UFPB

Centro de Informática – João Pessoa
Disciplina: Computação Gráfica
Klismann de Oliveira Barros - 20200085284
Thaís Gabrielly Marques - 20180135293

ATIVIDADE PRÁTICA 4 - MAPEAMENTO DE TEXTURAS

Descrição:

Nesta quarta atividade foi pedido que fizéssemos uma análise comparativa com diferentes filtros de textura do Three.js, utilizando imagens que demonstrassem a diferença.

Estratégias adotadas:

Primeiramente revisamos as aulas postadas pelo professor e o material disponível para começar a sanar as principais dúvidas. Logo após isso, passamos para o template disponível com os primeiros passos, o cubo já estava renderizando duas texturas como visto nos exemplos disponíveis. Durante as nossas pesquisas achamos um codepen que codificava a imagem em base64, otimizando bastante o nosso tempo. Feita a codificação e o teste, optamos por colocar mais uma imagem do jogo Minecraft. Por fim, dividimos em alguns campos de teste para começar as comparações (abaixo está sem /* */, mas no código testamos um de cada vez):

```
//Primeira comparação nearest neighbor fornecido pelo professor.

image.onload = function() {
    texture.needsUpdate = true;
    texture.magFilter = THREE.NearestFilter;
    texture.minFilter = THREE.NearestFilter;
    texture.anisotropy = 1;
    texture.wrapS = THREE.RepeatWrapping;
    texture.wrapT = THREE.RepeatWrapping;
};

//Segunda comparação LinearFilter
```

```
image.onload = function() {
    texture.needsUpdate = true;
    texture.magFilter = THREE.LinearFilter
    texture.minFilter = THREE.LinearFilter
    texture.anisotropy = 1;
    texture.wrapS = THREE.RepeatWrapping;
    texture.wrapT = THREE.RepeatWrapping;
};

//Terceira comparação NearestMipmapNearestFilter
image.onload = function() {
    texture.needsUpdate = true;
    texture.magFilter = THREE.NearestFilter;
    texture.minFilter = THREE.NearestMipmapNearestFilter;
    texture.anisotropy = 1;
    texture.wrapS = THREE.RepeatWrapping;
    texture.wrapT = THREE.RepeatWrapping;
};


//Quarta comparação NearestMipmapLinearFilter

image.onload = function() {
    texture.needsUpdate = true;
    texture.magFilter = THREE.NearestFilter;
    texture.minFilter = THREE.NearestMipmapLinearFilter
    texture.anisotropy = 1;
    texture.wrapS = THREE.RepeatWrapping;
    texture.wrapT = THREE.RepeatWrapping;
};

//Quinta comparação LinearMipmapNearestFilter

image.onload = function() {
    texture.needsUpdate = true;
    texture.magFilter = THREE.LinearFilter;
    texture.minFilter = THREE.LinearMipmapNearestFilter
    texture.anisotropy = 1;
    texture.wrapS = THREE.RepeatWrapping;
    texture.wrapT = THREE.RepeatWrapping;
};
```

```
//Sexta comparação LinearMipmapLinearFilter

image.onload = function() {
    texture.needsUpdate = true;
    texture.magFilter = THREE.LinearFilter;
    texture.minFilter = THREE.LinearMipmapLinearFilter
    texture.anisotropy = 1;
    texture.wrapS = THREE.RepeatWrapping;
    texture.wrapT = THREE.RepeatWrapping;
};

// Sétima comparação filtragem anisotrópica no caso de  minificação

image.onload = function() {
    texture.needsUpdate = true;
    texture.magFilter = THREE.LinearFilter;
    texture.minFilter = THREE.LinearMipmapLinearFilter
     texture.anisotropy = 16; //também testado com 32 e 64(amostras ao
longo do maior eixo, quanto maior mais fica lento e nítido)
    texture.wrapS = THREE.RepeatWrapping;
    texture.wrapT = THREE.RepeatWrapping;
};
```

Como sugestão do professor, apresentaremos essas comparações em pares, na mesma cena, porém, com dois filtros diferentes.

Resultados gerados:

Primeira análise (Propriedades de Magnificação do Three.js):

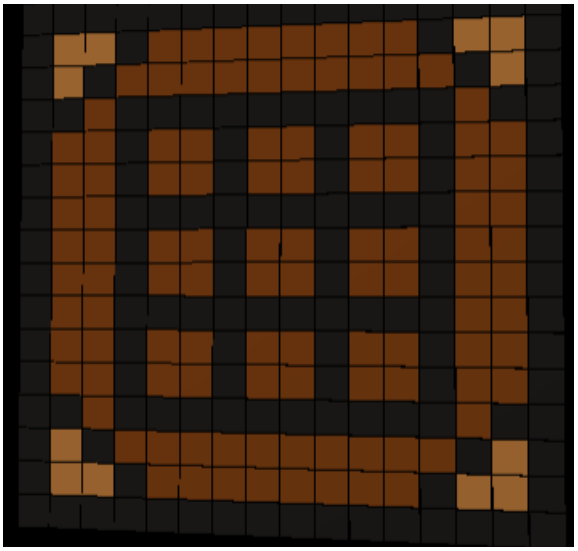

Imagem 01 - Nearest neighbor

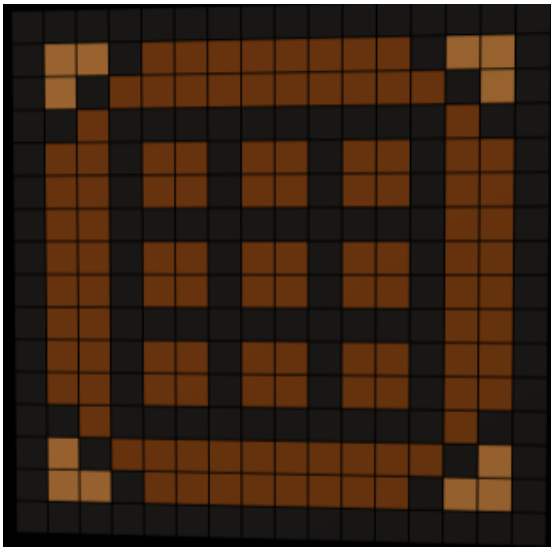

Imagem 02 - LinearFilter

Bom, acima, utilizamos dois filtros de ampliação (também podem ser usados na minificação), o NearestFilter e o LinearFilter. O primeiro, retorna o valor do elemento de textura que está mais próximo das coordenadas especificadas, já o segundo, retorna como padrão a média ponderada dos quatro elementos de textura que estão mais próximos das coordenadas especificadas. A comparação entre Nearest e Linear fica bem nítida na imagem, a Linear pode ser movida para qualquer direção que mantém sua integridade e nitidez, já a Nearest deforma apresentando ruídos, como pode ser visto na imagem 01.

Observação: Além de NearestFilter e LinearFilter, as quatro funções a seguir podem ser usadas para minificação:

Segunda análise (utilizando a textura fornecida pelo professor):

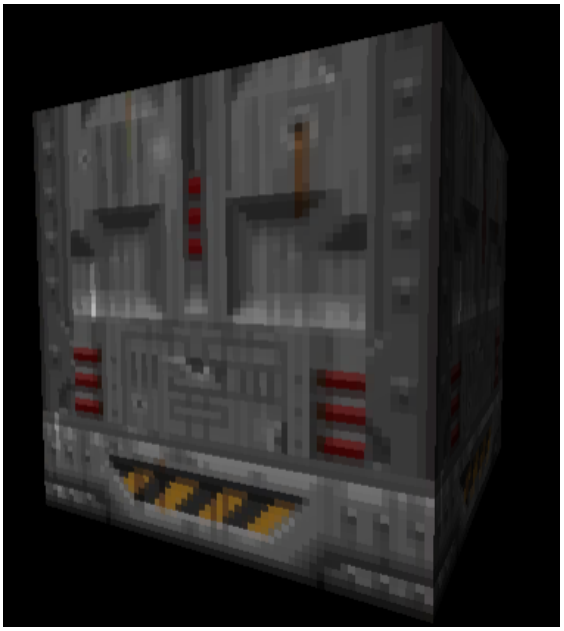

Imagem 03 - NearestMipmapNearestFilter

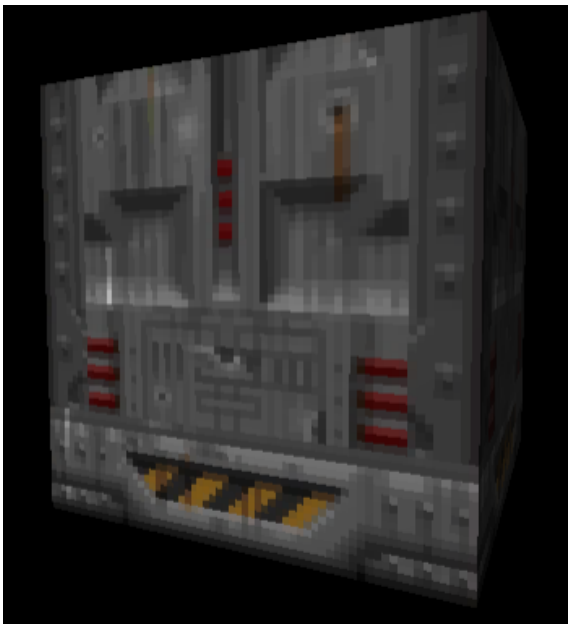

Imagem 04 - NearestMipmapLinearFilter

Acima, utilizamos os filtros NearestMipmapNearestFilter e NearestMipmapLinearFilter (propriedades utilizadas em minificação). O primeiro, escolhe o mipmap que mais se aproxima do tamanho do pixel sendo texturizado e usa o critério do NearestFilter para gerar um valor de textura. Já o NearestMipmapLinearFilter, escolhe os dois mipmaps que mais se aproximam do tamanho do pixel sendo texturizado e usa o critério NearestFilter, o valor final da textura é uma média ponderada desses dois valores. Comparando a imagem dentro

do codepen, tivemos a impressão que o filtro da imagem 04 teve um desempenho melhor de nitidez, mas manteve os problemas de ruído.

Terceira análise(utilizando a textura fornecida pelo professor):

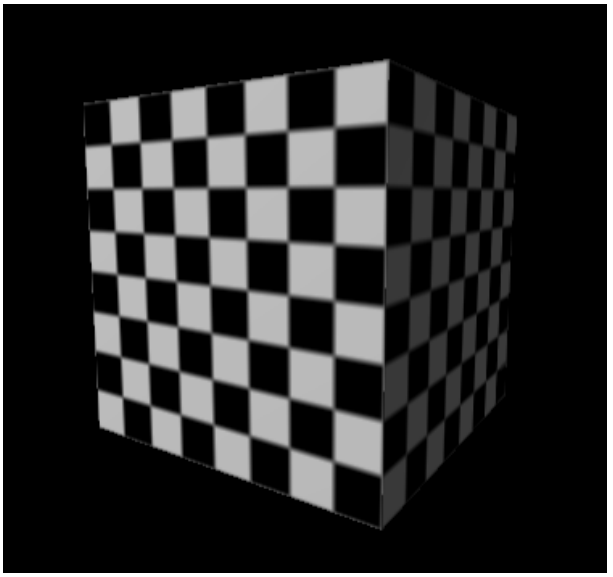

Imagem 05 - LinearMipmapNearestFilter

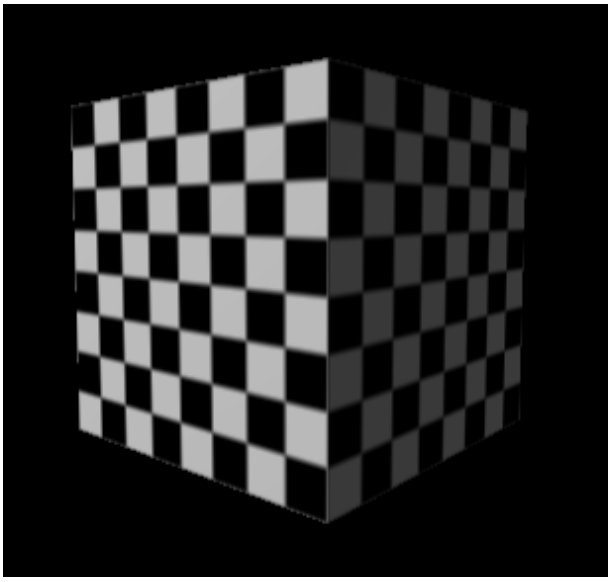

Imagem 06 - LinearMipmapLinearFilter

Acima temos os filtros LinearMipmapNearestFilter e LinearMipmapLinearFilter. O primeiro, 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. Já o segundo, 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. Comparando as imagens é possível perceber um ruído sutil na imagem 05 e existe uma camada de embaçamento, mas a imagem 06 é mais nítida, mesmo apresentando pequenos ruídos.

Quarta análise:

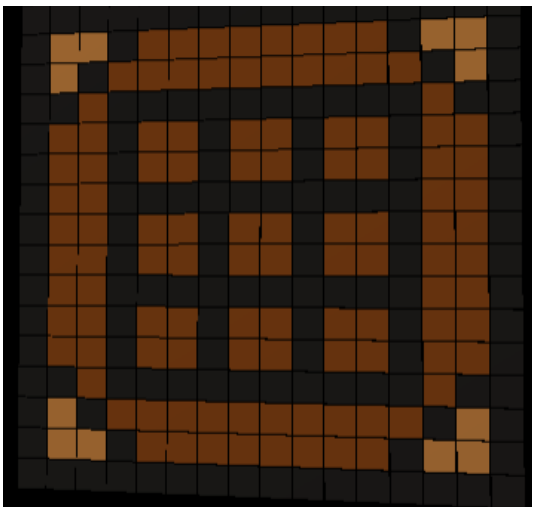

Imagem 07 - Nearest neighbor

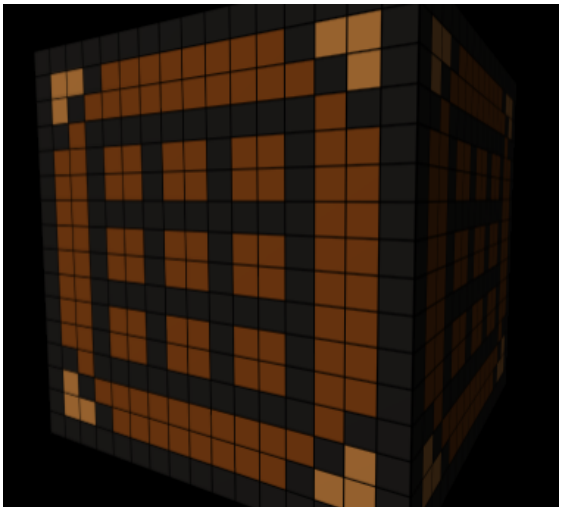

Imagem 08 - Filtragem anisotrópica

Na última comparação, trouxemos o Nearest já explicado acima e a filtragem anisotrópica. Nessa filtragem, o número de amostras obtidas ao longo do eixo através do pixel que possui a maior densidade de texels, utilizamos 64 amostras nessa análise. Por padrão, esse valor é 1. Um valor mais alto fornece um resultado menos borrado que um mipmap básico, ao custo de mais amostras de textura sendo usadas. Logo, a imagem 08 possui uma grande vantagem sobre a imagem 07, por quase não conter ruídos ou embaçamentos, detalhes que podem ser percebidos ao girar o cubo no codepen.

Observação: Em alguns filtros utilizando o critério Linear, existe um maior custo computacional que pode ser percebido devido ao tempo de processamento do Codepen.

Dificuldades e possíveis melhorias:

Tivemos algumas dificuldades em anexar outras imagens para a textura pelo simples fato de erro de sintaxe, mas por serem códigos extensos dificultou-se um pouco a leitura na aba do HTML. Também paramos em algumas dúvidas sobre as filtragens, que logo foram sanadas pelas revisões das aulas e material disponível pelo Three.js. Como possíveis melhorias, poderíamos realizar testes com o MeshPhongMaterial para deixar o objeto mais nítido e luminoso, trazendo melhores resultados ao nosso trabalho. Como nas imagens a seguir:

Imagem 09

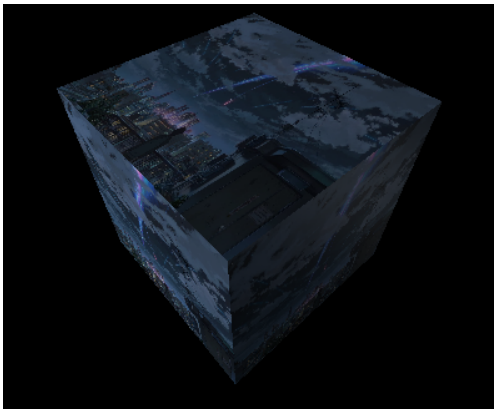

Imagem 10

No futuro, esperamos estender as comparações para outras formas geométricas, dessa forma obtendo novas margens de estudo e de comportamento dos filtros.

Referências bibliográficas:

- Aulas e notas do professor.
- https://codepen.io/monkeyraptor/pen/QbMNPB (CodePen Home Base64 Image Encoder via HTML5 Canvas, usado para gerar os códigos em base64).
- https://threejsfundamentals.org/threejs/lessons/threejs-textures.html
- https://adolfoguimaraes.github.io/threejs/03_aplicandotextura/
- https://threejs.org/docs/#api/en/constants/Textures
- https://threejs.org/docs/#api/en/renderers/WebGLRenderer.getMaxAnisotropy
- https://br.pinterest.com/pin/211106301261160174/ (link da imagem).

Link repositório:

- https://codepen.io/gabriellymarques02/pen/bGqQOdJ