**Relatório da primeira atividade**

**CCI-36: Fundamentos de Computação Gráfica**

**Professor Carlos Henrique Quartucci Forster**

**Felipe da Conceição Guimarães**

**Gustavo Nahum Alvarez Ferreira**

**Introdução**

Esta atividade teve por objetivo desenvolver os conceitos iniciais da disciplina de Computação Gráfica, que abordavam a teoria de cores e o sistema visual humano. Os tópicos trabalhados envolveram:

* as ilusões de óptica: este fenômeno é explicado a partir da hipótese de que o olho humano seria otimizado para identificar adequadamente situações frequentes e cotidianas; apesar disso, o olho humano seria susceptível a equívocos, quando na tentativa de interpretar cores ou dimensões que estejam dispostos em cenários enganosos.
* o halftone: consiste no efeito de se representar o conteúdo de uma imagem através de círculos coloridos com um conjunto de cores primárias, em um dado espaço de cor. No caso tomado, recebia-se uma imagem de entrada colorida em escala de cinza e, com base na intensidade da cor de cada pixel, produziu-se um círculo preto com raio correspondente.
* o seletor de cores: a partir de um espaço de cor escolhido, a proposta sugerida era a de encontrar uma forma geométrica adequada à sua distribuição. Então, essa forma geométrica deveria modelar a variação de cores no espaço tomado. Assim que uma das cores apresentadas nesse modelo fosse clicada, uma barra situada abaixo da distribuição de cores deveria adquirir a cor selecionada.

**Exercício 1**

[Guima]

**Exercício 2**

Solicitava-se o código de um programa em Python que lesse uma imagem pequena e gerasse o seu halftone em um arquivo SVG, onde o halftone deveria ser uma matriz de círculos de forma que o tamanho de cada círculo correspondesse à intensidade do pixel da imagem.

Para montar esse código, fez-se uso das bibliotecas Numpy e Pillow. Primeiramente, abriu-se a imagem usando-se o método *Image* do Pillow, e mudou-se o seu formato para o de um Numpy array.

A partir daí, criou-se um novo arquivo SVG, no qual seriam inseridos os círculos pretos de raio adequado. Para isso, montaram-se dois loops aninhados, cujos índices percorriam as duas dimensões da imagem; para cada pixel encontrado, reconhecia-se a sua intensidade: para pixels claros, a intensidade era elevada, logo o tamanho do círculo preto deveria ser menor; para pixels escuros, a intensidade era baixa, então o tamanho do círculo preto deveria ser maior. Admitiu-se que essa tendência poderia ser estabelecida determinando-se o raio do círculo preto de forma linearmente decrescente com a intensidade do pixel.

Testou-se o algoritmo com algumas imagens, e verificou-se uma semelhança visual muito grande entre as imagens originais e as imagens que foram criadas a partir dos círculos. Com efeito, uma vez que a quantidade de círculos foi igual à quantidade de pixels, nem sequer o efeito “artístico” produzido pelos círculos pôde ser notado, uma vez que eles eram pequenos demais para isso.

**Exercício 3**

Para este exercício, solicitava-se a montagem de um seletor de cores em SVG e o atributo onclick de cada objeto; dever-se-ia escolher um espaço de cor e uma forma geométrica correspondente apropriada.

Neste código, usou-se a biblioteca Colorsys.

Inicialmente, selecionou-se o sistema de cores YIQ, cuja distribuição pode ser descrita geometricamente a partir de um paralelepípedo. Essa forma geométrica foi escolhida por conta de sua possível representação a partir de um quadrado (a partir de uma fixação do valor de uma das dimensões do sistema, no caso, fixou-se Y = 0.5). A biblioteca Colorsys permitiu facilitar o trabalho de converter os valores desse sistema YIQ para o sistema RGB que, por sua vez, foi usado para colorir cada “círculo selecionável” no seletor.

A partir de então, precisou-se descrever a sintaxe de geração de um arquivo .svg a partir do Python. Aqui, apenas a função nativa “write” já bastou para resolver essa geração de arquivo.