Universidade do Minho
Escola de Engenharia

Non-Photorealistic Rendering

Practical Assignment of Lighting and Visualization I

Practical Assignment submitted by:

Diogo Nogueira
A78957

Pedro Santos
PG42847

Valeriy Apostolyuk
PG42647

Mestrado (Integrado) em Eng. Informática 2020/2021

Conteúdo

Técnica do *Non-Photorealistic Rendering*

Estudar a técnica do *Non-Photorealistic Rendering* é um passo basilar para perceber quais os possíveis métodos/categorias que abrange e de que modo podem ser reproduzidos sob a forma de *shaders* nos diversos modelos.

A *Non-Photorealistic Rendering* é definida como uma técnica para a criação de imagens que não aspiram o Realismo, combinando a Computação Gráfica com técnicas artísticas. Apesar desta ser uma definição muito abstrata, a verdade é que esta técnica é muito subjetiva, dado que não existe uma forma errada de renderizar toda a ideia do *Non-Photorealistic*.

A ideia deste trabalho prático passa por aplicar três dos métodos do *Non-Photorealistic Rendering* e, para cada um deles, tentar desenvolver um conjunto de *shaders* que permitem verificar e validar os diferentes resultados obtidos:

Método NPR	*Shader* 1	*Shader* 2
Cartoon Rendering	*Dot Shader*	*Toon Shader*
Drawing/Painting Rendering	*Stippling Shader*	*Painterly Shader*
Technical Illustration Rendering	*Gooch Shading Shader*	

Tabela 1 Métodos/Técnicas do *Non-Photorealistic Rendering* Abordados

Métodos do *Non-Photorealistic Rendering*

Método 1 – *Cartoon Rendering*

Descrição Método

Este método caracteriza-se, em particular, pelo uso de efeitos de sombra e bordas que se assemelham ao efeito reproduzido pelos tão conhecidos *Cartoons*. Apesar de existir já uma evolução muito grande na área contrária a esta (*Photorealistic Rendering)*, esta subárea do *Cartoon Redering* é, e continua a ser uma opção mais do que válida no que toca a determinados tipos de figuras presentes na Computação Gráfica.

Basta pensar num caso muito concreto, em que se pretende obter uma imagem com contornos e sombras mais salientes/evidentes. O *Cartoon Redering* é uma opção mais do que válida e permite obter estes efeitos.

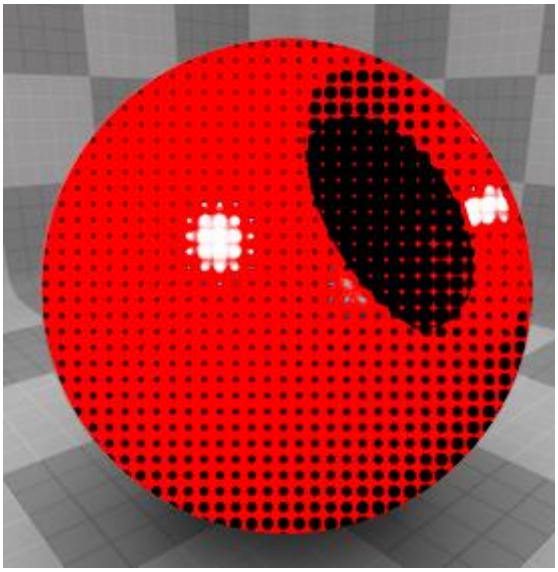

Figura 1 Efeito do *Dot Shade* (*Halftone Material)*

Figura 2 Efeito do *Toon Shader* (*Creating a Toon Shader)*

Implementação dos *Shaders*

1. Dot/Halftone Shader

O *Halftone Shader* é também denominado de *Dot Shader*, uma vez que consiste numa *grid* de pontos redondos, que criam assim um padrão através de um cálculo que define a cor do fragmento.

Para o algoritmo em causa, existiu o delineamento dos seguintes passos, que foram depois convertidos sobre a forma de código:

- Garantir a Rotação da Posição da *Grid* - Tendo em conta que o Sistema Visual Humano é ajustado para discernir linhas horizontais e verticais mais claramente do que quaisquer outros ângulos, uma prática comum passa por fazer um *rotate* da *Grid* em cerca de 45º.

- Criação do Padrão da *Grid* e Filtro *Anti Aliasing* - A ideia consistiu em desenvolver um conjunto de *Dots* partindo de um determinado *radius* para os vários Dots, com a aplicação de um Filtro *Anti Aliasing* por recorrência à função smoothstep() para realizar esta operação de uma forma correta e também universal.

  Este contorno do problema de *Aliasing* é algo muito comum e que deve ser feito, evitando que os *dots* possuam bordas demasiado nítidas.

- Garantir que os *Dots* variam de Tamanho - Este é um passo muito importante para este *Shader*, dado que cria um efeito mais ou menos concentrado de *Dots* considerando a quantidade de luz refletida nesse mesmo local.

Por isso que se calcula o valor da Componente Difusa e se aplica esse mesmo valor de modo a influenciar o tamanho do raio do *Dot*.

2. Toon Shader

Este *Shader* é um estilo de *rendering* muito usado na área dos jogos e que consiste na alteração de todo o modelo de iluminação empregue no desenho em modo 3D. A ideia base está na substituição do sombreamento do modelo 3D para cores sólidas, criando um desenho em modo *flat* onde todas essas sombras e realces surgem como "pedaços" de cor ao invés de serem unidos pelo *smoothing* habitual que torna impercetível a diferença/quebra das várias cores da Imagem/Modelo.

Para o algoritmo em causa, foram acionados 4 tipos de *Level Toons* que permitem verificar o efeito de suavização progressivamente a aumentar:

- Calcular Valor Componente da Luz Difusa Para cada *Toon Level*, a ideia consiste em aplicar uma *Color Out* díspar para os vários valores da percentagem de Componente Difusa que um determinado local recebe/possui.

Para este *Shader*, escolheu-se a cor azul para representar as diferentes variações de sombras e realces. Dessa forma, a quantidade de azul a usar para cada área da imagem, depende, como seria de esperar, da percentagem de Componente Difusa empregue nesse mesmo sítio.

Análise dos Resultados

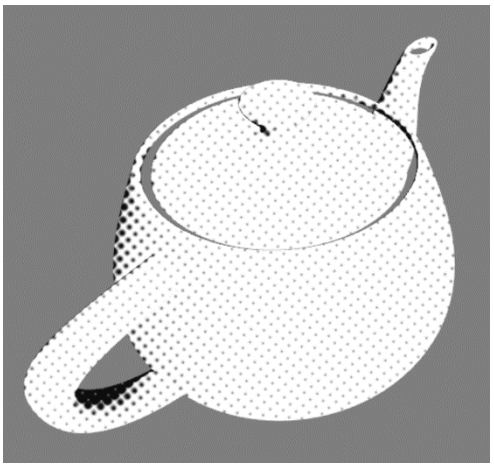

Figura 3 Efeito do *Dot Shader* num *Teapot*

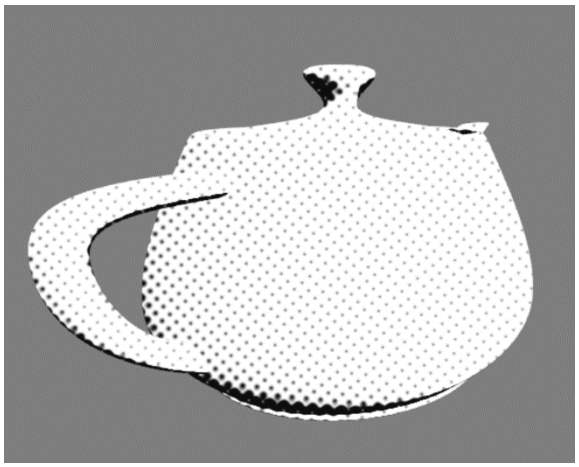

Figura 4 Efeito do *Dot Shader* num *Teapot*

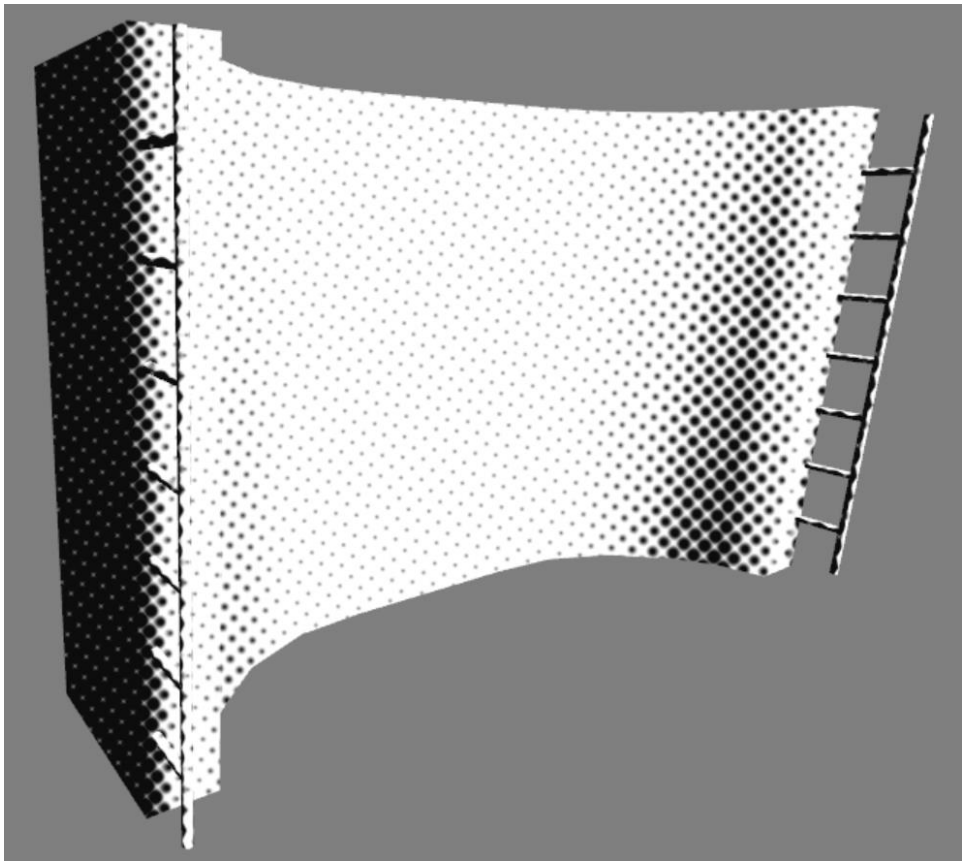

Figura 5 Efeito do *Dot Shader* numa *Skate Ramp*

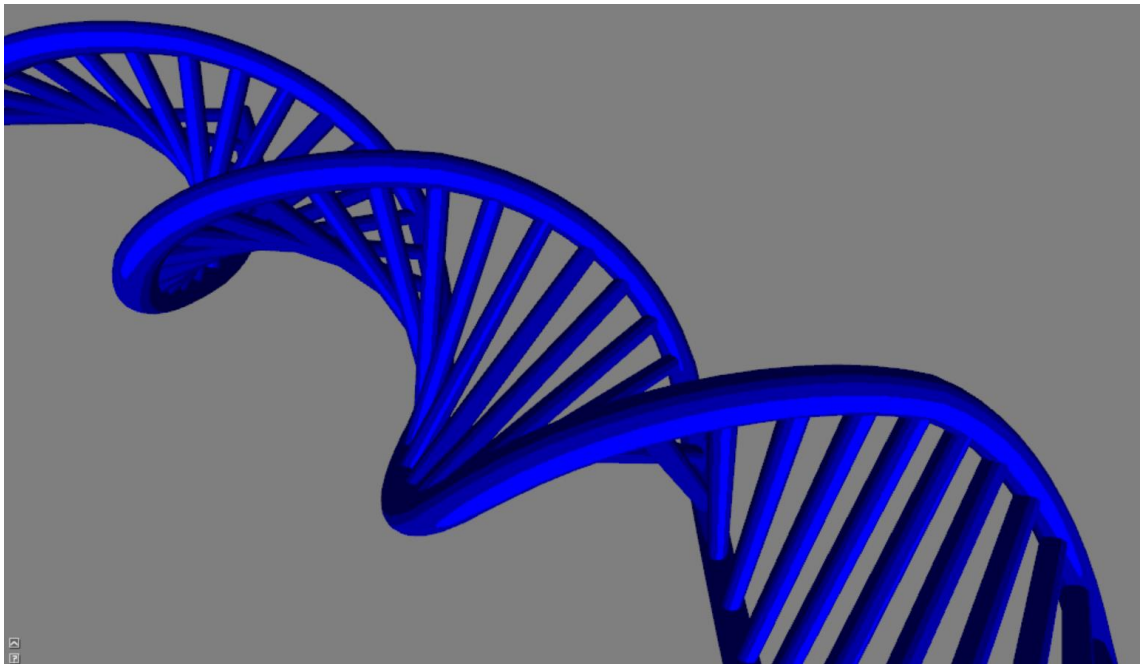

Figura 6 Efeito do *Toon Shader* numa amostra de DNA

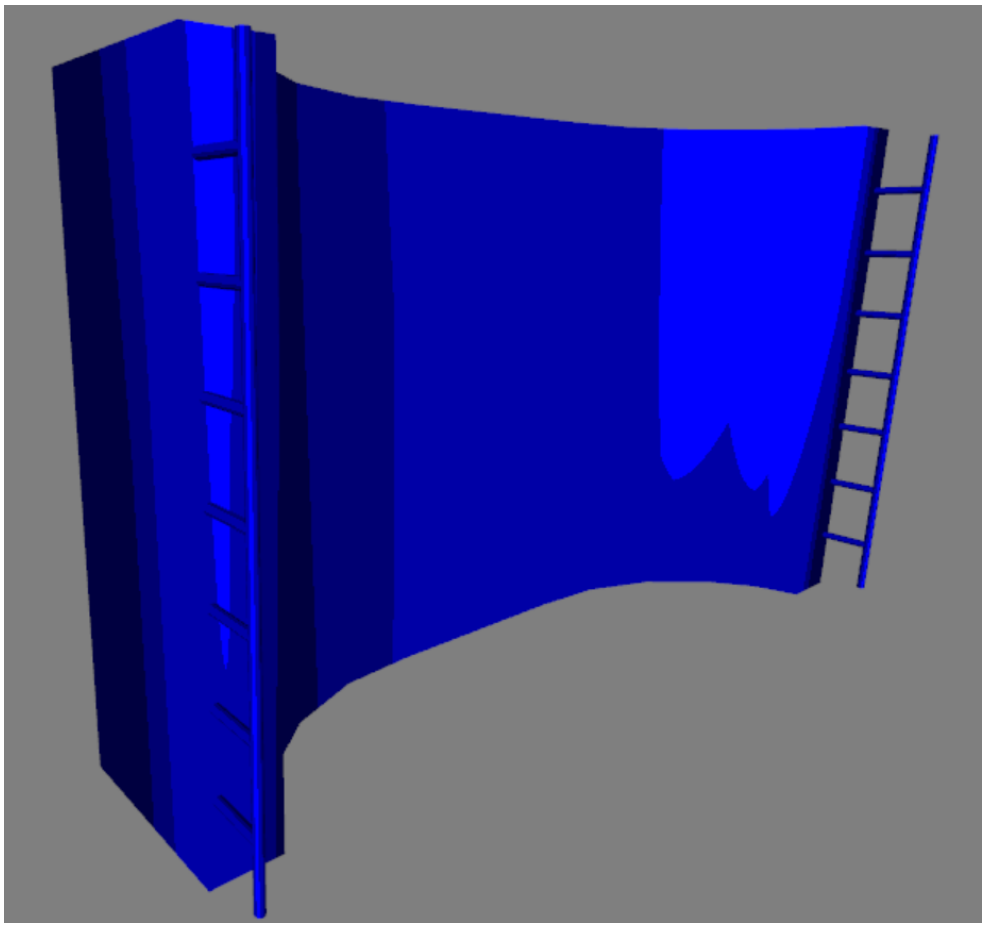

Figura 7 Efeito do *Toon Shader* numa *Skate Ramp*

Método 2 – *Drawing/Painting Rendering*

Descrição Método

Este método caracteriza-se em criar *renderings* que tenham a aparência de serem pintados à mão ou de serem cridos a partir de um padrão simulando vários graus de solidez. Este tipo de processo é muito utilizado nas indústrias de filmes de animação e de uma forma mostra as pessoas como um artista interpreta a sua arte. Em relação a esta subárea, como foi dito, é muito utilizada na Computação Gráfica, quer seja em filmes de animação ou em vídeo jogos.

Figura 8 Efeito do *Painterly Shader* (Aaron Hertzmann, 1998)

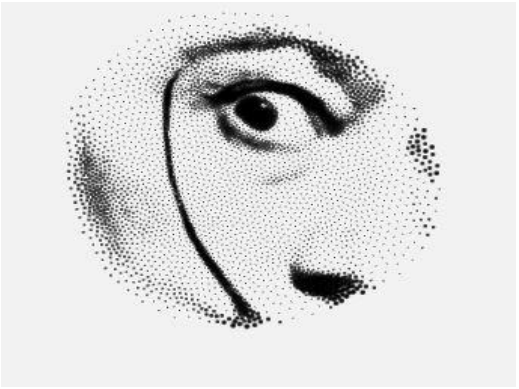

Figura 9 Efeito do *Stippling Shader* (WebGL Shaders)

Implementação dos *Shaders*

1. Stippling Shader

A ideia principal deste *Shader* é criar uma imagem ao utilizar sombreamento com pequenos pontos. Quando os artistas implementam esta técnica nos desenhos ou pinturas, os pontos são feitos de pigmento

de uma única cor, aplicados com uma caneta ou pincel: quanto mais densos forem os pontos, mais escura será o tom; ou mais clara, se o pigmento for mais claro que a superfície.

Nos *Shader*s, a função principal vai utilizar um detetor de bordas (*Edge Detector*) para ver se as cores dos pontos ficam escuros ou se mantém as cores originais, e se for igual a 0 a cor final será a soma da Luz Ambiente, Difusa e a Iluminação Especular; caso contrário, a cor final será a multiplicação entre o fator da escala e a luz ambiente.

Ao analisar o modelo *Teapot*, o utilizador pode observar os resultados obtidos.

2. Painterly Shader

Como foi dito anteriormente, a ideia principal deste *Shader* é criar um modelo com uma textura e utilizar "pinceis" para dar várias camadas ao modelo. Neste exemplo, nós iremos implementar o estilo pintura a óleo.

Nos *Shaders,* a função principal irá ler as cores existentes, os níveis da pintura e as cores da textura implementada, e em cada ciclo da leitura do nível da pintura, dependendo da opção de textura, o programa irá retornar o valor da cor da textura para as coordenadas fornecidas.

Ao analisar o modelo, o utilizador pode selecionar nas definições os diferentes tipos de modelo (*Canvas*, *Teapot*, *Sphere*), escolher 4 tipos de textura, e selecionar um número inteiro para o nível da pintura.

Análise dos Resultados

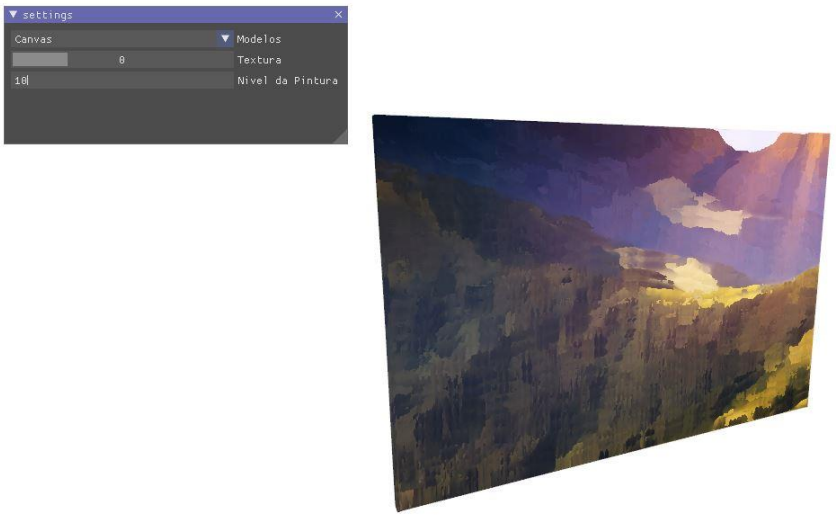

Figura 10 Canvas com *Painterly Shader*

Figura 11 *Teapot* com *Painterly Shader*

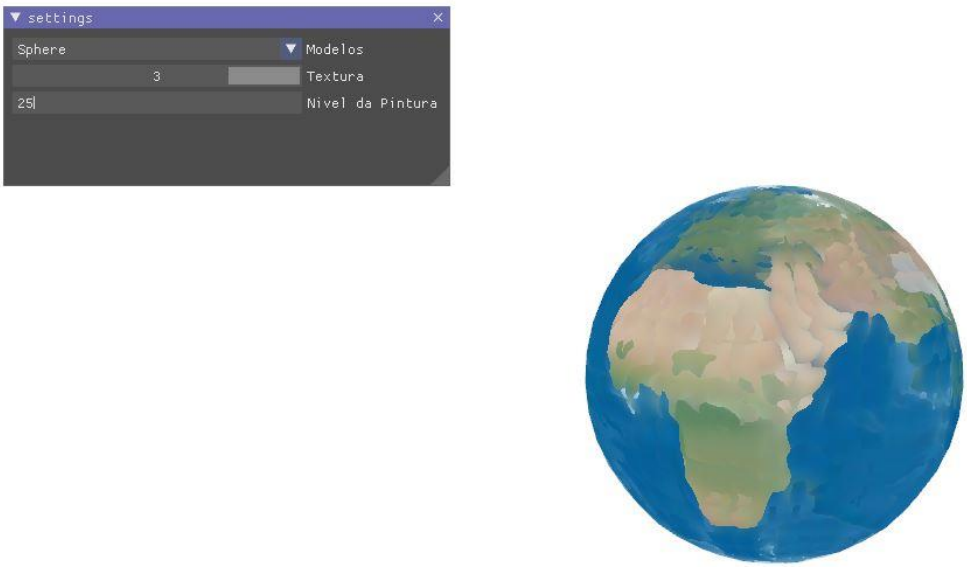

Figura 12 *Sphere* com *Painterly Shader*

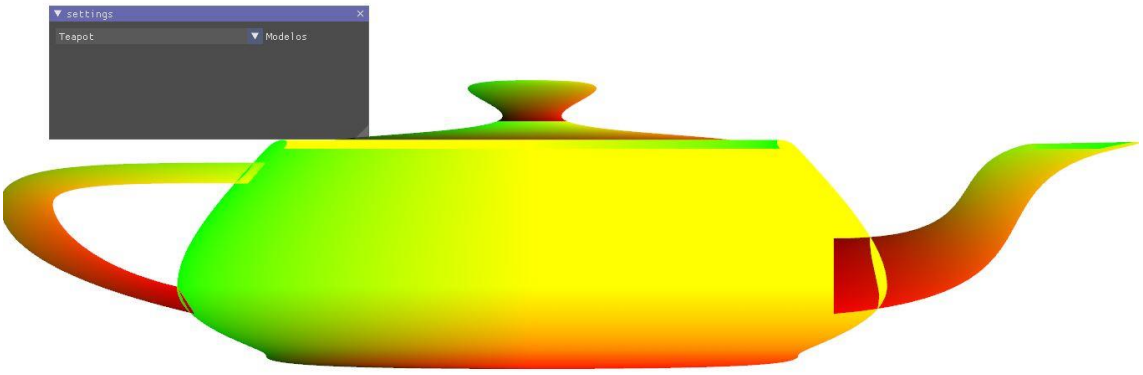

Figura 13 *Teapot* com *Stippling Shader**

*NOTA: Os resultados da secção *Stippling* não foram obtidos devido a erros na programação.

Método 3 – *Technical Illustration Rendering*

Descrição Método

Este método de ilustração difere da Computação Gráfica convencional, na qual uma enorme quantidade de detalhes pode ser apresentada numa imagem para torná-la mais realista. O objetivo de criar uma Ilustração Técnica é transmitir informações da estrutura de um objeto da maneira mais eficiente e detalhada possível. Com isto, os detalhes que não contribuem para a compreensão do objeto são omitidos e os detalhes cruciais são claros e diretos. A Ilustração Técnica normalmente segue uma série de características:

- As linhas de borda são desenhadas com curvas pretas;
- Os objetos foscos são sombreados com intensidades distantes do preto ou do branco, com calor ou frieza de cor indicando a curvatura da superfície;
- Uma única fonte de luz fornece destaque branco;
- Sombras raramente são incluídas, mas se forem usadas, são colocadas onde não obstruem detalhes ou características importantes;
- Objetos metálicos são sombreados como se fossem muito anisotrópicos.

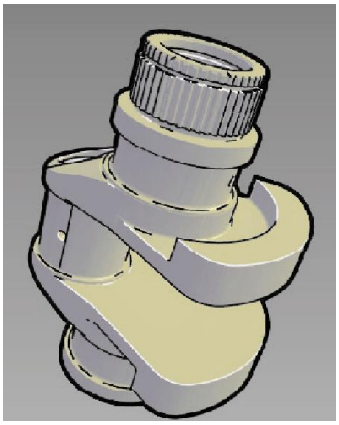

Figura 14 Exemplo de *Technical Illustration* (Non-Photorealistic Rendering - Scientific, s.d.)

Implementação do *Shader*

1. *Gooch Shading Shader*

O *Gooch Shading* define duas cores adicionais em conjunto com a cor do modelo original: uma cor quente (como amarelo) e uma cor fria (como azul). A cor quente indica as superfícies viradas para a fonte de luz, enquanto a cor fria indica as superfícies voltadas para o lado oposto. Isto permite que o sombreamento ocorra apenas em tons médios, para que as linhas de borda e os realces permaneçam visualmente proeminentes.

O *Gooch Shading* é normalmente implementado em duas partes: inicialmente, todos os objetos na cena são desenhados com o sombreamento "frio para quente"; de seguida, faz-se o *rendering* das bordas do objeto a preto, destacando assim os contornos.

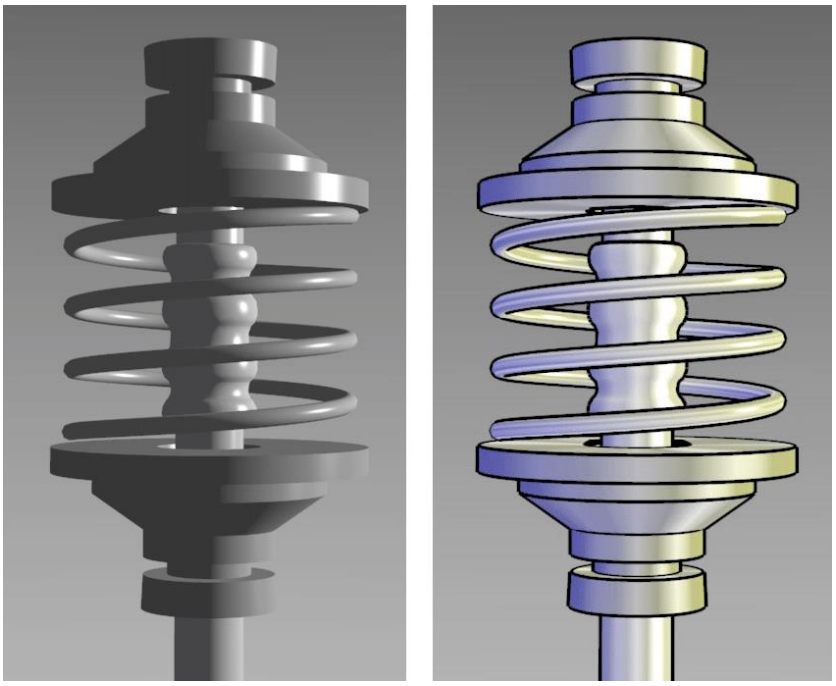

Figura 15 Modelo *rendering* com sombreamento de *Phong* (Esquerda) modelo *rendering* com sombreamento de *Gooch* (Direita) (Cmolik, 2011)

Análise dos Resultados

Como podemos observar na Figura 16, foi possível renderizar um modelo onde as superfícies viradas para a fonte de luz são desenhadas com tons de cor quente e as superfícies voltadas para o lado oposto são representadas em tons de azul, cor fria. Além disso os contornos estão realçados com o preto.

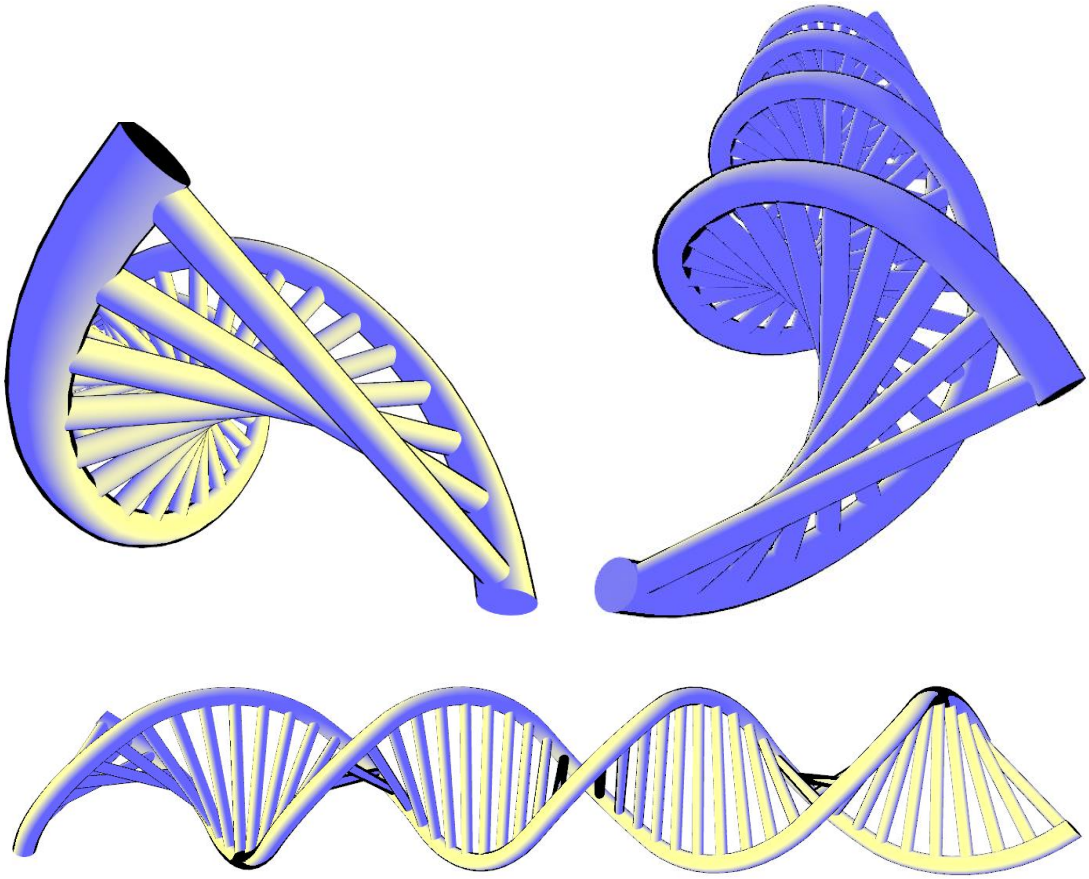

Figura 16 Modelo *rendering* com sombreamento *Gooch*, vista de frente (Esquerda), vista de trás (Direita) e vista de lado (Baixo).

Referências

Cmolik, L. (11 de 2011). Interactive Illustrative Rendering of 3D Meshes.

Non-Photorealistic Rendering - Scientific. (s.d.). Obtido de ResearchGate: https://www.researchgate.net/figure/Technical-illustration-showing-how-interior-lines-are-drawn-in-white-to-show-highlights_fig4_236973460

Aaron Hertzmann - Painterly Rendering with Curved Brush Strokes of Multiple Sizes. Obtido de https://mrl.cs.nyu.edu/publications/painterly98/

WebGL Shaders. Obtido de https://webgl-shaders.com/stippling-example.html

Halftone Material. Obtido de https://learn.foundry.com/modo/902/content/help/pages/shading_lighting/shader_items/halftone.html

Creating a Toon Shader. Obtido de http://rbwhitaker.wikidot.com/toon-shader