Universidade Federal de Santa Maria
Departamento de Computação Aplicada
Prof. Cesar Tadeu Pozzer
Disciplina: Computação Gráfica
pozzer@inf.ufsm.br
18/01/2024

Síntese de Imagens e Realismo

1 Sinais e Imagens Digitais

O mundo real é composto por sinais **contínuos**. Pela própria natureza do computador, este tipo de sinal não se mostra eficiente, surgindo a necessidade de ser **convertido** em digital. Esta solução exige cuidados especiais para que a perda de informação, quase inevitável, seja bem contornada, de forma a reduzir o efeito conhecido como ***alias***.

A síntese de imagens digitais usa vários conceitos de Processamento Digital de Sinais. Uma imagem digital, que é o resultado de um processo computacional, é uma representação digital **discreta** de sinais ópticos contínuos, ou o resultado da amostragem de modelos geométricos que são matematicamente contínuos [EBER 98]. Informações suplementares e representações formais a este material podem ser encontradas em [GLAS 95, OPPE 89].

O QUE É UM SINAL?

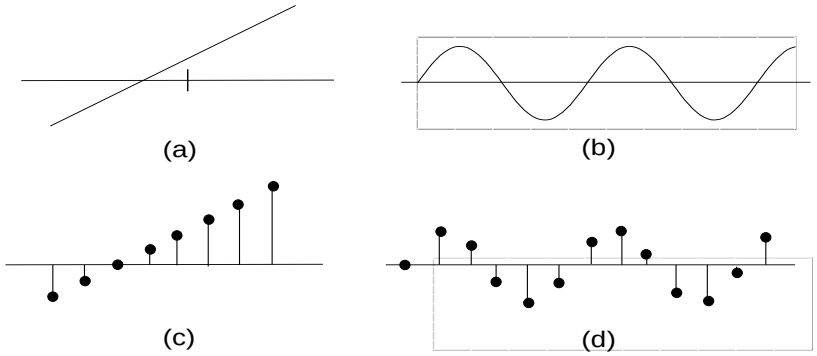
Sinal pode ser definido como uma **função** que transporta informação, geralmente sobre o estado ou comportamento de um sistema físico. Matematicamente, é definido como uma função de uma ou mais variáveis independentes, que possui valores dentro de algum padrão de variação [OPPE 89]. Imagens digitais são exemplos de funções com duas variáveis independentes.

Em relação à forma como podem ser representados, pode-se dividi-los em dois grupos:

1) **Sinais Contínuos:** Também chamados sinais analógicos, encontram-se presentes na natureza na forma de luz, som, etc. Não possuem uma aplicação prática em computação, pois como sua representação é dada por

infinitas amostras, torna qualquer processo computacional inviável. Assim, em Computação Gráfica, possuem somente aplicação teórica [GLAS 95]. Pode-se citar como exemplos as funções de uma variável independente, as funções *y(x)=x+2* e *sen(x)* (Ver figura).

2) Sinais Discretos (Amostrados): Ao contrário dos sinais contínuos, estes são representados matematicamente como uma seqüência finita de números, denotada da forma *x[n]*, onde *n* é definido somente para valores inteiros e representa o n-ésino elemento da seqüência. Esta representação possui uma fácil implementação em computadores digitais com uso de matrizes multidimensionais e por isso, é a forma como os sinais são tratados por processos computacionais (Ver Figura).

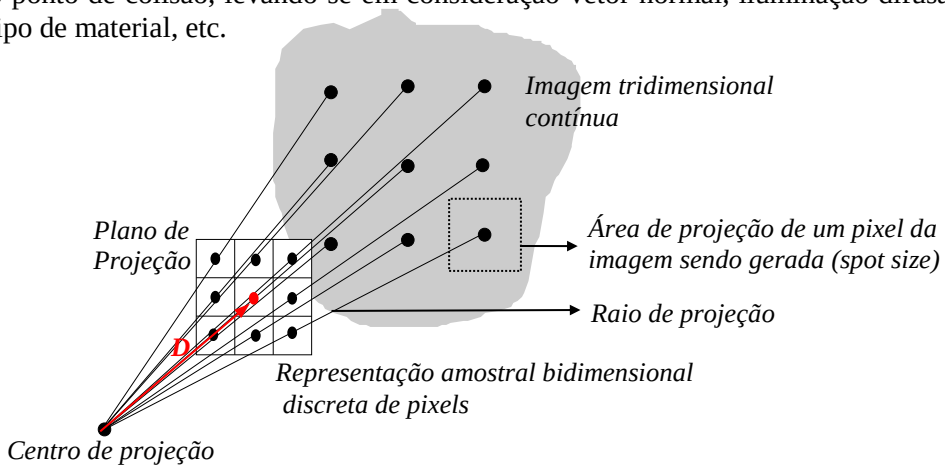


Representação de sinais contínuos (a-b) e equivalentes na forma discreta (c-d)

AMOSTRAGEM DIGITAL DE SINAIS ANALÓGICOS

Como os pixels dos monitores são discretos, Cook [COOK 86] define a Computação Gráfica como um processo de **amostragem** (*sampling*). Entende-se por amostragem, o processo de conversão de um sinal **contínuo em discreto**, seguido por uma **quantificação**, que é a atribuição de uma grandeza numérica a cada ponto amostrado [BRET 92].

Na Síntese de Imagens, o processo de amostragem é semelhante a uma projeção em perspectiva, pois conforme mostra a figura seguinte, as amostras do espaço tridimensional são tomadas pela projeção de raios (**conhecido como algoritmo ray-casting**) que partem de um centro de projeção (observador) e que passam pelos pixels da imagem a ser gerada. Quando um raio colide com algum objeto, deve-se determinar a cor do objeto no ponto de colisão, levando-se em consideração vetor normal, iluminação difusa, especular, ambiente, textura, tipo de material, etc.

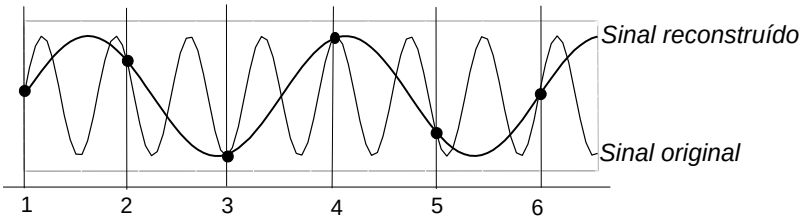


Processo de amostragem de um sinal tridimensional e respectiva projeção

Apesar dos algoritmos de amostragem serem simples, poderosos, e trabalharem facilmente com diferentes tipos de primitivas [COOK 87], eles podem gerar sinais indesejados chamados de ***alias***, que segundo Watt [WATT 93] são facilmente observados em imagens nas formas de:

1) Bordas de objetos dentadas (*jagged silhouette*);
2) Pequenos objetos que podem aparecer e desaparecer durante uma animação;
3) Distorções na imagem gerada (*Moiré interference*).

O *alias* ocorre quando um sinal é **reconstruído** (transformado novamente em contínuo por meio de interpolação) a partir de um conjunto de amostras discretas [JOY 87], o que pode ser visto como resultado de uso **impróprio** da amostragem [EBER 98]. O exemplo apresentado na seguinte figura permite observar que o sinal reconstruído possui frequência menor que o original (possui *alias*). Este fenômeno é o resultado de um erro na reconstrução e indica que toda a informação presente no sinal original não pôde ser fielmente representada pelo sinal amostrado.



Erro de reconstrução (*alias*) devido à baixa taxa de amostragem

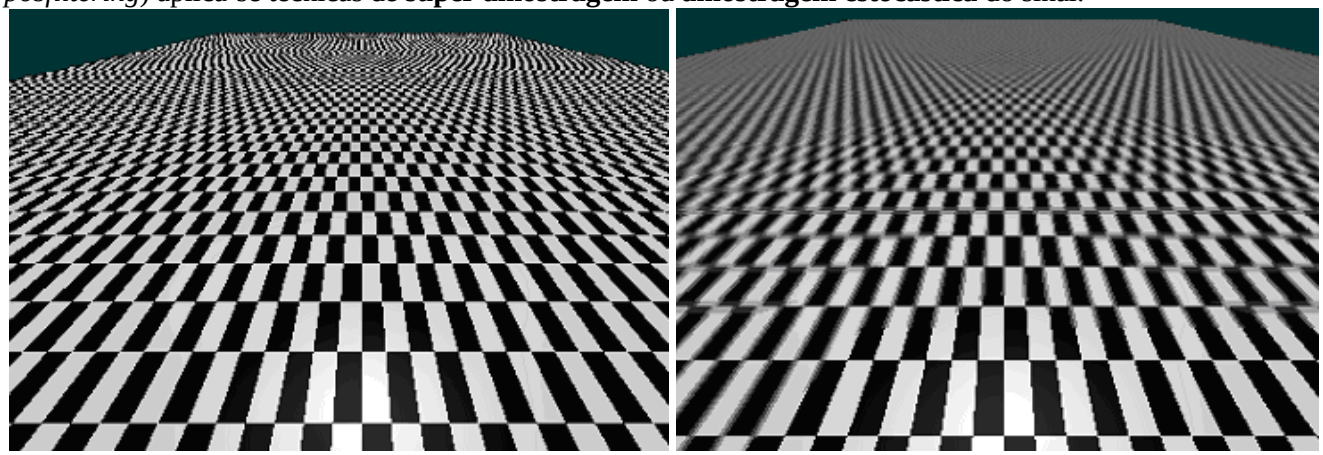
Este problema está associado com a relação que existe entre a quantidade de informação presente no sinal original (*bandwidth*) e a quantidade desta informação que pode ser capturada pelas amostras [EBER 98]. O **teorema de *Nyquist*** diz que o *alias* irá surgir quando a taxa de amostragem for menor que o dobro da frequência máxima do sinal [EBER 98, WATT 93]. Logo, qualquer tentativa de representar detalhes que possuam frequências maiores que a amostragem consegue representar (superior ao limite de *Nyquist*), irá resultar na introdução destes sinais espaciais de baixas frequências [JOY 87].

Segundo Whited [WHIT 80], o *alias* tem maior ocorrência em imagens sintetizadas nos 3 casos:

1) Regiões com mudanças bruscas em intensidade (frequência) e ocorre principalmente com funções do tipo *step*;
2) Quando pequenos objetos se localizam entre pontos de amostragem, ou na representação de objetos que são menores que um pixel [FUJI 86];
3) No mapeamento de uma função, que pode ser textura, sobre uma superfície.

PROCESSOS DE REMOÇÃO DE *ALIAS*

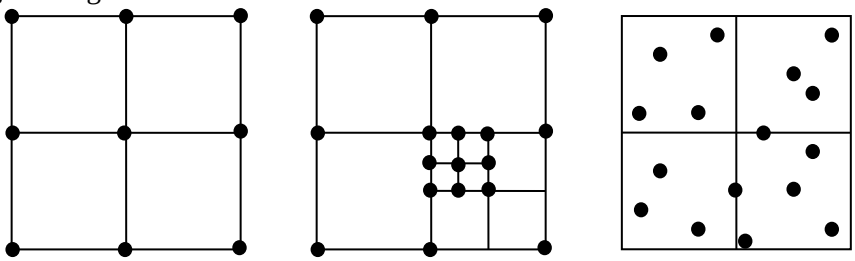
Existem várias formas de remoção de *alias* presentes na literatura. A essas técnicas dá-se o nome de ***anti-aliasing***, que têm como princípio não a sua remoção, mas sim transformá-lo em ruído, o que é bem menos sensível ao olho humano [COOK 86]. O *anti-aliasing* pode ser tanto analítico (quando aplicado a um sinal contínuo) quanto discreto (quando aplicado a um sinal amostrado). Para realizar o processo de forma analítica, são usados **filtros** digitais (processo chamado de *prefiltering*), e de forma discreta (processo chamado de *posfiltering*) aplica-se técnicas de **super amostragem** ou **amostragem estocástica** do sinal.



[http://www.cse.ohio-state.edu/~crawfis/cis782/Labs/SampleLabs/xue/anti_aliasing.html]

Em uma **super amostragem**, faz-se uma amostragem superior à resolução do periférico de saída, criando assim uma imagem virtual que será posteriormente filtrada e novamente reamostrada na frequência do periférico. A avaliação das amostras pode ser realizada pelo uso de "janelas de *Bartlett*" [WATT 93], onde para cada elemento da amostra está associado um peso, dependente da sua posição na janela, que deverá ser multiplicado ao elemento e posteriormente somado aos demais e dividido pelo peso total da janela. Desta forma, o valor final do pixel é a "**média**" do valor de cada uma das amostras localizadas dentro da área de projeção de cada pixel.

Pode-se usar matrizes com qualquer tamanho para fazer a super amostragem. Não existe uma regra fixa para determinar este valor de modo a obter a melhor relação custo/benefício. Um exemplo de uma matriz 3x3 é mostrado na seguinte figura.



(a) Uniforme *(b) Adaptativa* *(c) Randômica*

Diferentes tipos de amostragem. Fonte: [HEAR 97]

Uma solução é usar uma taxa de amostragem que aumenta à medida que a distância da câmera à superfície também aumenta [EBER 98].

Uma variação mais otimizada deste método, desenvolvida por Catmull [CATM 74], consiste em fazer uma amostragem adaptativa, de modo que regiões onde há grandes mudanças de intensidade tenham amostragem maior. Neste caso, o peso da janela é determinado pela área de cada sub-região.

Uma desvantagem deste método é que sendo a intensidade calculada a partir de vários pixels vizinhos, podem ocorrer borrões ou perda de resolução (*blurring*) da imagem, principalmente quando for composta de pequenos objetos [WATT 93].

A principal causa de *alias* em Computação Gráfica é a criação de imagens por um processo de amostragem regular [WATT 93, COOK 86]. Esta conclusão originou-se do trabalho realizado por Yellott [YELL 83] em macacos da espécie *rhesus* - os quais possuem um sistema de visão semelhante ao humano - sobre como é feita a distribuição dos fotoreceptores na retina. Apesar da retina ser completamente coberta por eles, observou-se que eles são **distribuídos de forma aleatória** e daí porque o olho humano não gera *alias* sobre as imagens que processa.

Esta técnica de distribuição não regular de pontos é chamada ***stochastic sampling*** (amostragem estocástica) e pode eliminar todas as formas de *alias*, que é substituída por ruído ou por sinais que são menos perceptíveis pelo sistema de visão humano [COOK 86].

A única restrição que deve haver é uma distância mínima entre cada amostragem, o que tende a uma redução do ruído gerado.

Com o uso de **filtros digitais**, também chamados analíticos, pode-se eliminar certos tipos de *alias* com eficiência [COOK 86]. Seu princípio baseia-se na filtragem de sinais com frequências muito altas (superiores a *Nyquist limit*), que são as responsáveis por *alias*, e daí serem conhecidos como filtros passa-baixa.

Eles são aplicados à imagem antes do processo de amostragem ser realizado, caracterizando um processo chamado de *prefiltering*. Entretanto, esta filtragem prévia pode gerar problemas como alteração significativa de toda imagem, pois toda ela irá perder a intensidade pela existência de algumas amostras que estão fora da faixa esperada de freqüências. Além disto, por serem complexos e por gastarem muito tempo de processamento, são pouco usados [COOK 86]. Vale observar desde já que o custo de geração de imagens usando técnicas de *anti-aliasing* é muito maior do que usando uma amostragem unitária por pixel.

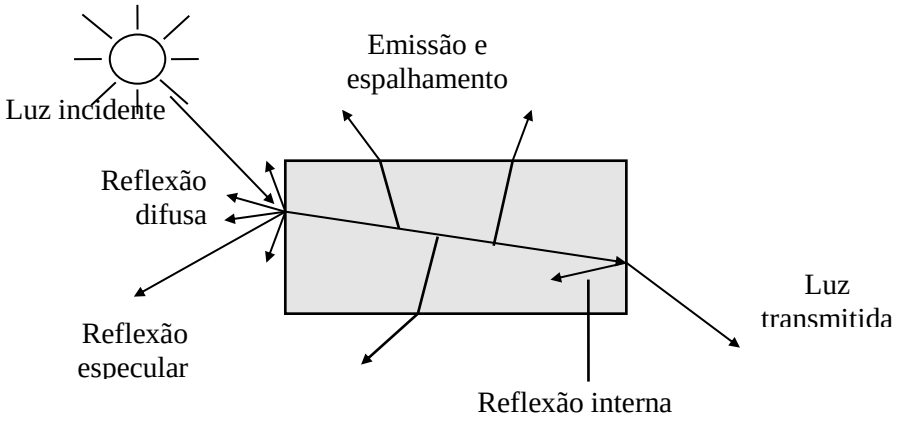
2 Modelos de Reflexão e Iluminação

A luz que incide (iluminação) sobre uma superfície pode ser dividida em 4 diferentes partes: refletida, espalhada, absorvida e transmitida, como mostrado na seguinte figura.

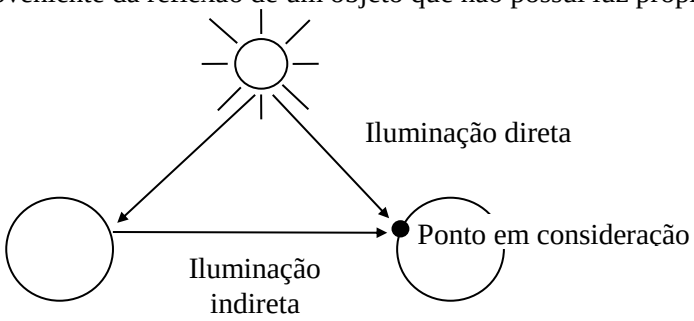
1) **Luz refletida:** proveniente de outros objetos ou resultado da iluminação ambiente. Pode ser tanto especular (direcional) quanto difusa (em todas as direções);
2) **Luz espalhada:** se o objeto for trasparente, parte da luz que ele absorve é distribuída ao ambiente e age como se o objeto tivesse luz própria;
3) **Luz absorvida:** luz que o objeto absorve. Ocorre principalmente em objetos opacos;
4) **Luz transmitida:** proveniente de refrações internas do objeto, no caso de transparência.

O modelo de iluminação é usado para calcular a quantidade de luz que deve ser vista em cada ponto da superfície de um objeto. O modelo de reflexão faz com que projeções em duas dimensões de objetos tridimensionais no mundo real se pareçam da forma mais real possível.

Ambas reflexão e iluminação estão diretamente relacionadas, pois somente irá ocorrer uma reflexão se houver um feixe de luz incidindo sobre o objeto e a iluminação somente terá significado se for considerado o feixe refletido em direção ao observador.

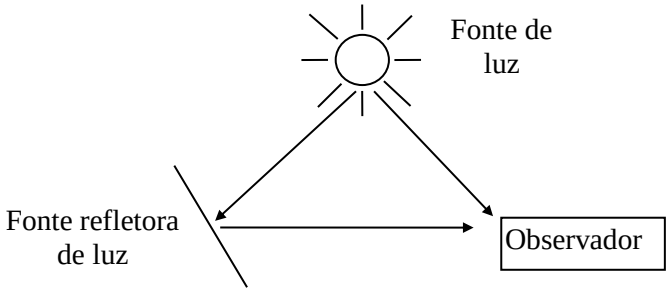


Este capítulo destina-se a dar uma visão geral e simples do processo de iluminação. Será considerada a iluminação local (direta), ou seja, somente pontos cuja luz incidir diretamente da fonte de luz serão tratados, ao contrário da iluminação global, onde ambas iluminações direta e indireta são consideradas. Iluminação indireta é uma iluminação proveniente da reflexão de um objeto que não possui luz própria.



É considerado uma **fonte de luz** um objeto que emite luz própria, como no caso de uma lâmpada ou o sol. **Fonte refletora de luz** é um objeto que emite luz proveniente de outro objeto.

O modelo mais simples para uma fonte de luz é a **fonte pontual**. Os raios de luz deste tipo de fonte partem radialmente da fonte em todas as direções. Este modelo de fonte de luz é uma boa aproximação para fontes cujas dimensões são menores comparadas com o tamanho dos objetos da cena. Fontes como o sol, pela sua distância, podem ser modeladas como fontes pontuais. Também existem fontes de luz não pontuais (***distributed light source***). Pode-se simular fontes não pontuais pela combinação de várias fontes pontuais.



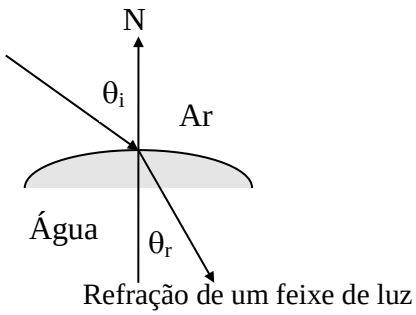
REFRAÇÃO DA LUZ

A aplicação de técnicas de reflexão de refração de luz dá ao processo de iluminação a capacidade de geração de imagens próximas ao natural.

A refração ocorre quando a luz atinge uma superfície que separa dois meios diferentes, sendo eles, meios transparentes. Quando um feixe de luz é incidido, parte desta energia é refletida e outra penetra no meio. Ao penetrar no meio, devido a diferentes índices de refração, a luz sofre um desvio. A relação entre os ângulos de incidência e reflexão é mostrado na seguinte equação.

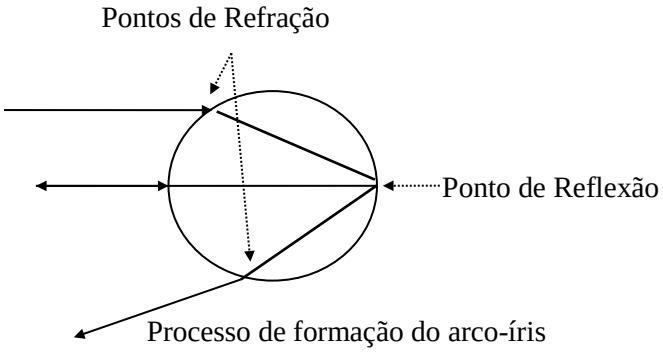
$$\cos\theta_r = \sqrt{1-\left(\frac{n_i}{n_r}\right)^2 (1-\cos^2\theta_i)}$$

A seguinte figura mostra um exemplo desta mudança de direção quando a luz passa do ar para a água. Como o índice de refração é a razão entre a velocidade da luz no vácuo com a velocidade da luz no meio, qualquer substância terá um maior índice de refração que o ar. Logo, quanto um feixe passa de um meio menos denso para um mais denso, a tendência é o feixe refratado se aproximar da normal [TIP 91].

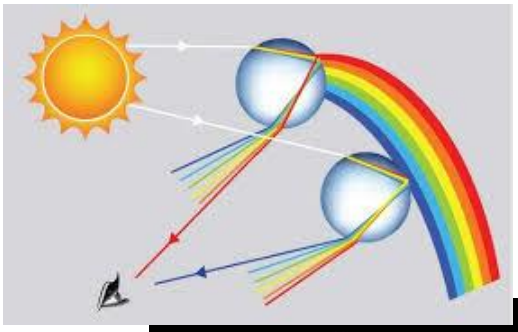


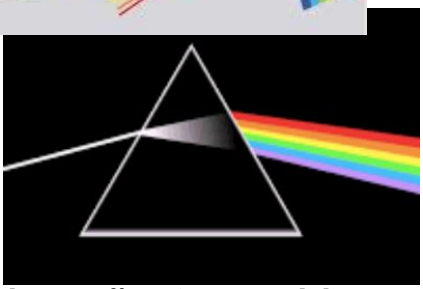
Refração de um feixe de luz

Um exemplo da ocorrência destes dois fenômenos simultaneamente pode ser visto quando um feixe de luz incide sobre uma gota de água. A luz ao incidir na superfície da gota, inicialmente sofre uma refração, ou seja, uma mudança de direção decorrente dos diferentes índices de refração do ar e da água. A luz então é refletida na parte inferior da gota e sofre outra refração ao sair dela. Caso a luz incidente for branca, o ângulo de refração na saída será diferente para cada componente que compõe a luz, gerando com isso o arco-íris. A seguinte figura mostra os pontos de reflexão e refração.



Processo de formação do arco-íris





MODELOS DE ILUMINAÇÃO

O cálculo da iluminação leva em consideração as propriedades óticas da superfície (opaca, polida, transparente), condições de luz de fundo e especificação de fontes de luz (pontuais ou não). Pela combinação destes diversos parâmetros pode-se determinar a luz refletida por cada objeto.

A cor de um objeto é definida pela componente da cor que ele reflete. Logo, se uma fonte vermelha incidir sobre um objeto que reflete luz verde, não ocorrerá reflexão e o objeto parecerá negro.

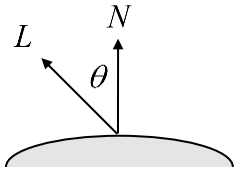
Três componentes de luz devem ser consideradas em um modelo de iluminação:

- **Luz Ambiente**: Qualquer objeto em um ambiente não escuro pode ser visualizado, mesmo que não exista a incidência direta de luz sobre ele. A iluminação gerada pela reflexão dos objetos da cena caracteriza a iluminação ambiente. A cor refletida é a cor do objeto. Esta iluminação não tem direção. A quantidade de luz incidente em cada objeto, independente de posição ou orientação, é sempre a mesma, ou seja, a intensidade é definida por uma constante I_a. Utilizando-se uma constante de reflexão de cada superfície k_d, tem-se que a iluminação ambiente difusa é dada por

$$I_{ambiente} = k_d I_a$$

- **Reflexão Difusa**: A reflexão difusa ocorre quando a luz incide numa superfície **não idealmente polida**. Um material refletor difuso ideal se caracteriza pelo espalhamento da energia em **todas as direções, com a mesma intensidade**, independente do ângulo de visão do observador. A cor refletida é a cor do objeto. A fração de luz incidente em uma superfície que é refletida de forma difusa é especificada por um parâmetro k_d, que varia entre 0 e 1. Quanto maior for seu valor, maior será a reflexão. Quando calculada para iluminação não ambiente, como fontes pontuais, a intensidade refletida é atenuada por um fator proporcional ao cosseno do ângulo formado pela fonte *i* com a superfície (vetor L) e atinge o valor máximo quando a luz incidir perpendicularmente à superfície. Assim o modelo assume a seguinte forma

$$I_d = k_d I_i \cos(\theta) \qquad (0 \leq \theta \leq \pi/2)$$



Se N e L estão normalizados, $cos\theta$ pode ser dado por

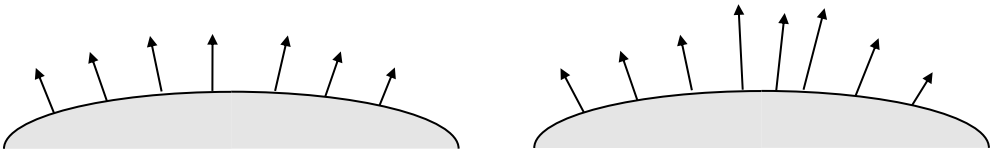
$$I_d = k_d I_i N \cdot L$$

onde L representa o vetor direção da luz e N o vetor normal. $N \cdot L$ representa o produto escalar entre a Normal e o raio L. Combinando-se tanto a iluminação ambiente quanto a de fontes pontuais, pode-se expressar a reflexão difusa total como

$$I_d = I_a k_d + k_d \sum_{i=0}^{n} I_i N \cdot L_i$$

onde I_i é a intensidade da fonte *i*.

- **Reflexão Especular**: Superfícies lisas, como metal polido, apresentam pontos brilhosos, sob certos ângulos de visualização. Isso é resultado da reflexão **especular**, que consiste em refletir a maior parte da energia incidente em uma direção específica, ao contrário da reflexão **difusa** onde a energia reflete igualmente em todas as direções. A cor refletida é a cor da fonte luminosa.

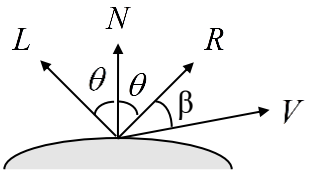


Difusa Especular

Diferença entre reflexão difusa de especular

MODELO *PHONG* DE REFLEXÃO ESPECULAR

O processo de iluminação de objetos polidos, como o ferro, tende a gerar pontos brilhantes da cor da fonte luminosa. Este efeito se deve à reflexão especular e é resultado da reflexão da luz na direção do vetor de reflexão especular *R*.



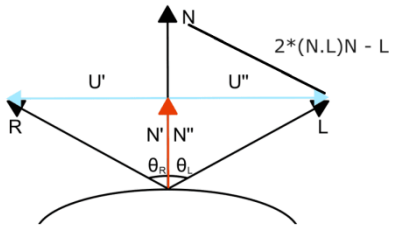
Formação da reflexão especular

A reflexão especular é dada por $I_s = w(\theta)\, cos^n \beta$, onde $w(\theta)$ é uma função que fornece a razão entre a luz especular refletida e a luz incidente, levando em consideração o ângulo de incidência θ. Muitas vezes, por questão de simplicidade e para aumento de desempenho, a função $w(\theta)$ é substituída pelo coeficiente k_s, que é um valor médio da função w(θ).

$$I_s = k_s\, cos^n \beta$$

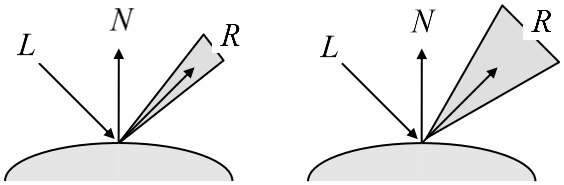
Observa-se que o vetor de reflexão especular *R* forma o mesmo ângulo que o vetor *L*, proveniente da fonte, em relação a normal *N*. *V* é o vetor que aponta para o observador e forma um ângulo β com o vetor de reflexão especular *R*. Somente em superfícies refletoras ideais, a luz é refletida somente na direção do vetor *R*. Neste caso, um observador em *V* somente perceberá a reflexão *R* se ambos vetores forem coincidentes, ou seja, $\beta = 0$.

O vetor R é definido pela equação:



$$R = 2(N \cdot L)N - L$$

Como na prática não existem superfícies ideais, a reflexão especular para superfícies polidas concentra-se nas proximidades do vetor de reflexão especular *R* (cone preferencial de reflexão especular), como é mostrado na seguinte figura (a). Já superfícies menos polidas, tendem a divergir o feixe refletido.



Reflexão especular para diferentes graus de polimento

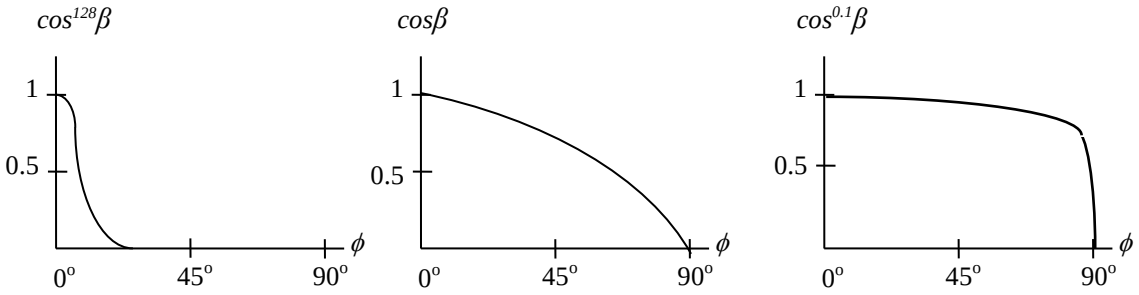
Pode-se concluir que quanto mais polida for a superfície, a componente especular é mais ativa e, à medida que a superfície se torna mais rugosa, a componente especular perde a intensidade para a componente difusa. A relação entre as duas componentes é dada pela expressão

Reflexão = f(i)Especular + (1- f(i))Difusa

onde *f(i)* representa o grau de polimento da superfície *i* e está definido no intervalo 0≤ *f(i)*≤1.

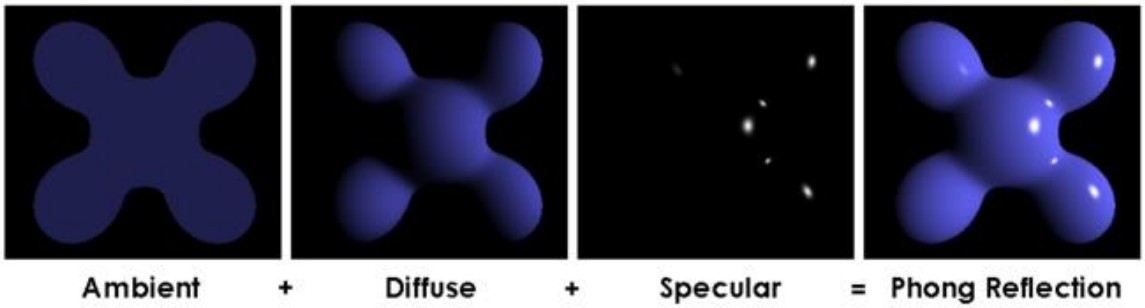
Um modelo para calcular a reflexão especular foi proposto por Phong [PHON 75] e é conhecido como *Modelo Phong de Reflexão Especular*. Este modelo define a intensidade de reflexão especular como sendo proporcional a $cos^n\beta$, onde β varia entre 0 e 90°, e representa o ângulo entre *R* e *V*. O valor de *n* (parâmetro de reflexão especular) é determinado pelo polimento *f(i)* da superfície e assume valores maiores para superfícies polidas.

A seguinte figura mostra o efeito de *n* na determinação da reflexão especular. Observa-se que, quanto maior for seu valor, menor será o cone de reflexão especular.



Valores de $cos^n\beta$ para diferentes ângulos de incidência e diferentes valores de *n*

Combinando-se a iluminação ambiente + difusa + especular, obtém-se o modelo phong de reflexão especular.



[http://en.wikipedia.org/wiki/Phong_reflection_model]

A iluminação (difusa e especular) pode ser aplicada por vértice ou por pixel. Na iluminação por vértice (forma como o OpenGL antigo operava), dada uma geometria (ex: triângulo), define-se a cor de cada um de seus vértices em função dos vetores normais associados. Uma vez calculada a cor, faz-se uma interpolação linear da cor dos vértices para calcular a cor de cada pixel da geometria. Em uma iluminação por pixel (versão atual do OpenGL - shaders), calcula-se o vetor normal para cada pixel pela interpolação dos vetores normais da geometria. Então aplica-se sobre esse vetor os cálculos de iluminação para calcular a cor do pixel.

enade 2017
Exame Nacional de Desempenho dos Estudantes

QUESTÃO 27

Em computação gráfica, existem vários modelos de iluminação diferentes que expressam e controlam os fatores que determinam a cor de uma superfície em função de um determinado conjunto de luzes. Uma vez definido um modelo de iluminação, pode-se aplicar luz sobre as várias faces dos objetos de uma cena, processo denominado sombreamento.

As figuras a seguir ilustram a aplicação de dois modelos de iluminação, a saber: o modelo de sombreamento constante (à esquerda) e o modelo de Phong (à direita).

AZEVEDO, E.; CONCI, A. **Computação gráfica**: geração de imagens. Rio de Janeiro: Campus, 2003 (adaptado).

Disponível em: <https://www.cs.cmu.edu>. Acesso em: 17 jul 2017.

Em relação aos modelos de iluminação apresentados, avalie as afirmações a seguir.

I. A aplicação do modelo de sombreamento constante causa na imagem um efeito visual denominado Bandas de Mach.
II. Embora seja útil para gerar imagens realísticas, o modelo de Phong mostra-se pouco eficiente na apresentação das reflexões especulares.
III. O modelo de sombreamento constante não é útil para gerar imagens realísticas porque ele dá destaque ao aspecto facetado da representação poliedral das superfícies.
IV. Para a utilização do modelo de Phong, é necessário supor que a fonte de luz localiza-se no infinito.

É correto apenas o que se afirma em

A) I e II.
B) I e III.
C) II e IV.
D) I, III e IV.
E) II, III e IV.

3 Textura

A síntese de Imagens é uma área destinada à geração de imagens realistas, a partir de uma descrição geométrica de um cenário tridimensional. A geração da cena 3D é uma etapa inicial da síntese e, sobre este modelo, são adicionados vários níveis de realismo, que vão de simples processos de remoção de superfícies escondidas até as técnicas mais complexas para descrição de detalhes sutis, que procuram imitar ao máximo aspectos presentes na natureza.

A textura é uma técnica que quando aplicada junto à iluminação, procura dar às superfícies dos objetos características que os façam parecer mais reais, quando comparados a simples técnicas de iluminação e sombreamento.

A textura pode ser descrita genericamente como um processo para **remoção da continuidade de cores**. Algoritmos de iluminação, como já estudados, também procuram fazer a mesma coisa.

A pergunta que surge é: os métodos de iluminação não são bons o suficiente para dar as superfícies um alto grau de realismo? A resposta é sim. Geralmente, síntese de imagens baseada apenas em iluminação produz superfícies com uma variação muito contínua de cores. Apenas variações de intensidades são notadas.

Neste momento, entra em cena a textura, que pode agir sobre diversos parâmetros que definem a superfície. Textura é um conjunto de informações, em alguma dimensão (2D, 3D), que é usado para remover o aspecto contínuo de variação de cores de uma imagem gerada por um processo computacional a partir de uma descrição 3D de uma cena.

A textura pode ser obtida pela modulação de atributos do objeto [HECK 86]:

1) **Cor da superfície**: Este é o parâmetro mais usado no mapeamento de textura (*texture mapping*), pela modulação do coeficiente difuso da luz.
2) **Reflexão Especular e Difusa**: Este método é conhecido como *environment mapping* e não é considerado uma técnica de textura, mas uma versão simplificada do método de iluminação *Ray Tracing*;
3) **Perturbação do vetor normal**: Muda-se a direção do vetor normal às superfícies. Esta técnica é chamada de *bump mapping*.
4) **Transparência**: Neste método não são alterados atributos dos objetos, mas sim, o objeto é gerado usando uma função matemática para modular sua transparência.

TIPOS DE TEXTURAS

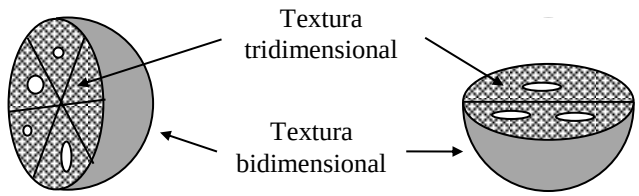
As texturas, sob o enfoque visual, podem ser **bidimensionais** ou **tridimensionais**. Ambas são visíveis na superfície externa do objeto, porém as tridimensionais [PERL 85, PEAC 85] estão definidas em todo o espaço ocupado pelo objeto.

A textura 2D é a forma mais comum de textura, que geralmente é definida na forma paramétrica T(u,v). O termo bidimensional não se refere ao fato que este tipo de textura deva ser plana, mas sim ao fato que ela não possui um volume. Pode-se citar exemplos deste tipo de textura a pele de um animal, tecido estampado, dentre outros. Geralmente é representada por meio de **imagens** (mapa de bits).

A textura 3D, também chamada textura sólida, a cor do objeto é determinada pela interseção de sua superfície com o correspondente ponto presente no espaço da textura. Este processo é equivalente a esculpir a forma do objeto de um bloco do material que se deseja moldar. Como exemplos de texturas tridimensionais, pode-se citar materiais não homogêneos como mármore, madeira, nuvens, etc.

Não somente o exterior do objeto possui a textura, mas sim, todo o seu interior também. Entretanto, a textura interior do objeto somente pode ser visualizada se ele for transparente. Caso não, deve-se fazer cortes no objeto para observar seu interior. São exemplos claros deste tipo de textura o mármore e a madeira, que possuem a característica de dependendo de onde se observa, pode-se ver nitidamente não possuem uma textura homogênea no espaço. Um exemplo mais ilustrativo pode ser visto com uma fruta cítrica. Sua textura externa pode ser definida por uma textura bidimensional, pois é homogênea; já seu interior, irá possuir sementes e gomos, que dependendo de como a fruta for cortada, se mostrarão de forma diferente.

Os dois tipos de textura podem ser gerados de forma procedimental (com o uso de algoritmos) ou não procedimental. As não procedimentais podem ser definidas por **imagens** digitalizadas (mapa de bits) ou qualquer outro tipo de informação que não seja resultado de um processo algorítmico, geralmente necessitando de maior quantidade de memória para armazenamento, principalmente quando forem tridimensionais (Texturas 3D geralmente são procedimentais).



(a) Corte Transversal (b) Corte Longitudinal

Algumas vantagens decorrentes desse tipo de textura são: fácil geração e menor tempo de avaliação quando comparadas aos métodos procedimentais. Existem também os aspectos negativos associados ao método não procedimental, que ocorrem quanto a textura é aplicada a objetos sem formas bem definidas. Nestes casos, ocorrem falhas ou sobreposições dos padrões, que exigem a aplicação de métodos de *anti-aliasing* para serem removidos, eliminando desta forma a vantagem apresentada pela técnica em consumir pouco tempo de processamento para aplicar a textura.

Ao contrário das não procedimentais, as procedimentais são geradas por procedimentos baseados em métodos implícitos que, geralmente, usam como argumentos de entrada as coordenadas cartesianas do ponto do objeto em que é aplicada a textura e o tempo, no caso de animações. Como resultado, retornam um valor correspondente ao ponto especificado.

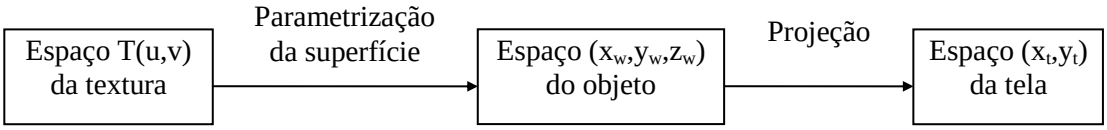
A principal vantagem deste método é a abstração [EBER 98]. Detalhes não são mais explicitamente especificados, pois podem ser gerados pela adição/combinação de técnicas/funções e pela alteração das variáveis de controle. Isso traz uma economia de armazenamento e, em muitos casos, transfere o trabalho do programador visual para o computador.

Pelo fato da textura ser avaliada e gerada por um processo algorítmico, ela pode ser obtida em qualquer **nível de detalhe** e sem limites de extensão. Não existe a limitação de tamanho imposta pela textura digitalizada, ou seja, para qualquer argumento de entrada, sempre existirá um valor associado.

APLICAÇÃO DE TEXTURA

Muitos métodos foram desenvolvidos para fazer a aplicação de texturas. A técnica que usa o princípio de colar um padrão de imagem é chamada ***texture mapping*** [CATM 74].

De forma genérica, o Texture mapping consiste em fazer o mapeamento de uma imagem bidimensional na superfície de um objeto (geralmente tridimensional). Este processo tem a função de aplicar texturas bidimensionais pelo mapeamento da superfície do objeto sobre a textura, e que pode ser realizado em uma ou duas etapas. Na seguinte figura é mostrado o processo em duas etapas, ou seja, primeiramente a textura é aplicada sobre o objeto para posterior projeção na tela.



Etapas para visualização de texturas

A segunda etapa deste processo pode ser feita usando algum método de iluminação e visualização de superfícies, como *Ray-Tracing*, Radiosidade, dentre outros.

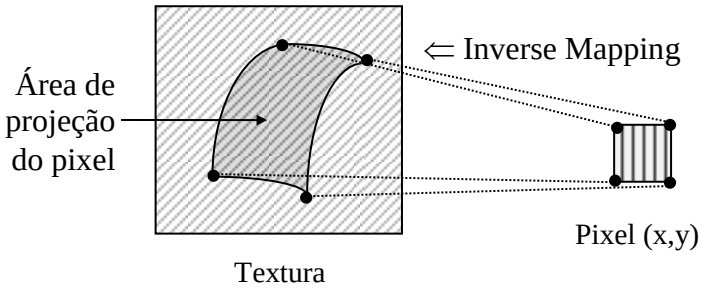
O primeiro já não é um processo tão direto quanto o segundo. Ele está ligado com a forma como a superfície está representada e com o tipo de textura que se deseja aplicar. Basicamente a superfície do objeto pode ser representada de duas formas:

1) **Por faces**: Neste caso, a textura é aplicada individualmente sobre cada face, e caso não havendo um casamento das áreas, deve-se fazer um redimensionamento da textura para cada uma das faces (problema do mapeamento local), o que pode gerar problemas de falta de continuidade como um todo, sem contar que em muitos pontos do objeto serão visíveis grandes contrastes (problema do mapeamento global). Em outras palavras, neste tipo de representação é extremamente difícil encontrar uma forma de parametrizar esta superfície para que se possa fazer o mapeamento da textura de forma homogênea sobre todo objeto. Outro problema que surge neste caso é em relação ao *aliasing*;

2) **Parametricamente**: Neste tipo de superfície (Bézier ou B-spline, por exemplo), o processo de mapeamento da textura é de forma mais direta, pois variando-se os índices *u* e *v*, presentes tanto na textura quanto no objeto, obtêm-se projeção de uma superfície na outra. O primeiro uso de textura em computação gráfica foi feita usando este método, com um algoritmo proposto por [CATM 74].

Em termos mais gerais, precisamos encontrar qual o valor da textura para cada pixel do objeto. Pode-se fazer uso da técnica de ***Inverse Mapping*** ou **Mapeamento por Superfície Intermediária**.

1. No ***inverse mapping***, em vez de fazer a "projeção" da textura sobre o objeto, é realizado o processo inverso, ou seja, cada pixel do objeto é projetado sobre a textura e a imagem correspondente será aplicada ao pixel. Como geralmente o objeto não possui uma superfície plana, a área de projeção do pixel será um quadrilátero curvilíneo. Como esta projeção não é paralela, ao se fazer a projeção da textura sobre o pixel, verifica-se pela figura que podem haver modificações da textura original. Uma grande vantagem deste método, é que ele facilita o uso de *anti-aliasing*, que é um dos maiores problemas encontrados neste tipo de textura.



Projeção do pixel sobre a textura

2. O **mapeamento por superfície intermediária** visa suprir os problemas e dificuldades oriundas do mapeamento global. É um método novo, proposto por [BIER 86] para mapeamento de texturas bidimensionais sobre qualquer tipo de superfície definida por polígonos, sem a necessidade da geração de uma equação paramétrica que a represente. O processo, também conhecido como mapeamento em duas fases, resulta em uma dupla distorção da textura original até o mapeamento sobre o objeto:

    1) *Mapeamento da textura sobre uma superfície intermediária*: Nesta fase, a textura é mapeada sobre uma superfície intermediária tridimensional. Esta transformação é feita sem dificuldade, pois esta superfície intermediária possui uma forma simples, como um cilindro, um plano com qualquer orientação, uma esfera ou uma caixa. Esta transformação é conhecida como *S mapping*, e é dada por

$$T(u,v) \rightarrow T'(x_i, y_i, z_i)$$

    A escolha correta desta superfície intermediária que melhor se adequa a textura deve ser feita baseada na sua forma. Ao se usar esferas, a distorção será mais concentrada nos pólos.

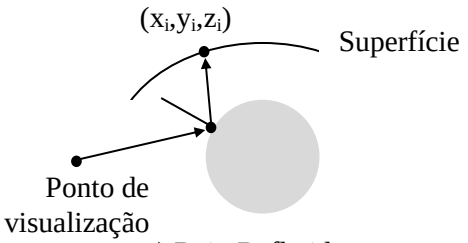
    2) Mapeamento da textura intermediária sobre o objeto: Esta operação é conhecida como *O mapping* e é dada por

$$T'(x_i, y_i, z_i) \rightarrow O(x_w, y_w, z_w)$$

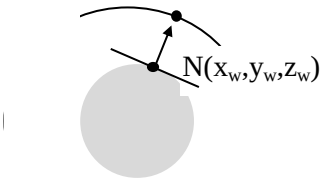
    Agora tendo-se a textura em um espaço tridimensional, como resultado da primeira etapa, o mapeamento para o objeto se torna um processo tridimensional, e é feito por uma parametrização do vetor normal e não mais pela parametrização da superfície.

Existem 4 tipos de *O mapping*, conforme mostrados na seguinte figura.

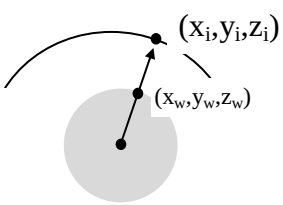
1) O mapeamento é baseado no ângulo do raio refletido proveniente do observador sobre a superfície intermediária *T*'. Este tipo de mapeamento é conhecido como *Environment Mapping*. Pode-se utilizar como superfície intermediária um cilindro, uma baixa, uma esfera ou um plano. ;
2) A interseção é baseada no vetor normal ao objeto no ponto (x_w, y_w, z_w) com *T*';
3) Calcula-se a interseção do vetor formado pelo centro do objeto com o ponto (x_w, y_w, z_w) de sua superfície com *T*';
4) É usado um vetor perpendicular a superfície intermediária que passa pelo ponto (x_w, y_w, z_w).



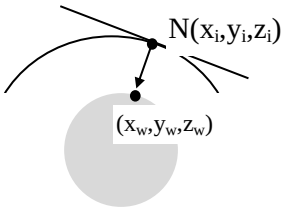
(1) Raio Refletido



(2) Normal ao objeto



(3) Centro do objeto



(4) Normal a superfície intermediária

Tipos de *O mapping*

A aplicação de **texturas tridimensionais** é a mais simples de todas, pois não há a necessidade de se usar superfícies intermediárias ou qualquer outra função de ajuste de modo que a textura seja homogeneamente distribuída sobre o objeto, o que implica que não haverá distorções da textura, independente da forma e tamanho do objeto.

O processo de mapeamento é realizado de forma direta. Ou seja, tendo-se o objeto definido de forma paramétrica ou por faces, acha-se os pontos que definem sua superfície externa e mapeia-se diretamente estas coordenadas sobre a função que determina a textura tridimensional, ou seja, a cor correspondente para o dado pixel.

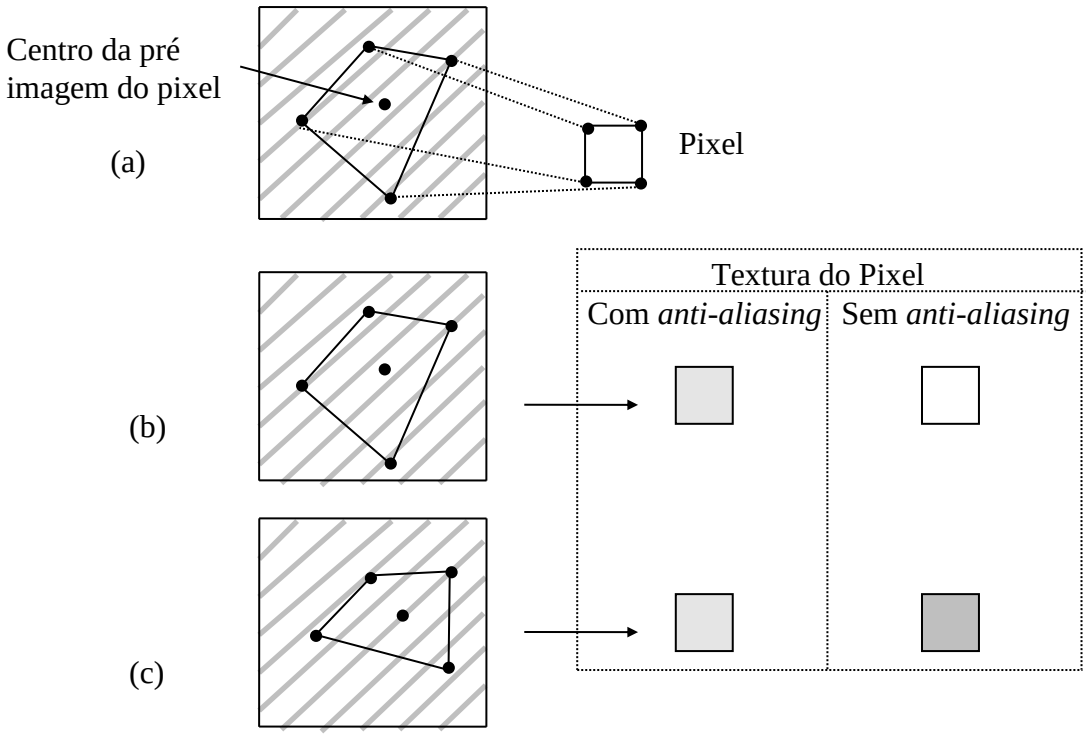
corPixel = textura[x][y][z]

ANTI-ALIASING DE TEXTURA

Independentemente do tipo de textura usado e da forma do objeto, o processo de *anti-aliasing* deve ser aplicado, pois descontinuidades são facilmente visíveis quando texturas que apresentam periodicidade são aplicadas sobre objetos não uniformes.

Para texturas tridimensionais definidas procedimental, este é um processo simples, pois deve ser aplicado somente a pixels, e não sobre a forma como a textura é distribuída sobre a superfície. Pode ser realizado pela utilização de uma super amostragem (*supersampling*) de pontos, gerados pelo procedimento. Tendo-se vários pontos que definem o pixel, torna-se fácil verificar a cor que mais se aproxima aos pontos dados.

Já para processos onde também existem problemas com o encaixe da textura, como no *texture mapping*, é indispensável o uso de um eficiente *anti-aliasing*. Outro problema que surge neste caso é quando os objetos estão muito longe do observador e sua área de projeção sobre a textura é muito grande, conforme mostrado na seguinte figura. Nela, temos em (a) a projeção do pixel sobre a textura. Nas figuras (b) e (c) são mostrados resultados da aplicação da pré imagem sobre o pixel com e sem a utilização do *anti-aliasing*, caso for tomado como referência o centro geométrico desta pré imagem. Verifica-se que com aplicação do anti-aliasing, o pixel resultante não tem influência só do valor central da pré imagem, mas sim de toda ela.



Uso do *anti-aliasing* em *texture mapping*

4 Realismo

A capacidade de gerar imagens o mais perto de realidade está, em grande parte, relacionado com a capacidade de processamento e de armazenamento nos computadores. A geração de imagens de objetos do mundo real é feita em várias etapas. Numa primeira etapa é feita uma definição dos pontos que irão definir a estrutura básica do componente a ser gerado. Por limitações de capacidade de armazenamento e processamento, geralmente esta representação inicial é feita em estrutura de arame (*wireframe*). Sobre esta estrutura, podem ser aplicados diversos métodos matemáticos, como a interpolação dos pontos que a compõem, para dar mais realismo, ou seja, amenizar a presença dos polígonos que a compõem. Os objetos presentes no mundo real, além de uma forma específica, que geralmente não é homogênea, possuem várias outras características que os tornam únicos, como no caso da textura. A textura pode ser vista como um padrão de imagem que, podendo não ser homogênea, como, por exemplo, o mármore, se repete por toda a extensão do objeto.

Outro fator muito importante a ser considerado na geração de imagens que possuem um grande número de detalhes é a iluminação que ela está recebendo. Pode-se ter infinitas representações de um objeto pela variação na direção ou intensidade de luz que ele está recebendo.

Relacionado também com a luz de incidência, o tipo de material e sua textura, pode-se verificar que certos objetos são transparentes, ou seja, deixam a luz passar por ele. Essa luz, por propriedades físicas, pode sofrer mudanças na sua direção (refrações e reflexões), que irão dar um toque de realismo à figura e, em muitos casos, mostrar a influência da presença de outros objetos na geração de sua imagem que será vista pelo observador. Para tratar a interação da luz com objetos e assim determinar a cor de cada pixel de forma um tanto realista pode-se utilizar a técnica de ***Ray tracing***.

Existem métodos que não se baseiam no princípio de "colar" sobre a superfície do objeto um padrão de imagem que irá caracterizá-lo, e sim, tratam a forma como a luz interage com a superfície. A técnica de ***Environment Mapping*** [BLIN 76] usa-se da reflexão para projetar sobre a superfície do objeto o ambiente que o cerca. Também chamada de *reflection mapping*, pode ser vista como uma simplificação do processo de ***ray tracing***, pois a textura que o objeto irá receber depende de onde o observador se encontra e do ambiente que o cerca. Nesta categoria, destaca-se o ***bump mapping*** [BLIN 78], que é um procedimento que altera a direção do vetor normal da superfície, mudando desta forma, a interação da luz com a superfície.

BUMP MAPPING

Embora o *texture mapping* possa ser usado para adicionar rugosidade a uma superfície, este não é o método mais adequado. A geração de texturas que se assemelham com cascas de frutas como da laranja ou morango, que são um caso típico de textura que se repete de forma homogênea, pode ser feita pelo uso de uma função de perturbação do vetor normal da superfície para posterior aplicação de um método de iluminação. Esta técnica é conhecida como *bump mapping* e utiliza-se de um mapa de perturbação (*bump map*) na definição da intensidade de perturbação que o vetor normal de cada pixel receberá. Com esta técnica, pode-se simular rugosidade sem a necessidade de se modelar geometricamente a superfície.

Este método foi desenvolvido para ser aplicado a superfícies paramétricas da forma *P(u,v)*, onde o vetor normal *N* é dado pelas derivadas parciais P_u e P_v, que definem o plano tangente da superfície em cada ponto.

De forma simplificada, o objetivo geral do método é encontrar um valor *D*, de modo que a "*normal perturbada" N'* seja dada por

$$N' = N + D$$

onde *D* é o valor da perturbação, definido em termos de uma função (*bump function*), também paramétrica, da forma *F(u,v)*, que define o deslocamento do vetor normal *N* para cada ponto *P* da superfície.

O primeiro passo do método consiste em encontrar os vetores *A* e *B*, conforme apresentados na seguinte figura, e cuja forma de geração é dada por

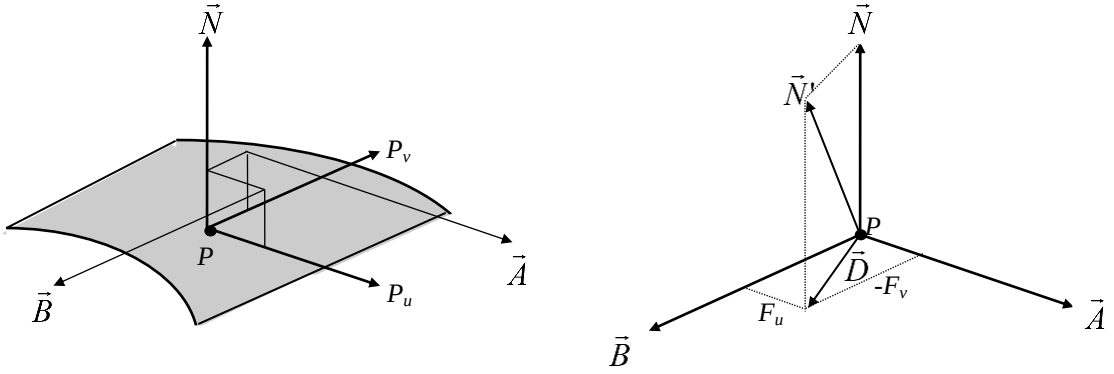
$$A = N \times P_v$$

$$B = N \times P_u$$

Obtidos os dois vetores, deve-se multiplicá-los pelas derivadas parciais F_u e F_v da função de perturbação *F(u,v)*, da seguinte forma:

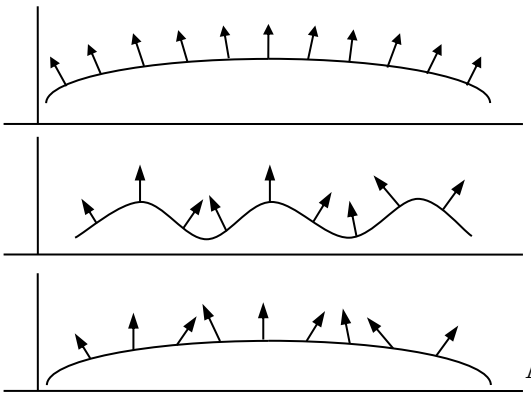
$$D = F_u A - F_v B$$

A visualização gráfica deste processo pode ser vista na seguinte figura, que mostra a influência do vetor *D* na perturbação do vetor normal *N*.



(a) (b)

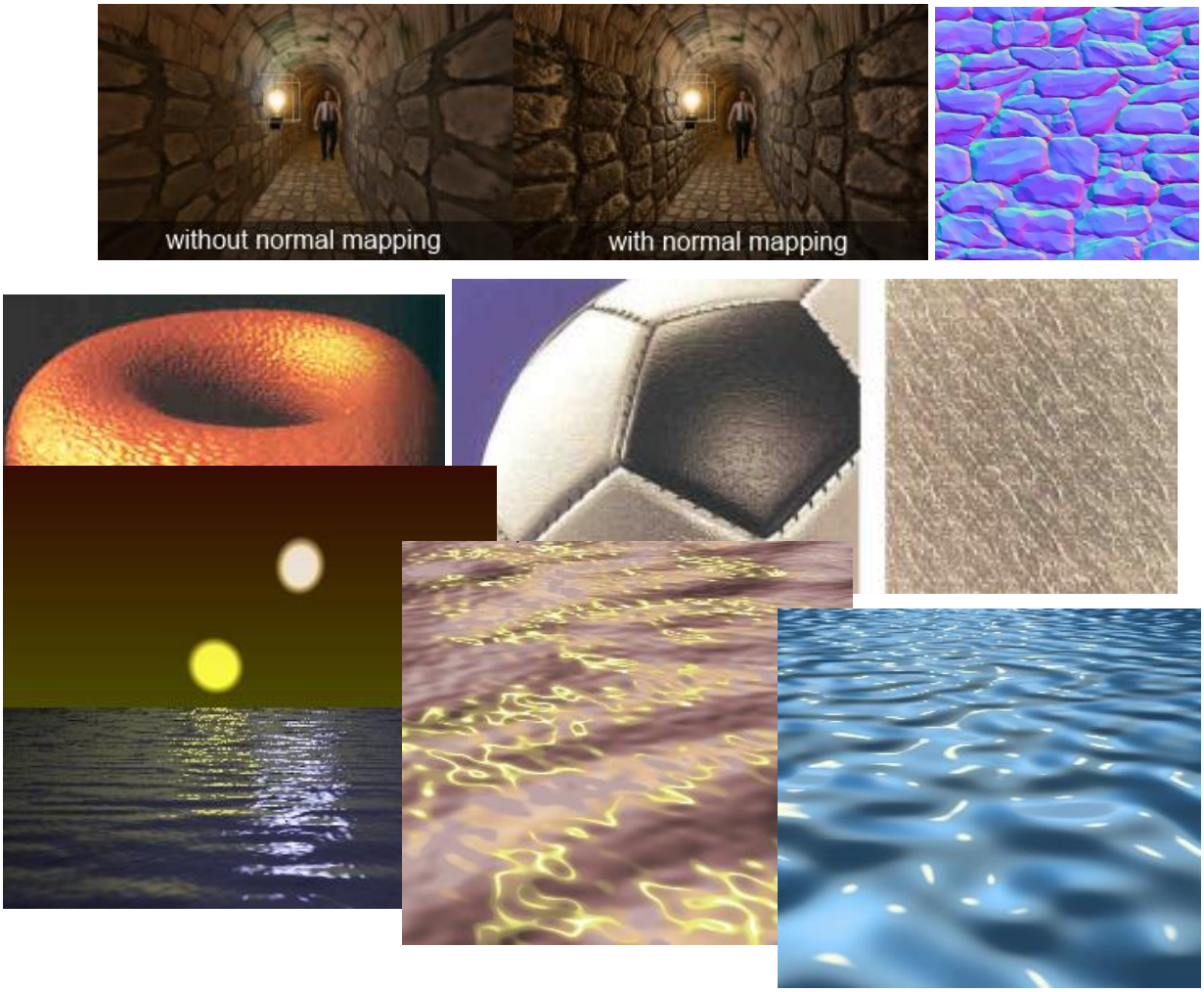
A visualização gráfica deste processo sobre uma superfície unidimensional é mostrada na seguinte figura.



Superfície original O(u)

bump map B(u)

Ajustando O(u) usando B(u)



Sem dúvida, o ponto chave do método é a definição da função *F(u,v)*, pois ela rege o comportamento da perturbação. O tipo de iluminação ou forma de amostragem é um processo a parte, mas que deve estar intimamente relacionado com o tipo de textura definido em *F(u,v)*.

Para texturas simples (com pouca complexidade), esta função pode ser definida usando-se polinômios, funções ou séries de *Fourier*. Porém, texturas mais complexas exigem um grande número de coeficientes, o que a torna complexa de ser modelada, exigindo maior tempo de processamento. Neste caso, o uso de texturas procedimentais baseadas em funções de ruído se mostra como uma boa solução para o problema.

Na representação de formas complexas, são usadas ***lookup tables*** bidimensionais. Elas são facilmente adaptáveis ao modelo porque podem ser indexadas pelos parâmetros *u* e *v* de *P*. A indexação de valores fracionários é definida por uma função de interpolação dos valores presentes na tabela, e pode ser uma interpolação bilinear ou de maior grau.

Pode-se atribuir a esta tabela padrões randômicos, regulares ou em forma de caracteres, por exemplo, dependendo do tipo de superfície que se deseja gerar. A geração dos valores desta tabela pode ser feita de maneira procedimental ou manual. Para padrões regulares, deve-se assegurar que exista continuidade nos pontos de união entre diferentes tabelas.

UFG/CS ***EXAME NACIONAL PARA INGRESSO NA PÓS-GRADUAÇÃO EM COMPUTAÇÃO*** ***2015***

QUESTÃO 70

MeshSmooth, Bump Map, Flat Shading são, respectivamente, tipos de:

(A) Modificador, Textura, Método de Renderização.

(B) Modificador, Método de Renderização, Textura.

(C) Textura, Método de Renderização, Modificador.

(D) Textura, Modificador, Método de Renderização.

(E) Método de Renderização, Textura, Modificador.

RAY TRACING

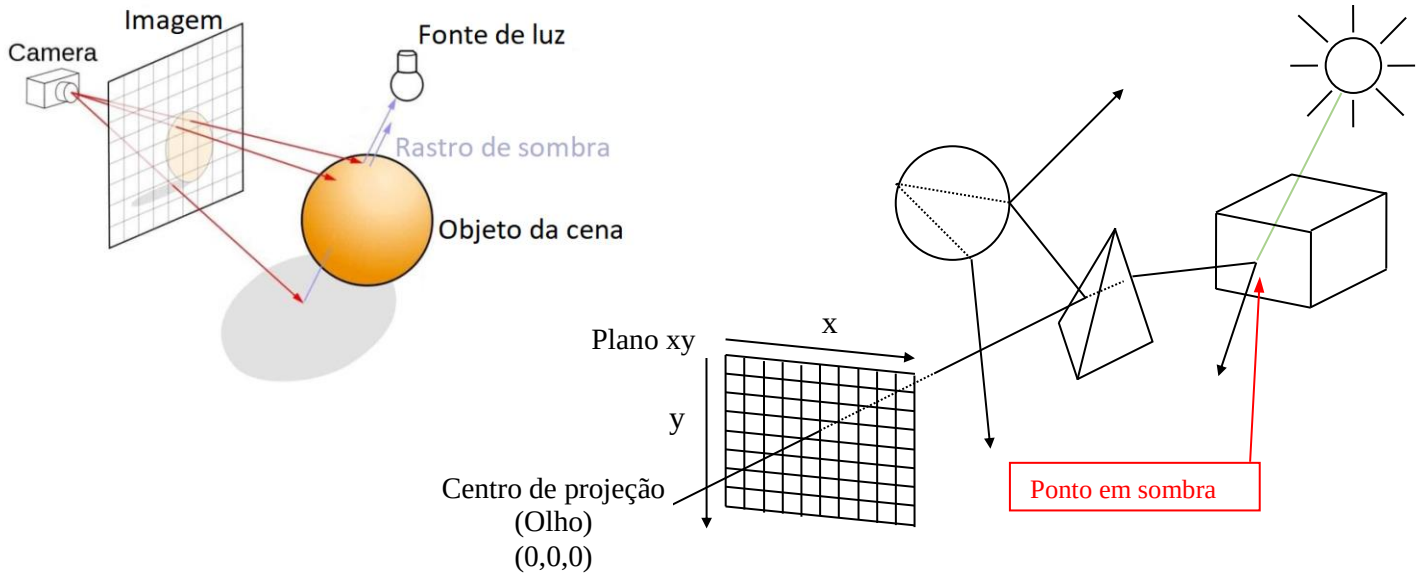
Este processo é atualmente o modelo mais completo de simulação de iluminação-reflexão em computação gráfica [WATT 93]. Ao contrário de uma simples reflexão da luz, neste método a luz refletida possui uma maior profundidade, ou seja, não é verificado somente a reflexão de objetos que interagem diretamente com a fonte de luz (reflexão local – algoritmo de ray-casting), mas também com a luz proveniente por reflexão de outros objetos que compõem o ambiente de iluminação.

A filosofia deste método é que um observador vê cada ponto de uma superfície como resultado da interação dela com luz proveniente de qualquer parte da cena. Como mostrado na seguinte figura, a luz que incide em um objeto pode ser proveniente de iluminação direta, luz transmitida, refletida, dentre outros. Isso é conhecido como **iluminação global**.

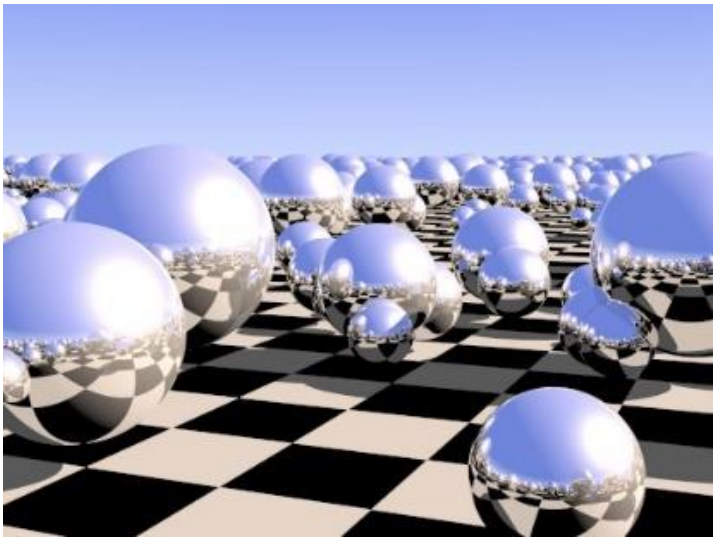
Dentre as vantagens deste método, em relação a outros mais, como o Z-buffer, está na capacidade de trabalhar com interação especular entre objetos. Este método tem a capacidade de solucionar o problema da iluminação global, através de combinação de quatro características:

- Remoção de superfícies escondidas
- Sombreamento devido a iluminação direta
- Sombreamento devido iluminação global
- Processamento da sombra

O funcionamento do Ray tracing é muito simples. Para cada pixel que compõe a imagem é enviado um raio (**cuja origem é o centro de projeção**). Durante seu percurso pode sofrer diversos tipos de interações com objetos que compõem a cena, e em cada uma dessas interações, é adicionada uma contribuição de luz que será devolvida ao pixel de onde o raio foi proveniente, como mostrado na seguinte figura.



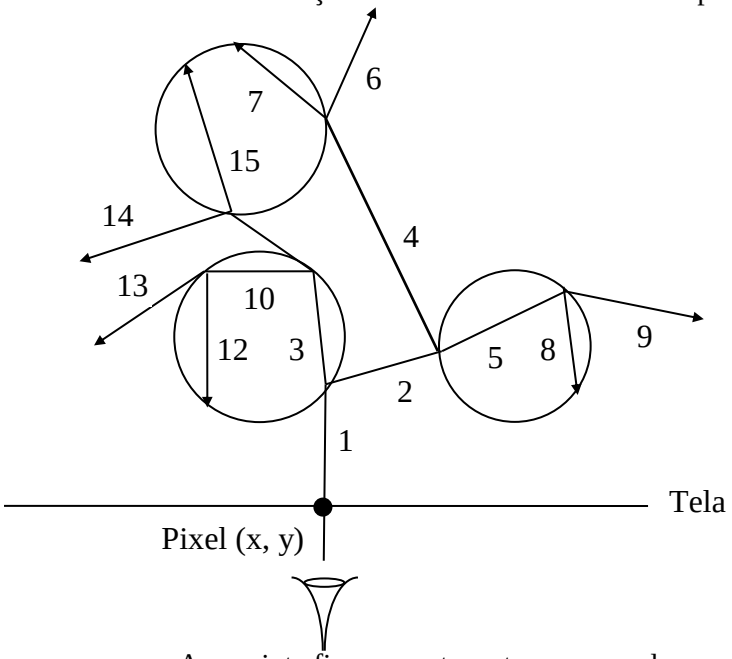
Interações de um raio através da cena. O raio em verde é usado para determinar se o ponto de colisão na cena tem visada até a fonte luminosa.



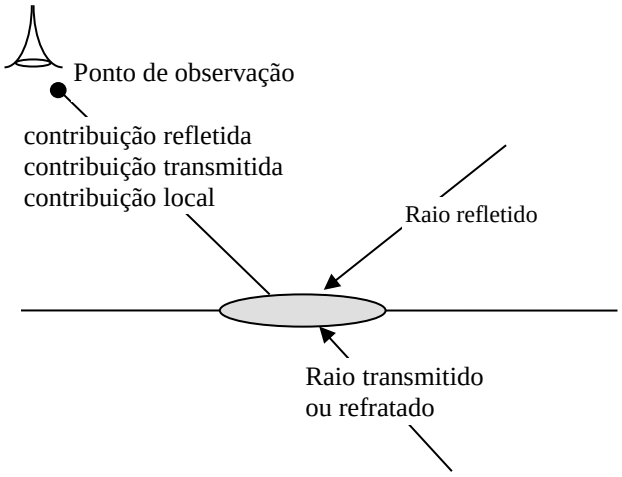
Quando cada raio é enviado, se mais de uma interseção com planos que compõe a cena for encontrado, será calculado o que for mais próximo de onde o raio partiu. Calculado o ponto de interseção, o raio é então refletido na direção do eixo de reflexão especular. Se este objeto for transparente, também será enviado o raio através da superfície, segundo o cálculo do ângulo de refração. Este processo é repetido até que um número n (profundidade de busca), definido pelo usuário, de reflexões ou refrações seja encontrado, ou até que o raio encontre uma superfície negra.

A implementação computacional do ray tracing é baseada praticamente em procedimentos recursivos, e é semelhante a uma busca em árvores binárias, pois em cada nó poderá haver duas ramificações (uma para raios refletidos e uma para os transmitidos).

Para exemplificar este método, consideremos a seguinte figura. Nela, existem 3 esferas semitransparentes, sobre as quais será traçado um raio. Para este caso, deverá ser calculado o valor da cor para o **ponto onde o raio 1 intercepta a primeira esfera** (mais próxima da tela). O cálculo da cor dependerá de três valores presentes no ponto de interseção: luz local, a refletida e a transmitida (por refração).
Neste ponto, a contribuição local pode ser dada pela iluminação Ambiente + Difusa + Especular. Para isso, deve-se também traçar um raio até a fonte luminosa para descobrir se o ponto está em sombra ou não.

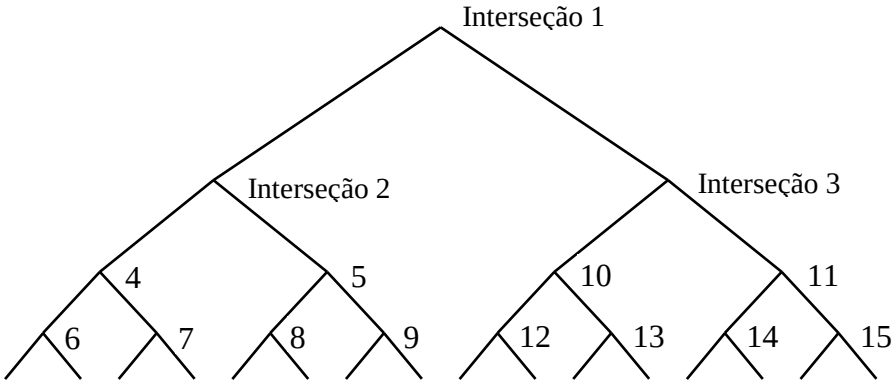


A seguinte figura mostra este processo de geração da intensidade da cor. A luz refletida para este ponto será calculada pelo envio de um raio na direção do eixo de reflexão. Quando este novo raio tocar um objeto, todo este processo se repete novamente e recursivamente. O cálculo da luz transmitida para cada ponto usa o mesmo processo. O que irá diferenciar é a direção do raio enviado. O cálculo dos valores da intensidade de luz refletida é dado pelo modelo Phong, descrito anteriormente. Este modelo também é usado para calcular a luz refratada de objetos transparentes, cuja fonte emissora esteja localizada atrás do mesmo.



Contribuições de luz em cada ponto de interseção

A árvore mostrada na seguinte figura ilustra todos os caminhos gerados para calcular o valor da intensidade do pixel. Em cada nó da árvore de derivação é aplicado o processo mostrado na figura anterior. Deve-se notar que esta árvore é apenas uma representação gráfica do processo de cálculo da intensidade de cor. Na verdade, ela não precisa ser gerada, pois neste tipo de aplicação, não existe caminhamento em árvore. O processo de recursividade realiza todo o trabalho sem que seja necessário gerar caminhos físicos. Com isso, a quantidade de memória necessária para desenvolver este processo recursivo é extremamente pequena. Como exemplo, para calcular a intensidade de cor para um cenário com n objetos transparentes, com uma profundidade p, o número de bytes utilizados será p * nbytes onde nbytes = num. de bytes em variáveis locais a função recursiva.



Árvore de derivação

Numericamente, o valor de nbytes, segundo o algoritmo descrito na sequência, terá um tamanho aproximado de 100 bytes. Utilizando-se uma profundidade 4, serão gastos aproximadamente 400 bytes. Deve-se notar que este valor representa somente a memória necessária para aplicar o processo de ray tracing. Não se está considerando a cena em si. O que será grande neste processo será o tempo de processamento necessário para processar todo o cenário.

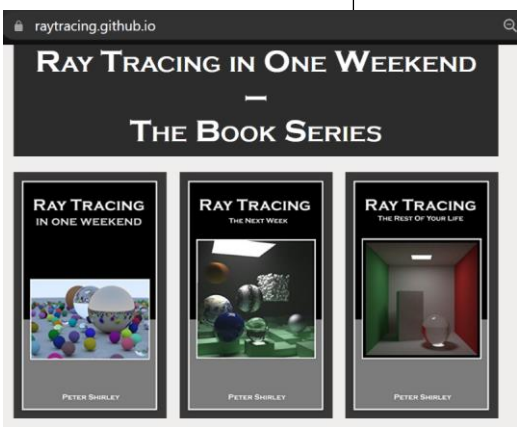
Este mesmo processo de busca recursiva sem o uso de árvores físicas é utilizado em funções heurísticas de busca, em problemas de inteligência artificial.

```
Cor RayTracing (inicio, direção: vetor; profundidade)
{
    vetor: pt_interseção, dir_reflexão, dir_transmissão
    cor: cor_local, cor_refletida, cor_transmitida
    Se profundidade > prof_max
    {
      retorna preto
    }
    pt_interseção = achar o ponto de interseção mais próximo da tela
    Se pt_interseção = NULL
    {
      retorna cor_fundo
    }
    cor_local = contribuição local (ambiente + difusa + especular + ... )
    dir_reflexão = calcular a direção do ângulo de reflexão
    cor_refletida = RayTracing(pt_interseção, dir_reflexão, prof+1)
    dir_transmissao = calcular a direção da luz transmitida
    cor_transmitida = RayTracing(pt_interseção, dir_transmissao, prof+1)
    cor = combinar_cores (cor_local, cor_refletida, cor_transmitida)
    retorna cor
}
```

O seguinte algoritmo, retirado do site https://raytracing.github.io/ ilustra a chamada do primeiro raio para cada pixel da tela. Todos os raios partem da posição (0,0,0).

```
void render(const std::vector<Sphere> &spheres, const std::vector<Light> &lights) {
    const int   width    = 1024;
    const int   height   = 768;
    const float fov      = M_PI/3.;
    std::vector<Vec3f> framebuffer(width*height);

    #pragma omp parallel for
    for (size_t j = 0; j<height; j++) { // actual rendering loop
        for (size_t i = 0; i<width; i++) {
            float dir_x =  (i + 0.5) -  width/2.;
            float dir_y = -(j + 0.5) + height/2.;
            float dir_z = -height/(2.*tan(fov/2.));
            framebuffer[i+j*width] = cast_ray(Vec3f(0,0,0),
                                              Vec3f(dir_x, dir_y, dir_z).normalize(),
                                              spheres, lights);
        }    }     }
```



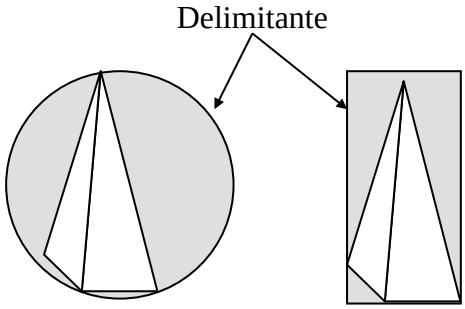
O seguinte algoritmo ilustra o cálculo de interseção raio-esfera. (https://raytracing.github.io/)

```
bool ray_intersect(const Vec3f &orig, const Vec3f &dir, float &t0) const {
        Vec3f L = center - orig;
        float tca = L*dir;
        float d2 = L*L - tca*tca;
        if (d2 > radius*radius) return false;
        float thc = sqrtf(radius*radius - d2);
        t0       = tca - thc;
        float t1 = tca + thc;
        if (t0 < 0) t0 = t1;
        if (t0 < 0) return false;
        return true;
    }
```

Otimizando cálculo de Interseções

Grande parte do processamento do ray tracing é usado na verificação dos pontos de interseção dos raios com os objetos que compõem o cenário. Para solucionar este problema, Clarck [CLAR 76] propôs **delimitar** (encapsular) cada objeto do cenário dentro de outro objeto de forma homogênea, como esferas, cilindros ou caixas, cuja interseção é mais fácil de ser calculada. Assim, inicialmente reduz-se o número de testes ao número de objetos que compõem a cena a ser processada.

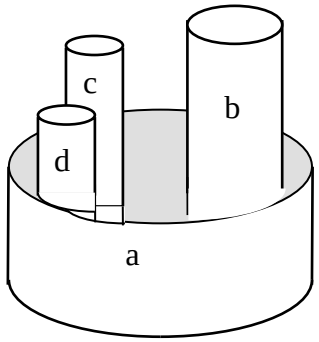
A escolha de qual envoltório irá encapsular cada objeto da cena depende basicamente da geometria da cena. Supondo que se deseje encapsular uma pirâmide, como mostrado na seguinte figura. Neste caso, o uso de uma caixa é bem mais apropriado que uma esfera, pois ela consegue delimitar o objeto com uma menor área. No caso da esfera envolvente poderá existir uma quantidade muito maior de raios que farão a interseção com o delimitante, mas que não fazem interseção com o objeto real. Em caso de uma folha de árvore, pode-se utilizar uma caixa como volume envolvente.

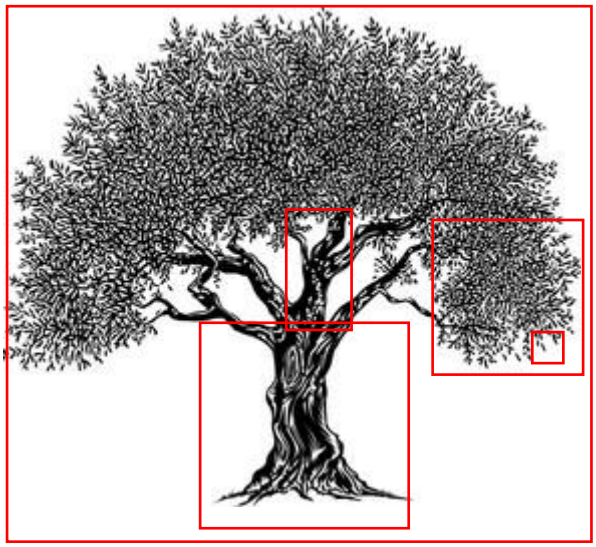


Deve-se notar que o processo de teste de interseção pelo delimitante tem apenas função de reduzir o campo de busca. Ele faz uma aproximação que as vezes pode ser pouco precisa. Entretanto, tendo-se eliminado parte do cenário, necessita-se apenas testar a interseção dos raios com cada face do objeto interno ao delimitante, que neste caso são os 5 lados da pirâmide.

O aumento da eficiência do método depende basicamente da correta escolha deste delimitante. Esferas são uma escolha popular pela sua simplicidade de testar a interseção. Deve-se observar a relação de proporção que existe entre a área do objeto encapsulado e a área do objeto delimitante. Quanto maior for esta razão, maior será o custo de processamento.

Um aprimoramento para o método de encapsulamento de objetos foi sugerido por Whitted [WHIT 80], e propõe definir uma **hierarquia estruturada** de objetos na cena. Objetos que estão próximos uns dos outros podem ser unidos para formar conjuntos ("*clusters"*), para então serem encapsulados por uma esfera, um cubo ou um cilindro, recursivamente. Desta forma, se o raio não colidir com o envoltório maior, também não irá colidir com os envoltórios e objetos dentro deste envoltório maior.





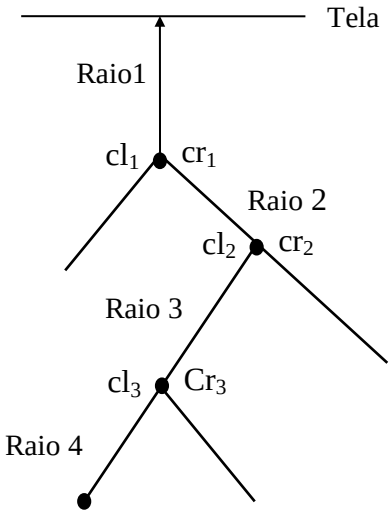
Encapsulamento de forma hierárquica de objetos próximos

Definindo contribuição da iluminação em cada ramo

Ao se aplicar o ray tracing é necessário especificar a profundidade máxima na qual os raios serão propagados. O processo recursivo pára quando esta profundidade é alcançada. Este valor pode ser determinado pelo tipo de cenário e pela qualidade da imagem a ser gerada. Uma cena contendo superfícies com um alto grau de reflexão de luz e transparência exigirá uma maior profundidade em relação a uma cena com objetos que possuem pouca reflexão (opacos) e nenhuma transparência. É sugerido um método que, para cada ocasião, verifique a profundidade a ser usada. Deverão também ser consideradas as propriedades dos materiais por onde os raios estão incidindo. Portanto, para cada caso, o valor da profundidade poderá ser de 1 até um máximo permitido.

Quando um raio é refletido em uma superfície, ele é atenuado pelo coeficiente de reflexão especular da superfície. Quando ocorre uma refração, o raio é atenuado pelo coeficiente de transmissão global da superfície.

O raio sendo analisado em um ponto da tela é o resultado da contribuição de todos os raios provenientes de outras interseções com objetos quando o raio era enviado para a mesma. Isto é ilustrado na seguinte figura, onde qualquer contribuição do raio 4 para a intensidade da cor no nível superior (tela) é atenuado por um produto dos coeficientes **cl** (coeficiente de reflexão) e **cr** (coeficiente de refração), e é fator de $cr_1cl_2cl_3$.



Fatores de atenuação de reflexão e refração

Se esta contribuição for menor que um limite, não faz sentido verificar a contribuição de raios com profundidade maior que 4. Logo, o processo recursivo pode ser finalizado. Isso leva a um aumento de velocidade, pois "caminhos" que não gerariam nenhuma contribuição ao pixel em questão são eliminados.

UFG/CS **EXAME NACIONAL PARA INGRESSO NA PÓS-GRADUAÇÃO EM COMPUTAÇÃO** **2015**

QUESTÃO 57

Simular a propagação da luz no ambiente, avaliando a sua interação com os objetos que o compõem e considerando a interação da luz com as suas superfícies, é o objetivo da técnica do algoritmo

(A) *Cohen-Sutherland*

(B) *Bresenham*

(C) Boundary-Fill

(D) *Sutherland Hodgman*

(E) *Ray Tracing*

ENADE 2014
EXAME NACIONAL DE DESEMPENHO DOS ESTUDANTES

QUESTÃO 19

O algoritmo de traçado de raios (*ray-tracing*) é considerado um marco no desenvolvimento de técnicas de computação gráfica para a geração de imagens realistas.

Imagem 1

Disponível em: <http://www.wikipedia.org>. Acesso em: 18 jun. 2014.

Imagem 2

Disponível em: <http://www.colegioweb.com.br>. Acesso em: 18 jun. 2014.

Imagem 3

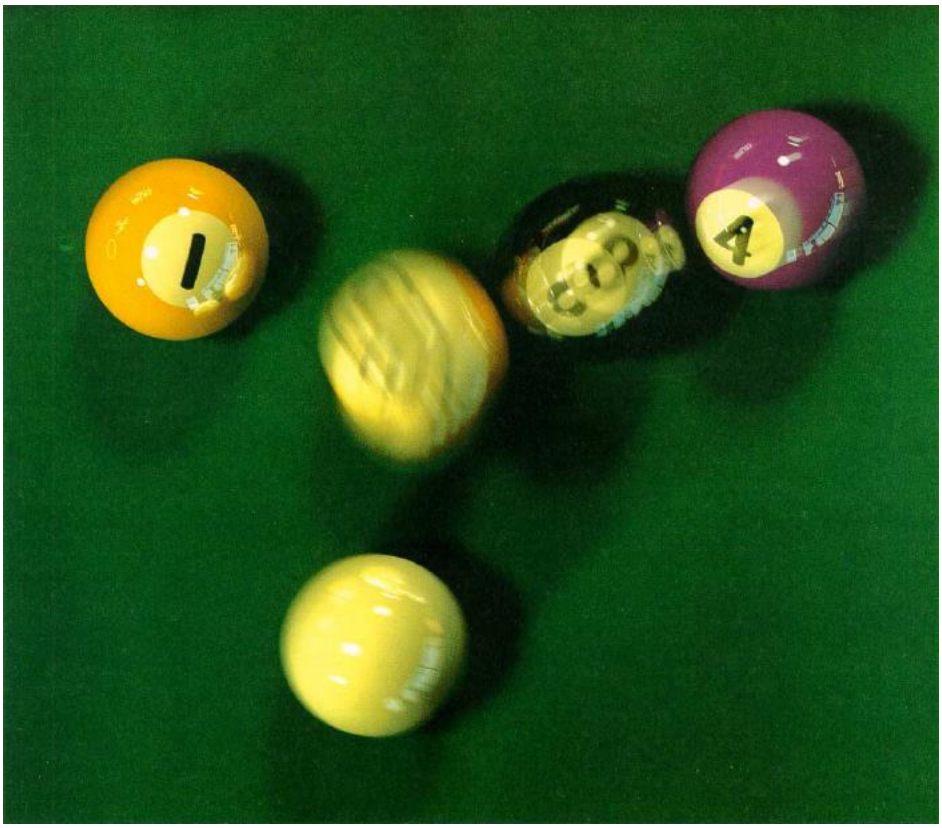
Disponível em: <http://www.cgw.com>. Acesso em: 18 jun. 2014.

A partir da análise das imagens apresentadas, conclui-se que a técnica de traçado de raios foi utilizada para a geração

A) apenas da imagem 1.
B) apenas da imagem 2.
C) apenas das imagens 1 e 3.
D) apenas das imagens 2 e 3.
E) das imagens 1, 2 e 3.

Ray tracing distribuído

Outro aprimoramento em relação ao *Ray tracing* é o ***Ray tracing* distribuído** (*distributed Ray tracing*) [COOK 84]. Esta técnica visa reduzir os efeitos de alias, bem como transições bruscas de cor em sombras, reflexões e refrações (*sharp shadows*). A técnica consiste em enviar vários raios por pixel (super amostragem estocástica) **segundo alguma função**. Se os raios forem lançados em tempos diferentes sob objetos em movimento, pode-se também obter um efeito chamado *motion blur*, como mostrado na seguinte figura intitulada 1984.



Segue resumo do artigo original [COOK 84]:

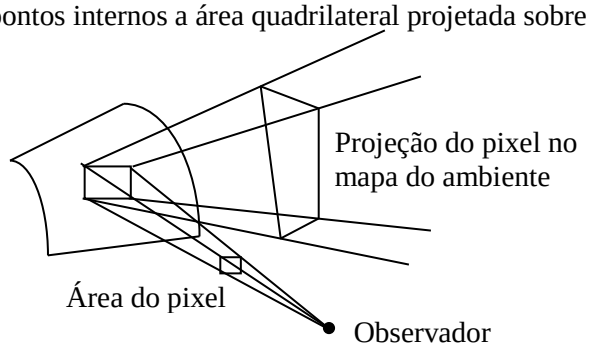
"Ray tracing is one of the most elegant techniques in computer graphics. Many phenomena that are difficult or impossible with other techniques are simple with ray tracing, including shadows, reflections, and refracted light. Ray directions, however, have been determined precisely, and this has limited the capabilities of ray tracing. By distributing the directions of the rays according to the analytic function they sample, ray tracing can incorporate fuzzy phenomena. This provides correct and easy solutions to some previously unsolved or partially solved problems, including ***motion blur****,* ***depth of field****,* ***penumbras****,* ***translucency****, and* ***fuzzy reflections****. Motion blur and depth of field calculations can be integrated with the visible surface calculations, avoiding the problems found in previous methods."*

ENVIRONMENT MAPPING

Esta técnica diferencia das até agora apresentadas pois mais se assemelha a um processo de iluminação do que um processo de textura. Ela pode ser vista como uma simplificação do processo de *Ray Tracing*, pois a textura que o objeto irá receber depende de onde o observador se encontra e do ambiente que o cerca, ou seja, é projetado sobre o objeto a reflexão dos objetos que estão a sua volta. Por isso se chama "*environment mapping* - mapeamento do ambiente". Pode-se também usar uma superfície auxiliar para armazenar o *environment map* (mapa de bits do ambiente).

Ao contrário das outras técnicas, a textura neste caso não fica fixa ao objeto, pois mudando a posição do observador, também irá mudar a reflexão da luz e, consequentemente, a imagem refletida.

O processo de mapeamento é mostrado na seguinte figura. São projetados raios (em número de 4) do observador sobre a área de cada pixel do objeto (Como no *Ray Tracing*, porém neste caso são considerados apenas duas iterações da luz, ou seja, aplica-se o *Ray Tracing* com profundidade 2), que são refletidos e projetados sobre o *environment map*. A intensidade do pixel será determinada pela média de intensidade dos pontos internos a área quadrilateral projetada sobre o environment map.



Referências Bibliográficas

[BIER 86] Bier, E., Sloan, K. R., *Two-part Texture Mapping*, IEEE Computer Graphics and Applications, 1986

[BLIN 76] BLIN, J. F. Texture and reflection in computer-generated images. **Communications of the ACM**, v.19, n.10, Oct. 1976, p.542-547.

[BLIN 78] BLIN, J. F. Simulation of wrinkled surfaces. **Computer Graphics**, v.12, n.3, Aug. 1978, p.286-292.

[BRET 92] BRET, M. **Image synthesis**. Netherlands: Kluwer Academic Pub, 1992, 289p.

[CATM 74] CATMULL, E. **A subdivision algorithm for computer display of curved surfaces**. Ph.D. Thesis - University of Utah, Salt Lake City, Utah, Dec. 1974.

[CLAR 76] CLARK, J. H. Hierarchical geometric models for visible surface algorithms. **Communication of the ACM**, v.19, n.10, 1976, p.547-554.

[COOK 84] COOK, R. L, Porter, T., Carpenter, L. Distributed ray tracing. **Computer Graphics**, 18 (3), 137, 45.

[COOK 86] COOK, R. L. Stochastic sampling in computer graphics. **ACM Transactions on Graphics**, v.5, n.1, Jan. 1986, p.51-71.

[COOK 87] COOK, R. L, CARPENTER, L., CATMULL, E. The REYES image rendering architecture. **Computer Graphics**, v.21, n.4, Jul. 1987, p.95-102.

[EBER 98] Ebert, D. et al. **Texturing & modeling**. 2.th. San Diego: AP Professional, 1998, 415p.

[FUJI 86] FUJIMOTO, A., TANAKA, T., IWATA, K. ARTS: acelerated ray-tracing system. **IEEE Computer Graphics and Applications**, v.6, n.4, Apr. 1986, p.16-26.

[GLAS 95] GLASSNER, A. S. **Principles of digital image synthesis**. San Francisco: Morgan Kaufmann, USA, 1995.

[HECK 86] HECKBERT, P. S. Survey of texture mapping. **IEEE Computer Graphics and Applications**, v.6, n.11, Nov. 1986, p.56-67.

[HEAR 97] HEARN, D., BAKER, M. P. **Computer graphics C version**. Local: Prentice Hall, 1997, 652p.

[JOY 87] JOY, K. I. et al. **Tutorial: computer graphics: image synthesis**. Local: ACM Press, 1987, 368p.

[OPPE 89] OPPENHEIM, A. V., SCHAFER, R. W. **Discrete-time signal processing**. New Jersey: Prentice-Hall, 1989.

[PHON 75] PHONG, B. Illumination for comupter-generated pictures. **Communications of the ACM**, v.18, n.3, Jun. 1975, p.311-317.

[PEAC 86] PEACHEY, D. R. Modeling waves and surfaces. **Computer Graphics**, v.20, n.4, Aug. 1986, p.65-74.

[PERL 85] PERLIN, K. An image synthesizer. **Computer Graphics**, v.19, n.3, Jul. 1985, p.287-296.

[TIP 91] Tipler, P, Física para Cientistas e Engenheiros - Volume 4 Ótica e Física Moderna, Ed. Guanabara Koogan, 1991.

[YELL 83] YELLOTT, J. I. Spectral consequences of photoreceptors sampling in the rhesus retina. **Science**. v.221, Jul. 1983, p.382-385.

[WATT 93] WATT, A. **3D computer graphics**. England: Addison-Wesley, 1993, 500p.

[WHIT 80] WHITTED, T. An improved illumination model for shaded display. **Communications of the ACM**, v.23, n.6, Jun. 1980, p.343-349.