UNIVERSIDADE FEDERAL DA PARAÍBA

Discente: Carlos Fabrício da Silva Pontes, Matrícula: 20190020637
Discente: Davi José Lucena Luiz, Matrícula: 20190115490
Docente: Christian Azambuja Pagot
Disciplina: Introdução à Computação Gráfica

ATIVIDADE PRÁTICA 4

João Pessoa, 2021

1. Desenvolvimento da atividade

Nesta atividade, realizamos uma análise comparativa entre os filtros de textura disponíveis na biblioteca ThreeJs. Abordamos os casos em que ocorrem magnificação e minificação, também verificando as mudanças que ocorriam na alteração do parâmetro de anisotropia, para os casos de minificação. Fizemos isso por meio de prints que enfatizam os resultados dos testes.

2. Explicação breve sobre as estratégias adotadas para o desenvolvimento da atividade:

Primeiramente analisamos os filtros disponíveis na biblioteca para tratar os casos de magnificação de minificação. Após isso testamos os casos de magnificação onde fizemos uso dos filtros disponíveis na biblioteca, THREE.NearestFilter e THREE.LinearFilter. Depois para verificar os filtros de minificação alteramos a câmera e aumentamos o tamanho da geometria do cubo para forçar a ocorrência da minificação, com isso pudemos testar os filtros THREE.NearestFilter, THREE.LinearFilter, THREE.NearestMipmapNearestFilter, THREE.NearestMipmapLinearFilter, THREE.LinearMipmapNearestFilter, THREE.LinearMipmapLinearFilter.

3. Printscreens dos resultados gerados, dificuldades e possíveis melhorias:

No primeiro caso, testamos os filtros responsáveis por tratar a magnificação, usamos a textura original disponibilizada no código, e comparamos o mag filter Nearest com o Linear.

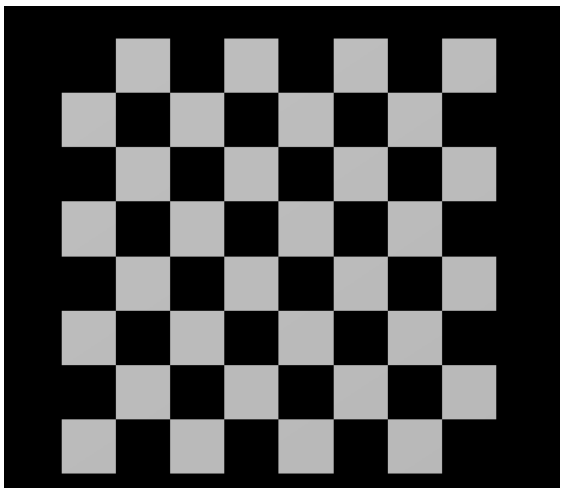

Figura 1: Nearest Filter

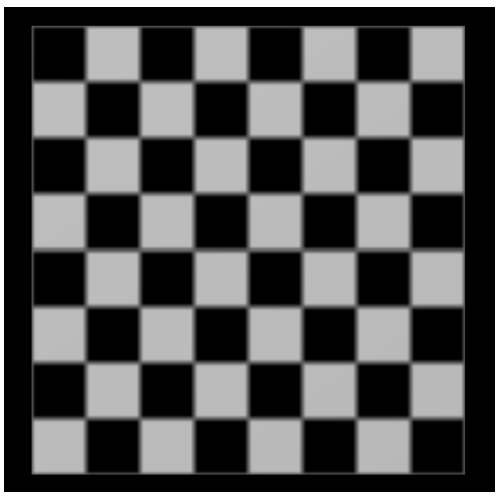

Figura 2: Linear Filter

Na magnificação, um texel da textura é maior que um pixel, então para corrigir isso o Nearest Filter ao renderizar o pixel, pega o valor do texel mais próximo, já o Linear Filter, interpola os 4 texels mais próximos. Por isso, o Nearest oferece uma maior nitidez do que o Linear. Essa diferença pode ser percebida na comparação entres as figuras 1 e 2 acima.

Para verificar quando ocorre uma minificação, mudamos a posição da câmera para z=0.9 e y=0.7, e aumentamos o tamanho do cubo para (1.2,1.2,1.2), para forçar a minificação, que acontece quando um texel é menor que um pixel. As figuras 3 e 4 demonstram a ocorrência da minificação de forma precisa após realizado os testes e algumas mudanças.

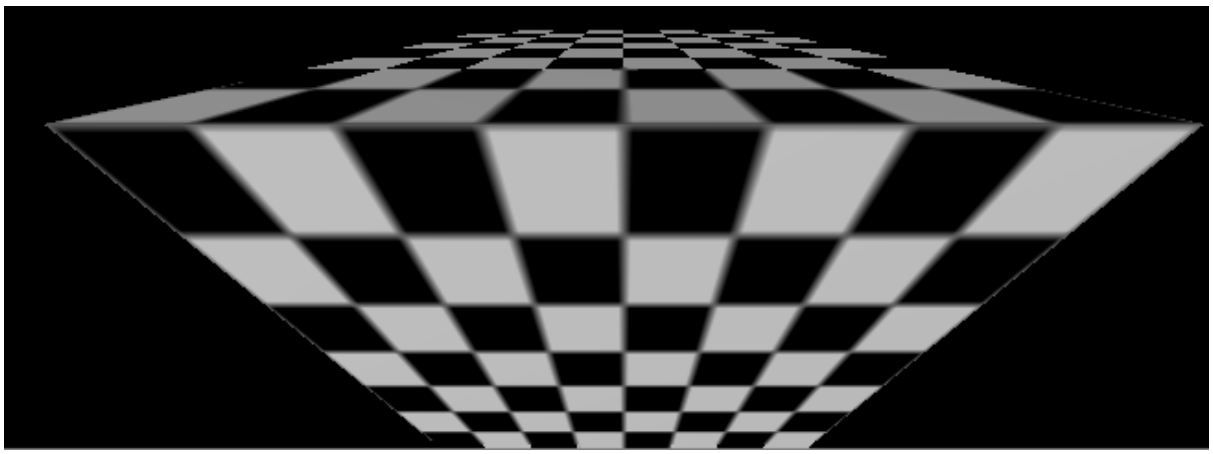

Figura 3: Cubo com o Nearest Filter

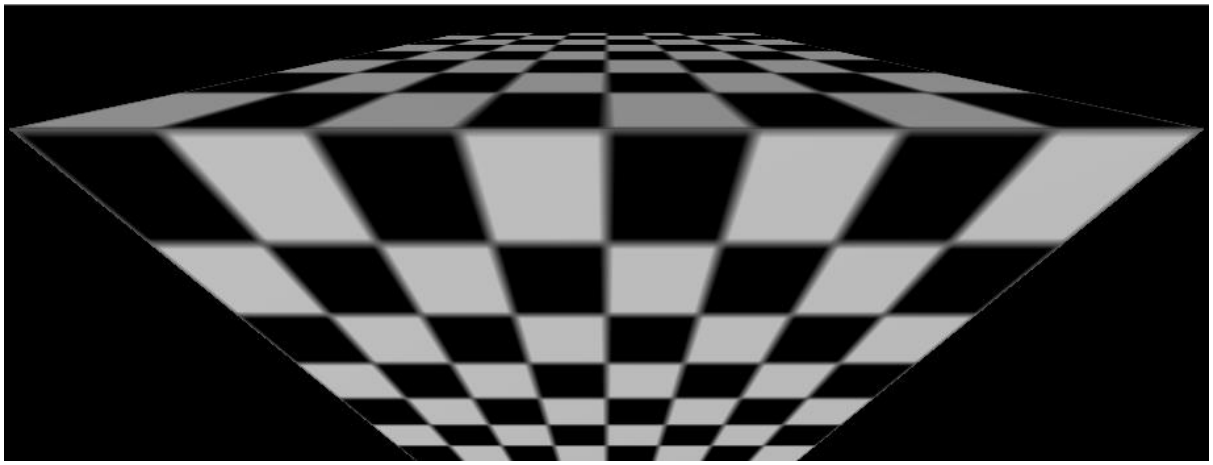

Figura 4: Cubo com o Linear Filter

Primeiramente, para corrigir os casos de minificação, testamos os filtros Nearest (figura 3) e o Linear (figura 4) que funcionam da mesma maneira que na magnificação.

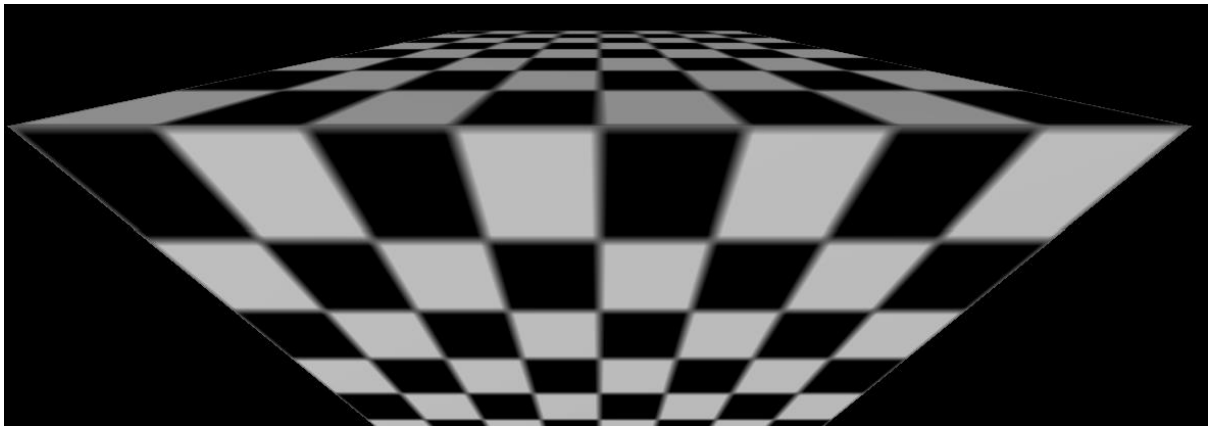

Figura 5: Cubo com o LinearMipmapLinearFilter

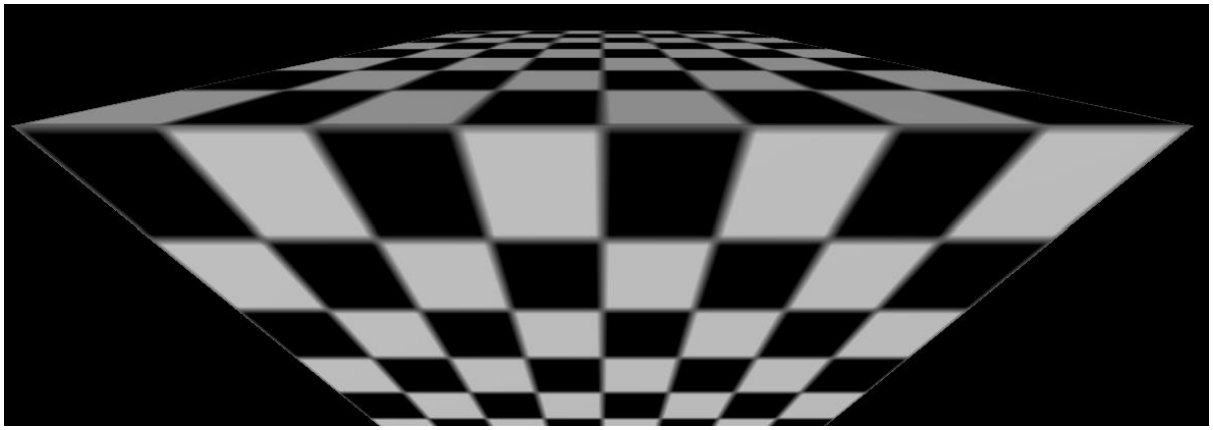

Figura 6: Cubo com o LinearMipmapNearestFilter

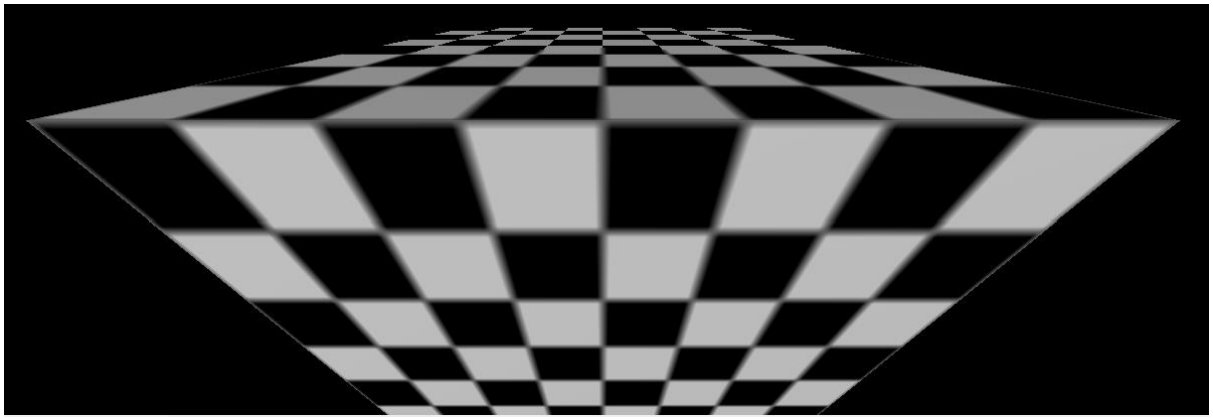

Figura 7: Cubo com o NearestMipmapLinearFilter

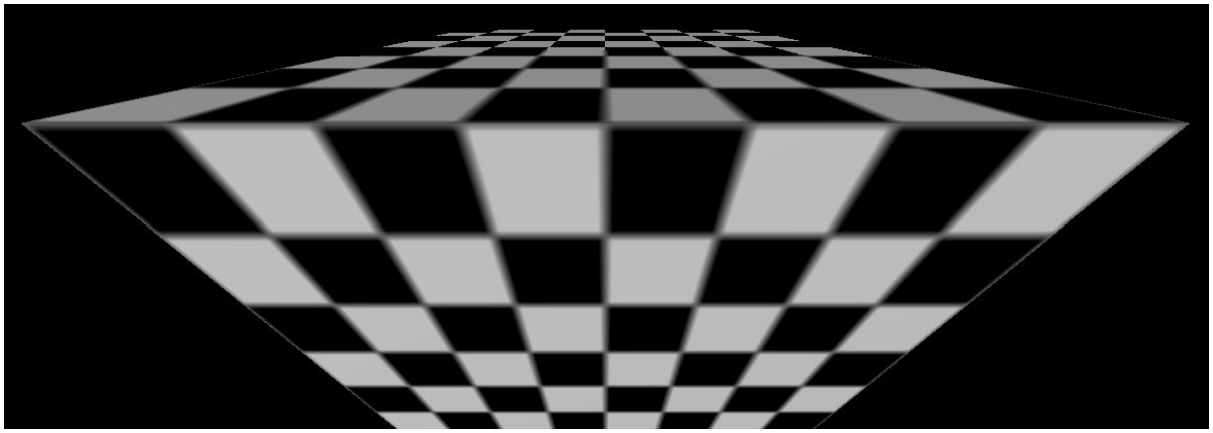

Figura 8: Cubo com o NearestMipmapNearestFilter

O Mipmap tem como objetivo reduzir o ruído da minificação levando 4 texels da textura até 1 pixel da imagem final que será exibida na tela e esse pixel deve receber a média de cada cor do texel que já está com a média de cores pré-calculada. Para comparar, testamos os 4 filtros que usam mipmaps, que são eles o NearestMipmapNearest, que seleciona o mipmap que mais se aproxima do tamanho do pixel, o NearestMipmapLinear, que escolhe os dois mipmaps que mais se aproximam e misturam os dois, o LinearMipmapNearest, que seleciona o mipmap que mais se aproxima do tamanho do pixel, usando o critério do LinearFilter, ou seja, mistura os 4 pixels mais próximos, LinearMipMapLinear, que seleciona dois mipmaps que mais se aproximam do tamanho do pixel, em seguida misturam os dois, causando assim uma mistura de 8 pixels. Levando isso em conta, é de se esperar que os MipMap Nearest sejam mais nítidos e o mipmap LinearLinear seja o menos nítido, devido a mistura dos pixels. Isso é possível perceber ao observar as figuras 5, 6, 7 e 8 acima.

Testes com a anisotropia:

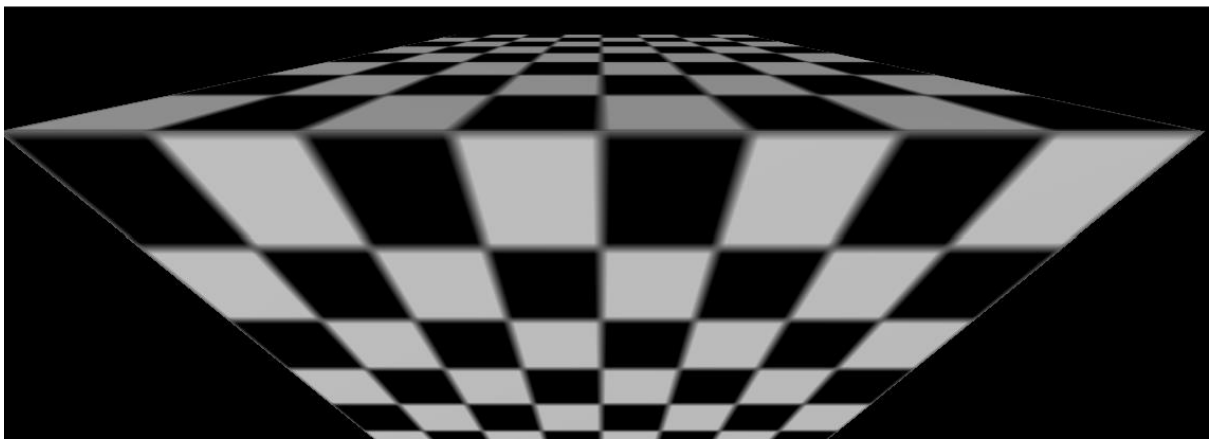

Figura 9: Cubo com o LinearFilter anisotropia 16x

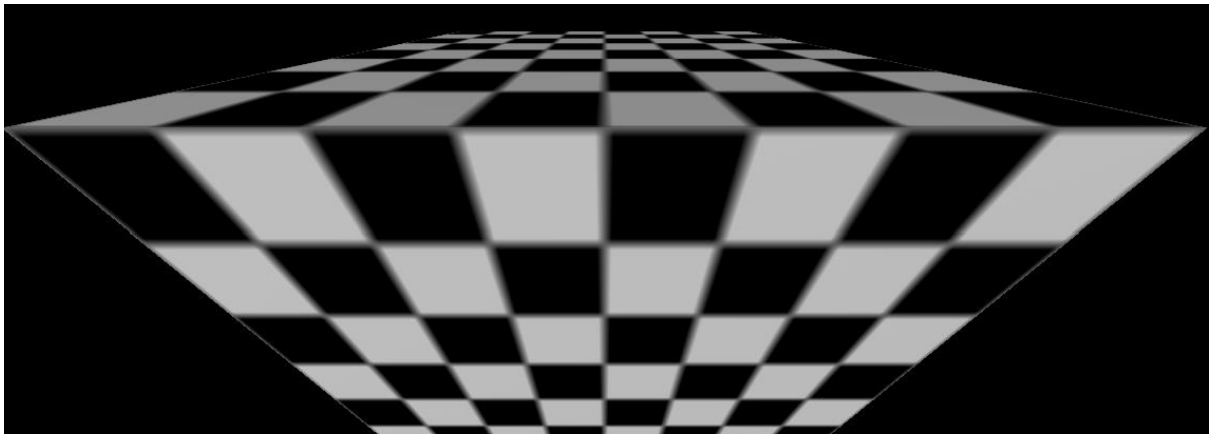

Figura 10: Cubo com o NearestFilter anisotropia 16x

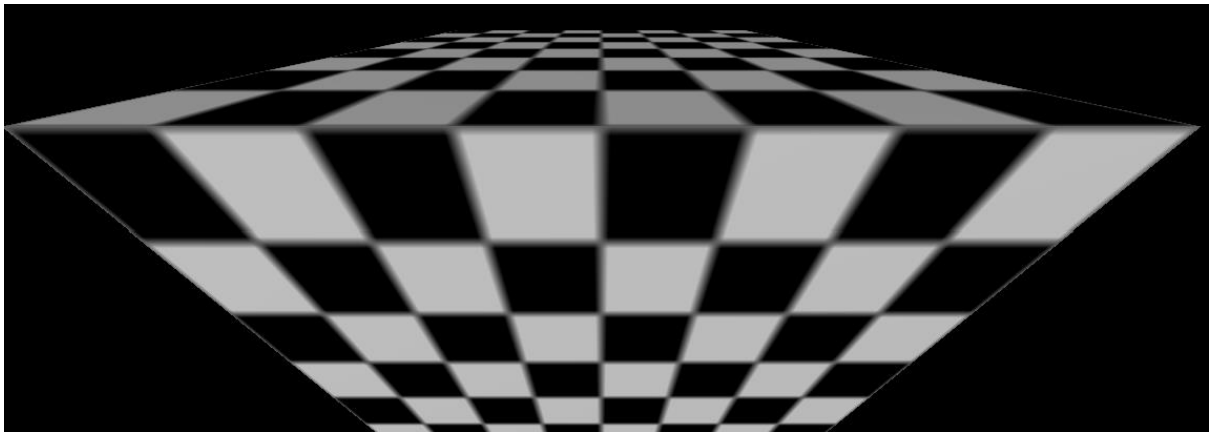

Figura 11: Cubo com o LinearMipmapLinearFilter anisotropia 16x

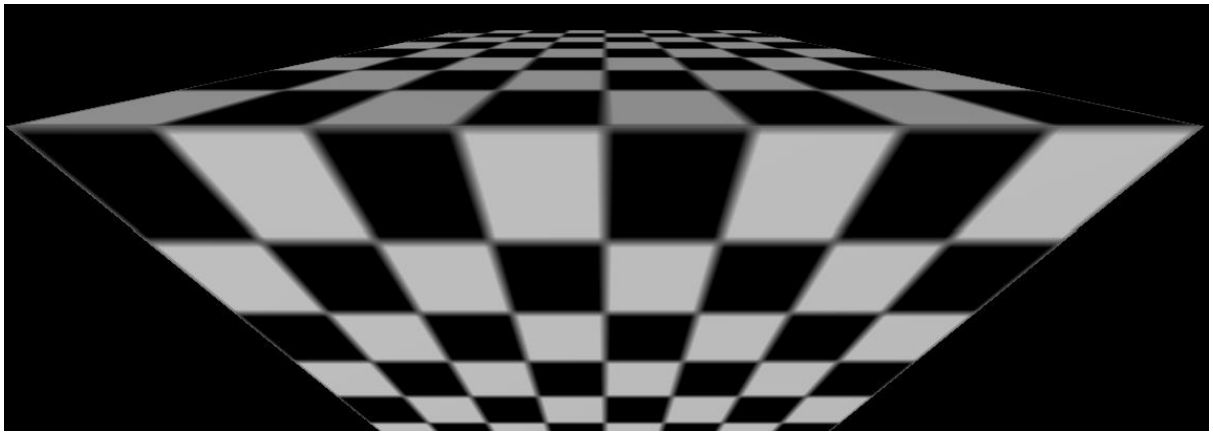

Figura 12: Cubo com o LinearMipmapNearestFilter anisotropia 16x

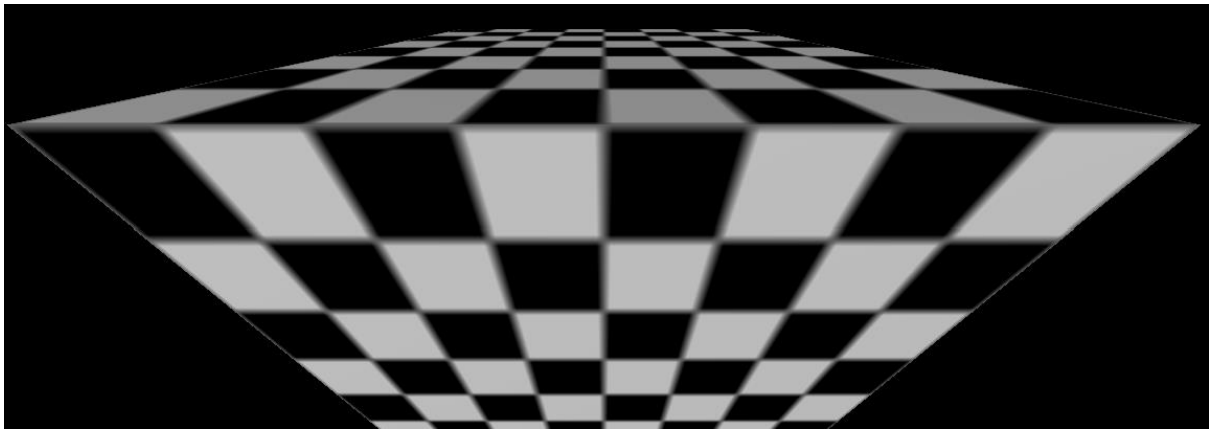

Figura 13: Cubo com o NearestMipmapLinearFilter anisotropia 16x

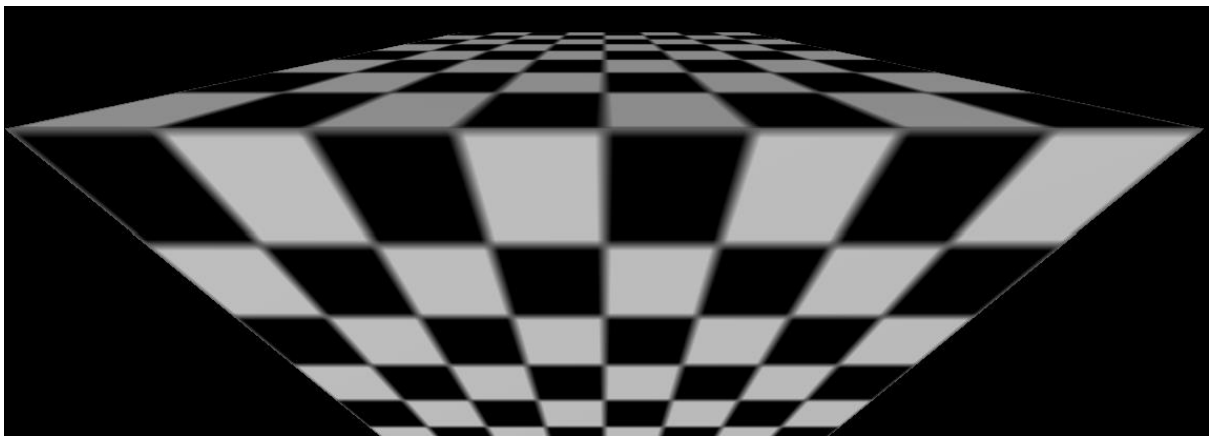

Figura 14: Cubo com o NearestMipmapNearestFilter anisotropia 16x

Como pode se observar nas figuras 9-14, ao fazermos o uso da anisotropia 16x, obteve-se uma maior nitidez do que quando usava apenas os filtros de minificação.

Utilizando a filtragem NearestFilter para realizar a comparação da anisotropia com diferentes valores:

Figura 15: NearestFilter com anisotropia 4x

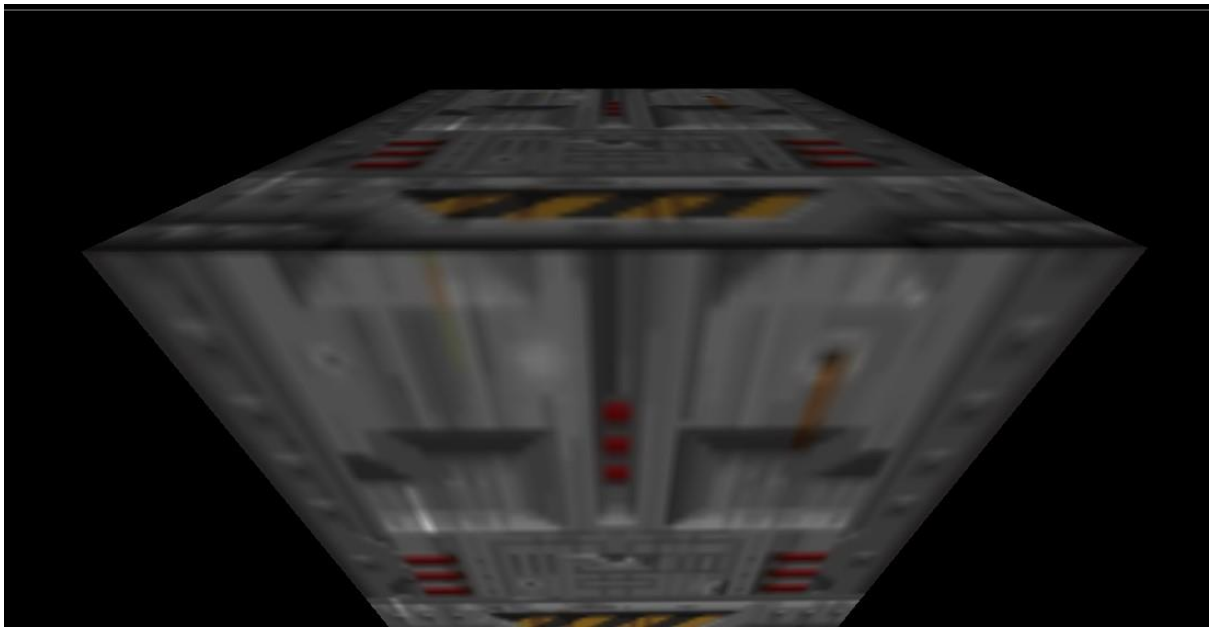

Figura 16: NearestFilter com anisotropia 8x

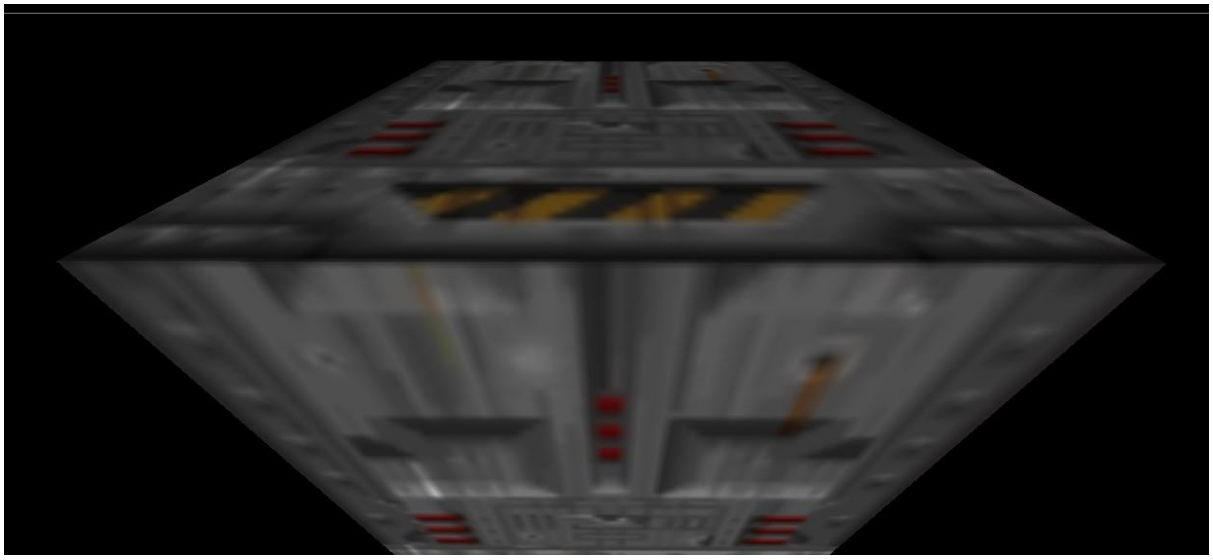

Figura 17: NearestFilter com anisotropia 16x

Partindo para análise da anisotropia utilizamos o NearestFilter para realizar alguns testes e comparações com o cubo.

Primeiramente de acordo com a figura 15 a qual demonstra o cubo com a sua anisotropia configurada em 4, foi capaz de fornecer ao cubo e a imagem uma maior nitidez e diretamente também uma característica mais suave.

Enquanto que na figura 16 realizamos o aumento da configuração da anisotropia para o valor 8, de acordo com a imagem é possível perceber que a suavidade ainda permanece na imagem da mesma forma que permanece na figura 15, com a anisotropia configurada com o valor 4. No entanto, a principal diferença entre as duas figuras está localizada na textura, a qual se demonstra de forma mais clara na figura 16 em comparação com a figura 15.

Já na figura 17 configuramos a anisotropia para o valor 16, e mais uma vez é possível perceber que a suavidade ainda é predominante e a textura permanece nítida e mais clara como citado anteriormente. Mas com as cores dos pixels um pouco mais clara em relação as duas imagens anteriores analisadas.

4. Referências Bibliográficas:

https://threejsfundamentals.org/threejs/lessons/threejs-textures.html
https://threejs.org/docs/#api/en/textures/Texture.anisotropy

5. Link para repositório online CodePen:

https://codepen.io/fabriciocc/pen/jOBJvKR?editors=0010