Universidade Federal da Bahia - UFBA
Instituto de Matemática - IM
Departamento de Ciência da Computação - DCC
Curso de Bacharelado em Ciência da Computação

MATA65 - Computação Gráfica Período: 2018.1 Data: 19/06/2017.
Prof. Antonio L. Apolinário Junior
Estagiário Docente: Rafaela Alcantara

Roteiro do Laboratório 7 – Transformações de Intensidade

Objetivos:

- Reforçar os conceitos básicos de imagem digital, transformações de intensidade e filtragem espacial**;**
- Aprofundar a compreensão do uso de shaders e seu funcionamento;
- Entender como o processamento de imagens digitais pode ser feito com a biblioteca *Three.js*.

Conceitos básicos:

Uma imagem pode ser definida como uma matriz bidimensional cujos elementos identificam um ponto (ou *pixel*) na imagem. Tipicamente, cada *pixel* é associado com um valor inteiro, que varia de 0 a 255, ou um valor flutuante, que varia de 0 a 1, correspondente a uma intensidade de cor da imagem. Nesse caso, 0 indica a ausência de cor, enquanto 255 (ou 1, caso seja representação flutuante) indica intensidade máxima de cor. Transformações de intensidade podem ser aplicadas sobre a imagem de forma a modificar a intensidade associada a cada *pixel* da imagem.

Na bibloteca *Three.js,* imagens são exibidas como texturas de uma forma geométrica que permita uma associação 1:1 entre os *pixels* da imagem e os fragmentos gerados pelo *pipeline* gráfico. Para carregar uma imagem em *Three.js,* basta criar uma instância do tipo **`THREE.TextureLoader`** e invocar o método **`load`**, que recebe como parâmetro a localização da imagem no diretório de arquivos e retorna a imagem para armazenamento em uma variável do tipo **`var.`** A forma geométrica para representação da imagem pode ser criada como uma instância do tipo **`THREE.PlaneBufferGeometry`** com dimensão unitária. A imagem pode ser mapeada como textura da forma geométrica a partir do atributo **`map`** do **`THREE.MeshBasicMaterial`**, ou como um *uniform* do tipo “t” do **`THREE.MeshShaderMaterial`**. Por fim, o renderizador deve ser redimensionado para o tamanho da imagem, para que se garanta o mapeamento 1:1 mencionado anteriormente.

Roteiro:

1. Baixe do Moodle os códigos fonte para esse Laboratório. Descompacte no diretório que será visível pelo servidor web.

2. Configure o servidor web[1] e execute o exemplo desse Laboratório.

3. Abra os exemplos deste laboratório e analise as diferentes formas de se exibir imagens com o Three.js.

4. Modifique o fragment Mantendo a imagem anterior, carregue outra imagem no seu código javascript e realize a "soma" das duas imagens no fragment shader.

5. Modifique o fragment shader desenvolvido no item 4 de forma que seja exibida a versão negativa da imagem.

6. Baseado no exemplo Compose.html, crie dois shaders para duas transformações de intensidade distintas (por exemplo, conversão para tons de cinza e binarização) e aplique as sequencialmente sobre a imagem original utilizando dois THREE.ShaderPass.hader do exemplo **`vColorChannel.html`** de forma que a imagem seja exibida em tons de cinza.

7. Modifique o fragment shader desenvolvido no item 4 para que a imagem seja binarizada. Utilize diferentes limiares de binarização (controlado por um slider, por exemplo) e verifique o efeitos deles sobre a imagem resultante.

8. Mantendo a imagem anterior, carregue outra imagem no seu código *javascript* e realize o *blending* das duas imagens no *fragment shader* utilizando a fórmula abaixo (onde f_0 e f_1 correspondem às imagens)

$$g(x) = (1 - \alpha)f_0(x) + \alpha f_1(x)$$

9. **Desafio:** Salt and pepper é um ruído muito comum em imagens, que tem como característica principal a ocorrência esparsa de pixels pretos e brancos na imagem (link). Simule o ruído salt and pepper no fragment shader.

[1] necessário para o ambiente Windows. Utilize o programa USBWebServer (http://www.usbwebserver.net/en/)