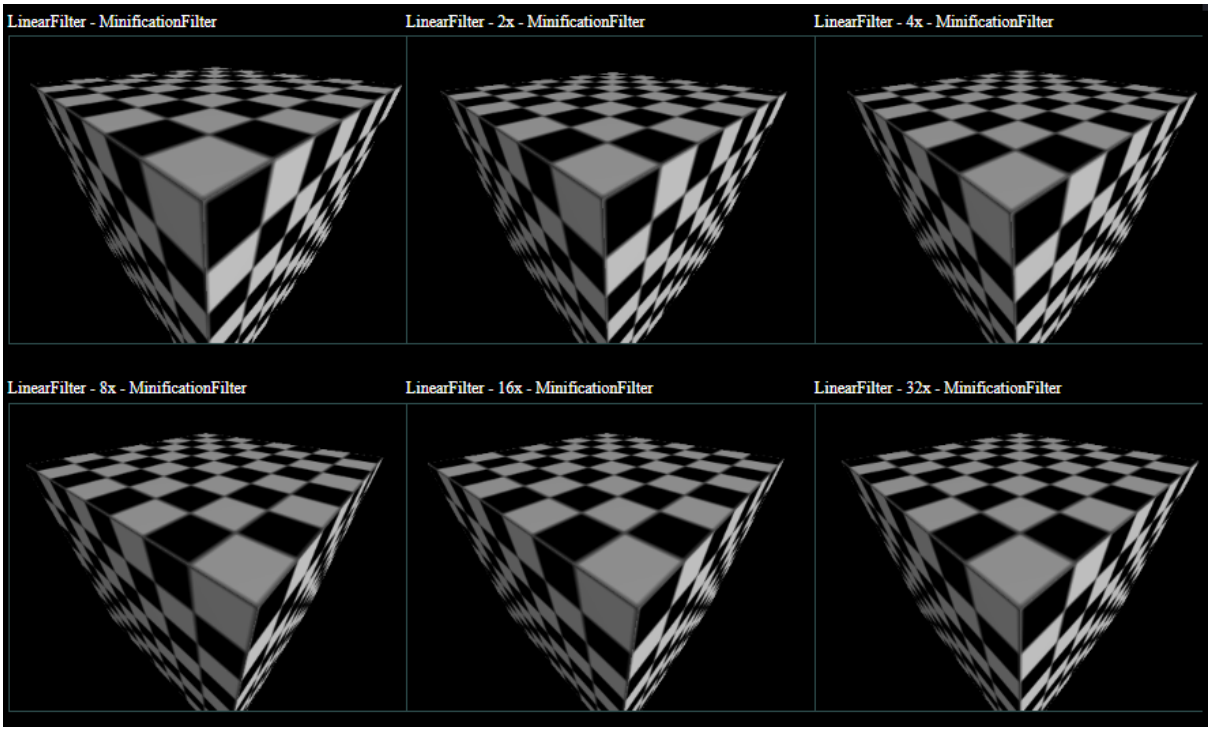


*figura 3. Usando os filtros de Minificação LinearFilter.*

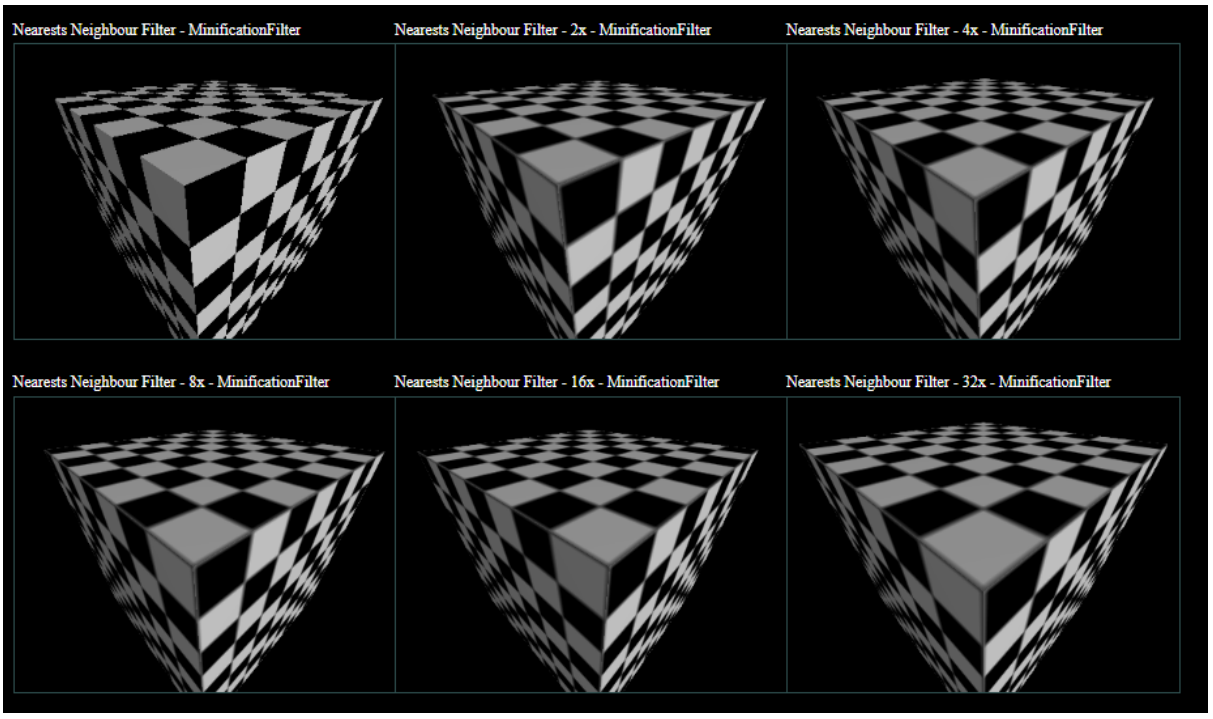


*figura 4. Usando os filtros de Minificação NearestFilter.*

Na parte de minificação o NearestMipmapNearestFilter escolhe o mipmap que mais se aproxima do tamanho do pixel sendo texturizado e usa o critério NearestFilter (o texel mais próximo do centro do pixel) para produzir um valor de textura, abaixo podemos observar esse filtro sendo

utilizado com mudança de anisotropia sobre a textura, pode-se notar uma melhora no tratamento da imagem, mas a partir de um certo valor não se nota muita diferença.

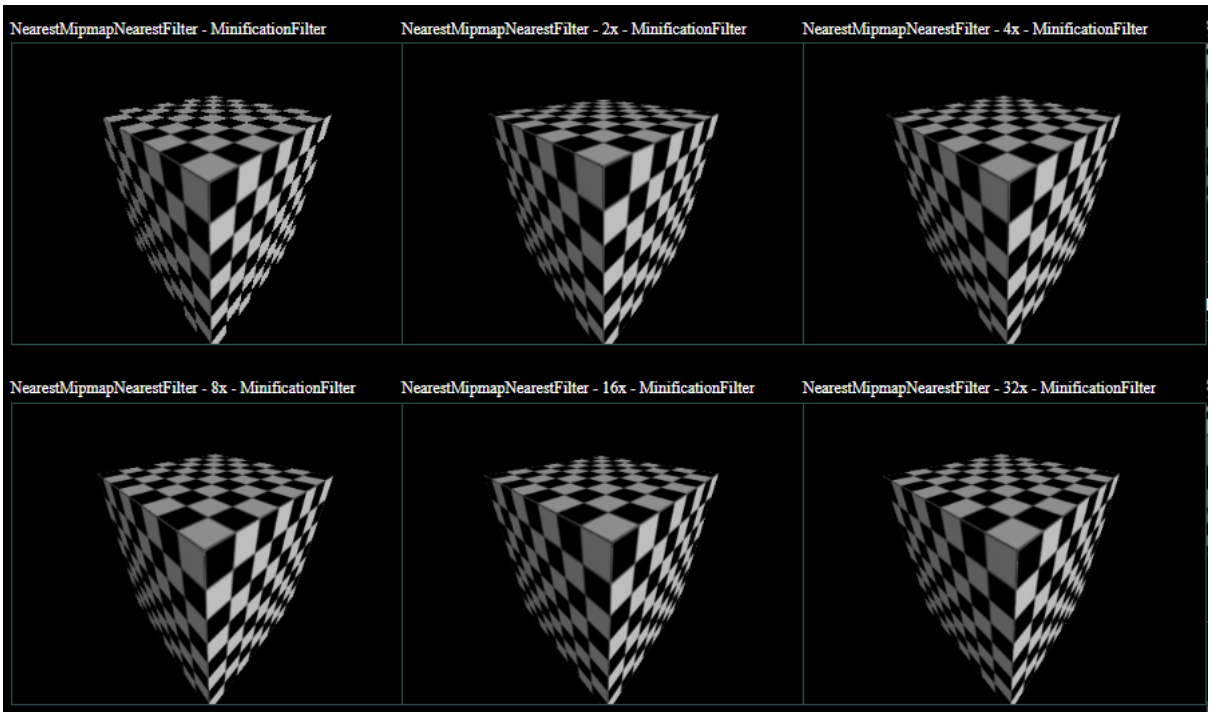


*figura 5. Usando o filtro de minificação de NearestMipmapNearestFilter com variação de anisotropia.*

Já o NearestMipmapLinearFilter escolhe os dois mipmaps que mais se aproximam do tamanho do pixel sendo texturizado e usa o Critério de NearestFilter para produzir um valor de textura de cada mipmap. O valor final da textura é uma média ponderada desses dois valores.

O resultado na variação de anisotropia é similar ao anterior.

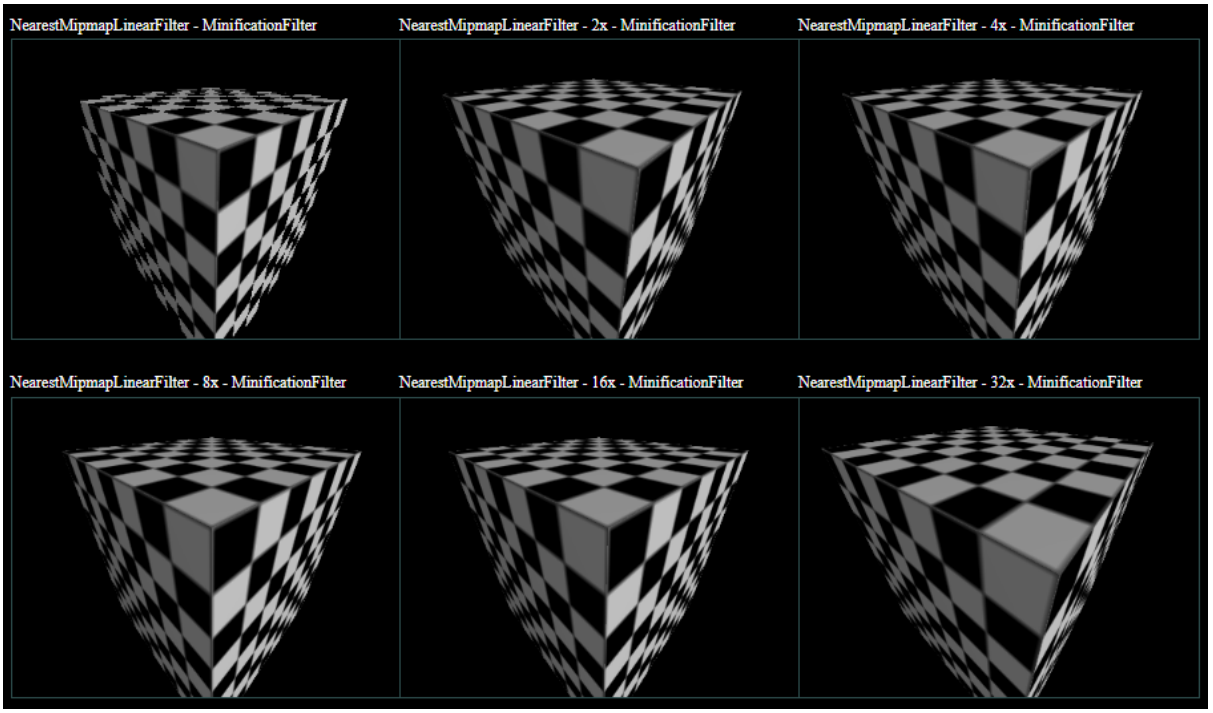


*figura 6. Usando o filtro de minificação de NearestMipmapLinearFilter com variação de anisotropia.*

LinearMipmapNearestFilter escolhe o mipmap que mais se aproxima do tamanho do pixel sendo texturizado e usa o critério LinearFilter (uma média ponderada dos quatro texels mais próximos do centro do pixel) para produzir um valor de textura.

Apesar do critério LinearFilter fazer uma texturização mais borrada no objeto, a princípio a olho nu observei uma ação mais significativa, mais fácil de ser vista do que usando o critério NearestFilter. O resultado na variação de anisotropia é similar no quesito de se notar a sua aplicação a partir de certo ponto, não valendo a pena no meu ponto de vista pelo custo usar um filtro superior a 8x.

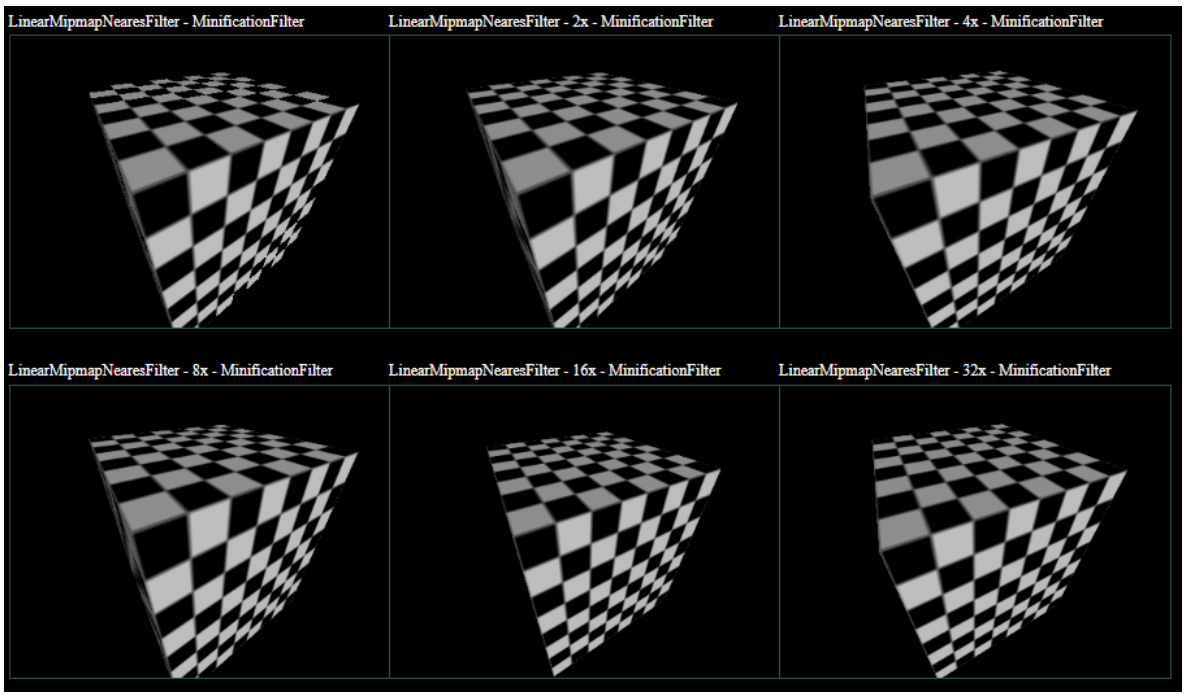

*figura 7. Usando o filtro de minificação de LinearMipmapNearesFilter com variação de anisotropia.*

LinearMipmapLinearFilter é o padrão e escolhe os dois mipmaps que mais se aproximam do tamanho do pixel sendo texturizado e usa o critério LinearFilter para produzir um valor de textura de cada mipmap. O valor final da textura é uma média ponderada desses dois valores. O resultado na variação de anisotropia é similar ao anterior.

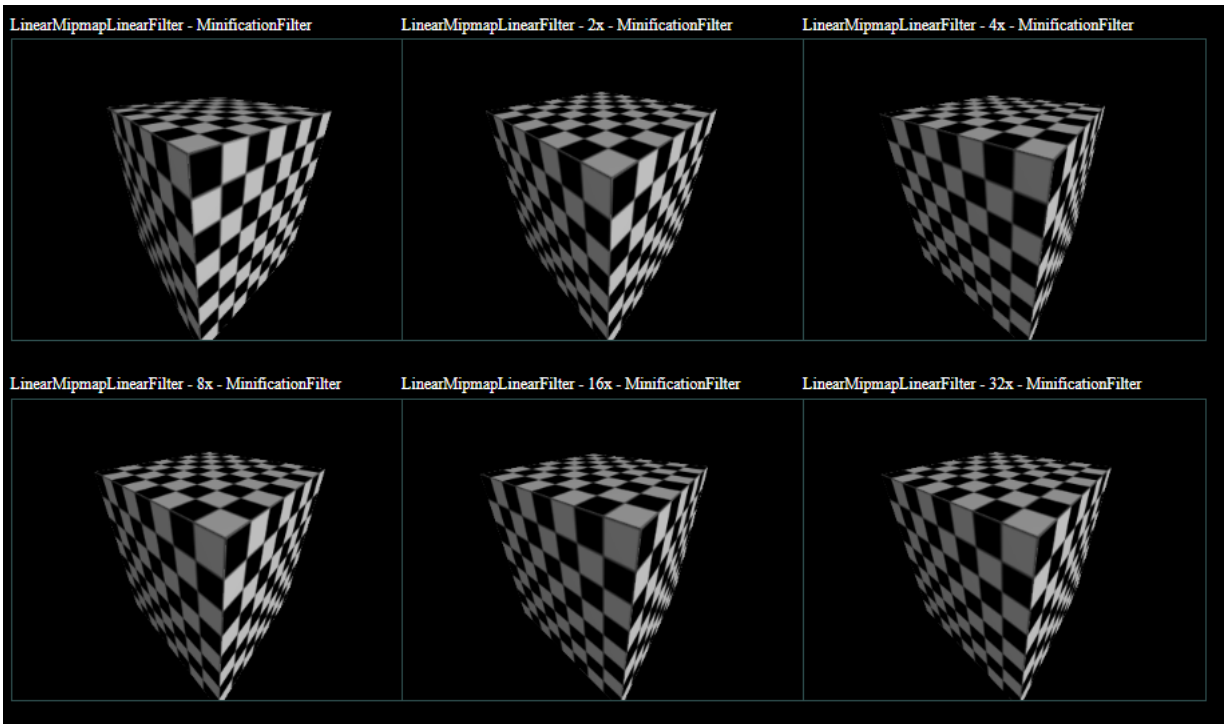


*figura 8. Usando o filtro de minificação de LinearMipmapLinearFilter com variação de anisotropia.*

Ao fim dos teste, observei que algumas variações tanto de filtro quanto de anisotropia são praticamente imperceptíveis, imagino numa aplicação real tornando inviável a mudança pelo o custo de recurso consumido, a tentativa de utilizar 32x não foi bem sucedida obviamente pelo o fato de a figura deitada não se alongar muito, não havendo necessidade para isso, na utilização da 16x e comparação com 8x já não se percebe muita diferença.

## Exemplo de Aplicação Real

Para finalizar nada melhor do que observar como a anisotropia é aplicada em uma aplicação real, optei por testar em jogo por mim jogado(world of warcraft), abaixo podemos ver a aplicação de anisotropía 2x, 4x, 8x e 16x respectivamente.

*figura 9. Anisotropia 2x Aplicada no world of warcraft*

*figura 10. Anisotropia 4x Aplicada no world of warcraft*


*figura 11. Anisotropia 8x Aplicada no world of warcraft*

*figura 12. Anisotropia 16x Aplicada no world of warcraft*

Para melhor visualização das imagens e poder comparar o efeito da anisotropia no jogo(world of Warcraft) eu disponibilizei as imagens no google drive no link no fim do relatório.

## Dificuldade e Melhorias Possíveis

Para esse trabalho não tive dificuldades, foram bem intuitivas as modificações a serem feitas para o desenvolvimento deste trabalho.

Trabalho poderia ser refeito em outra linguagem que o deixasse mais otimizado, poderia ter sido feito o desenvolvimento de funções que permitisse rotacionar o cubo em tempo de execução e poderia ser ampliado o número de filtros utilizando outras bibliotecas, implementando texturas "elaboradas".

## Referências


1. Notas e vídeos de aulas do Prof. Christian Azambuja Pagot.
2. https://threejs.org/docs/index.html#api/en/
3. https://pt.wikipedia.org/wiki/Anisotropia


## Link para o código e Imagens

Linear & Nearests Neighbour - Magnification: https://codepen.io/Reniwtz/pen/OJpowQq
LinearFilter - Minification: https://codepen.io/Reniwtz/pen/gOmZpxP
Nearests Neighbour Filter - Minification: https://codepen.io/Reniwtz/pen/OJprVEb
NearestMipmapNearestFilter - Minification Filters: https://codepen.io/Reniwtz/pen/yLMELjz
NearestMipmapLinearFilter - Minification Filters: https://codepen.io/Reniwtz/pen/QWpVZgx
LinearMipmapNearestFilter - Minification Filters: https://codepen.io/Reniwtz/pen/JjWaaoZ
LinearMipmapLinearFilter - Minification Filters: https://codepen.io/Reniwtz/pen/oNZPQGp
Imagens de Aplicação de Anisotropia no world of warcraft: encurtador.com.br/bpzIK