Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

Mapeamento de Texturas

André Tavares da Silva

andre.silva@udesc.br

Baseado no material de Rosalee Wolfe (Siggraph Education)

Pesquisa recente

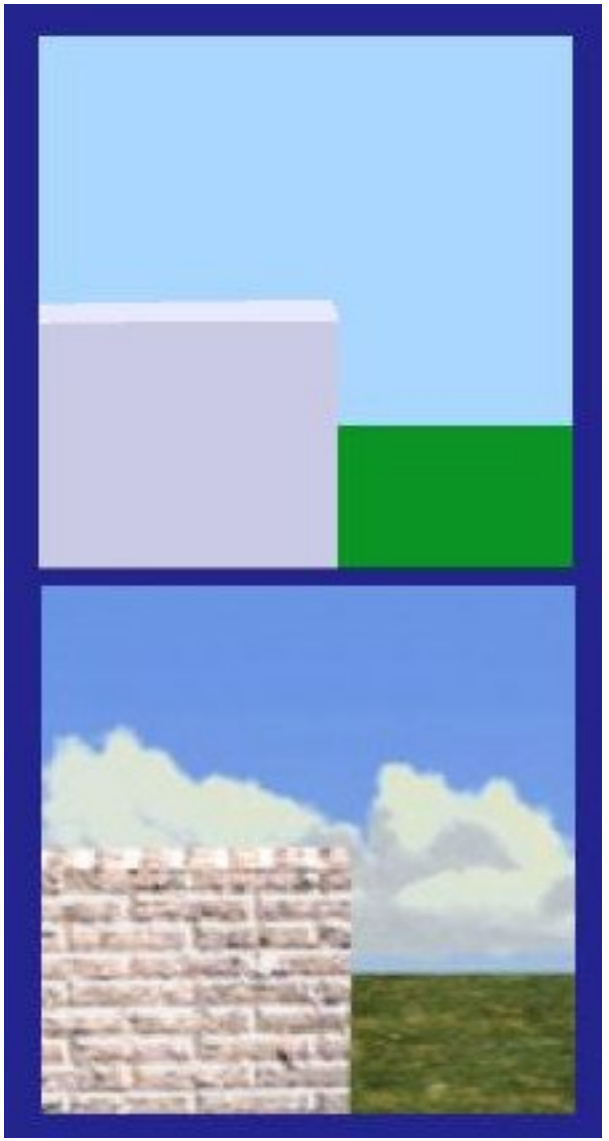
https://www.youtube.com/watch?v=KHT82-KYhRw

https://www.youtube.com/watch?v=YW9Gwn6cJ9o

https://www.youtube.com/watch?v=WL0UJPXm-do

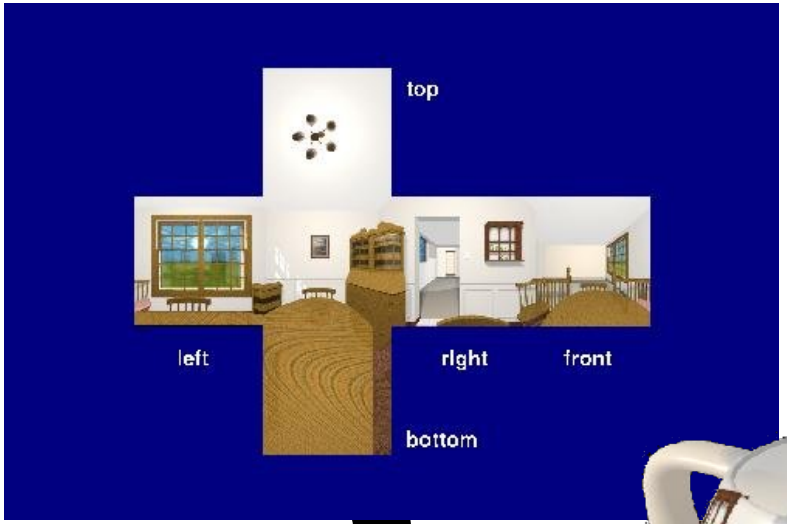
Mapeamento de textura

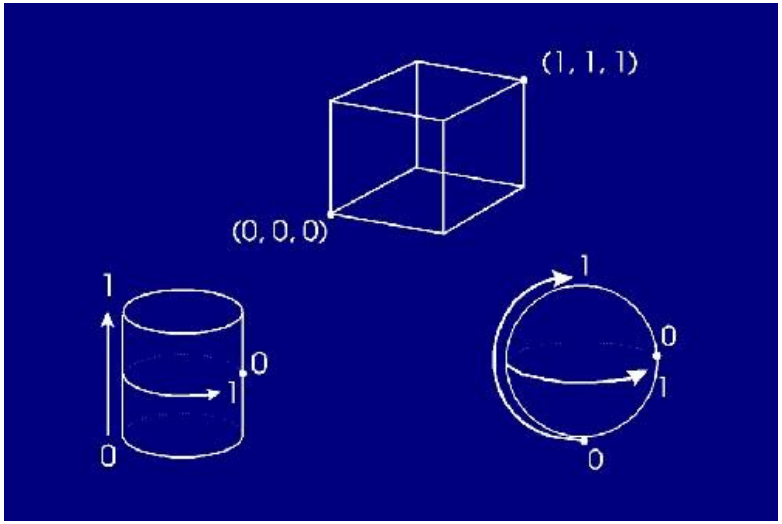
Ao criar detalhe da imagem, é mais barato empregar técnicas de mapeamento de textura do que usar milhares (milhões, bilhões,...) de pequenos polígonos. A imagem à direita retrata uma parede de tijolos, um gramado e o céu. Na realidade, a parede foi modelada como um sólido retangular, e tanto gramado quanto céu foram criados a partir retângulos. A imagem inteira contém oito polígonos. Imagine o número de polígonos que seria necessário para modelar as folhas de grama no gramado! O mapeamento de textura cria a aparência de relva, sem o custo de renderizar milhares de polígonos.



Mapeamento de textura

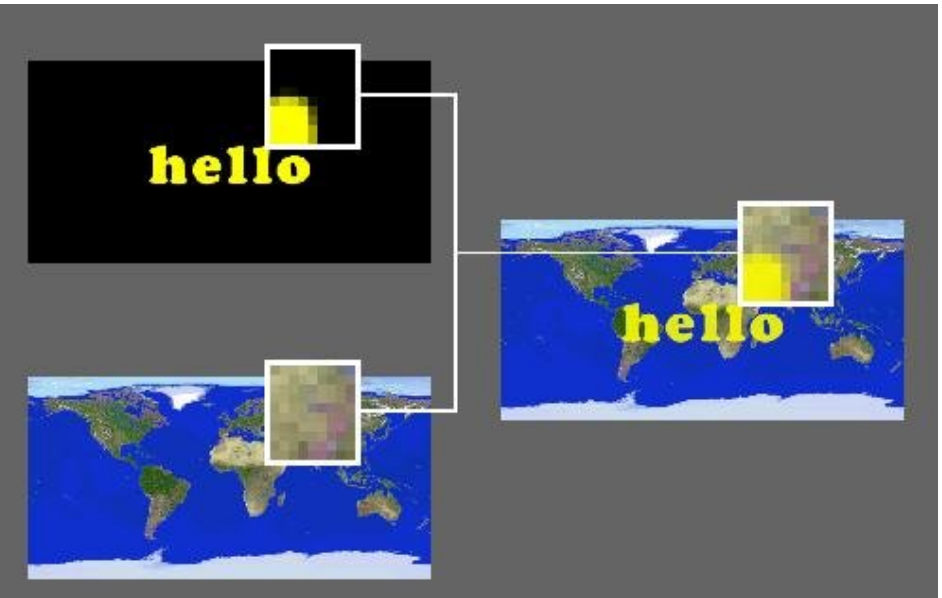
- Dependendo da situação, nós podemos precisar fazer o mapeamento de textura em um objeto por uma caixa, um cilindro ou uma esfera.





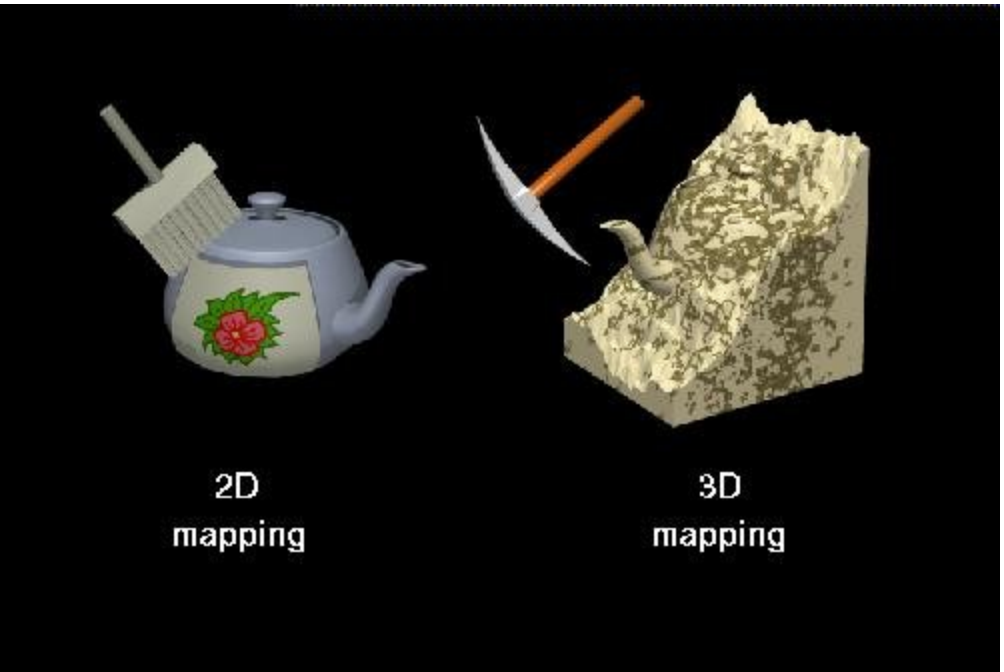
Mescla de texturas

Podemos colocar uma camada de textura um em cima de outra, utilizando uma técnica semelhante à *chroma key*. Na imagem abaixo colocamos a palavra "Hello" em cima de um mapa do mundo. O fundo preto da imagem "Hello" será tratado como transparente. Para criar um pixel na imagem final, encontramos as cores nos locais de *pixels* correspondentes nas duas imagens de entrada e depois combinamos ambas.



Mapeamento 3D

O mapeamento de textura pode ser dividido em técnicas bidimensionais e tridimensionais. As técnicas bidimensionais colocam uma imagem bidimensional (plano) sobre um objeto usando métodos semelhantes para colar papel de parede em um objeto. Técnicas tridimensionais são análogas a esculpir o objeto (sólido) a partir de um bloco de mármore.



Textura Procedural

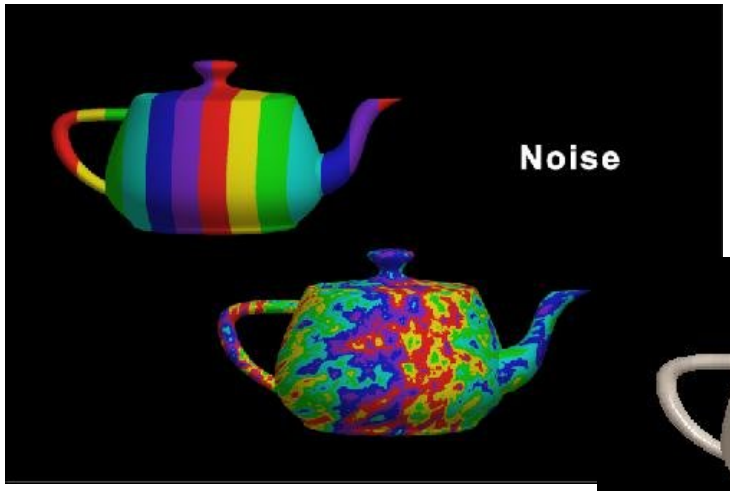
Funções processuais não só pode determinar a cor de um objeto, mas a sua geometria também. Funções de densidade de volume descreve a geometria de gases e são semelhantes aos procedimentos de texturização sólidas em que é tomado um ponto no espaço tridimensional como entrada e é retornado um valor. Em vez de retornar uma cor, retorna a densidade.

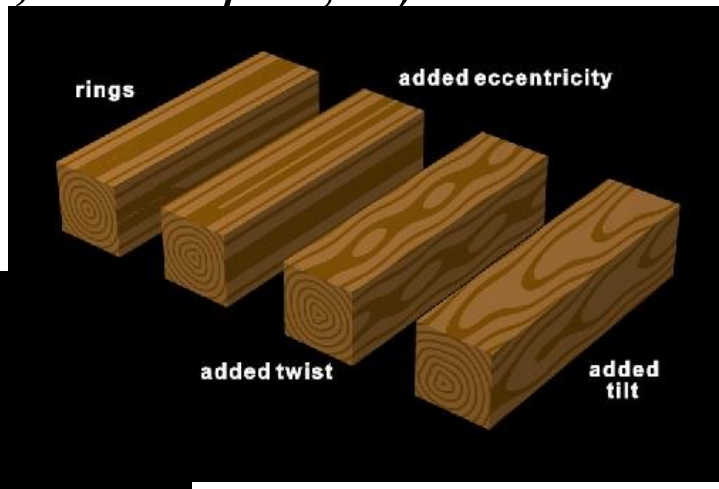
metaball criado por David Ebert

filme Getting Into Art de David Ebert

Ruído

Padrões regulares não são tão interessantes como os padrões com alguma aleatoriedade. Para o mapeamento de textura, essa aleatoriedade é produzida por uma função de ruído (*known range, stationary, band limited, isotropic*,...).

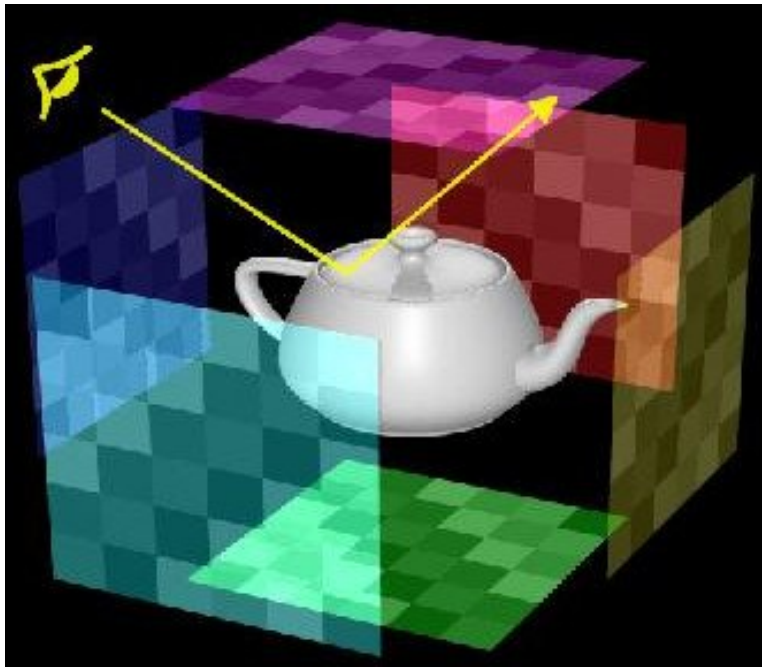




Environment Mapping

- É uma forma barata de criar reflexões (Blinn e Newell, 1976). Embora seja fácil de criar reflexos com um *Ray Tracing*, ele ainda é muito caro para animações longas.
- Adicionando o recurso de *Environment Mapping* para um *rendering* baseado em z-buffer irá criar reflexões que são aceitáveis em uma série de situações.
- É uma técnica de mapeamento de textura bidimensional que utiliza uma forma do mapa de uma caixa e um mapa de parâmetros de um raio reflexão.

Environment Mapping



EM x RT

Shadow Map

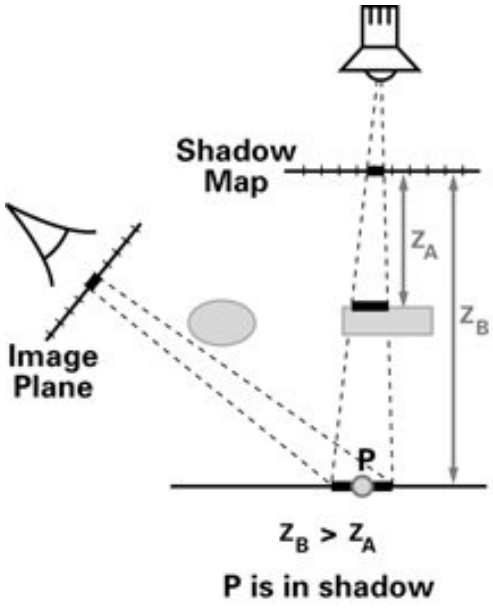
- É uma técnica para criar sombras em imagens 3D crada por Lance Williams em 1978, em um *paper* intitulado "Casting curved shadows on curved surfaces".
- Desde então vem sendo usada em cenas pré-renderizadas ou em diversos jogos e simulações em tempo real.
- As Sombras são criadas testando quando um pixel está visível a partir fonte de luz comparando o *z-buffer* com uma *depth image* (*shadow map*) da visão da fonte de luz, guardada na forma de textura.

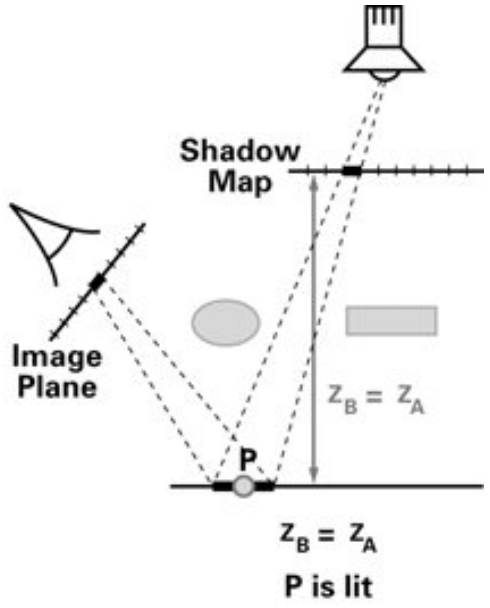
Shadow Map

- Primeiro cria-se uma imagem (*shadow map*) vista da posição da luz que se deseja projetar a sombra. Ao contrário de uma imagem qualquer, cujos *pixels* armazenam cores, a imagem gerada armazena as distâncias dos respectivos *pixels* em relação à luz (semelhante ao *z-buffer*).
- Depois de criada essa imagem, a cena vista da posição da câmera é renderizada. É feita uma comparação da distancia do *pixel* com a luz e o valor armazenado no *shadow map*.

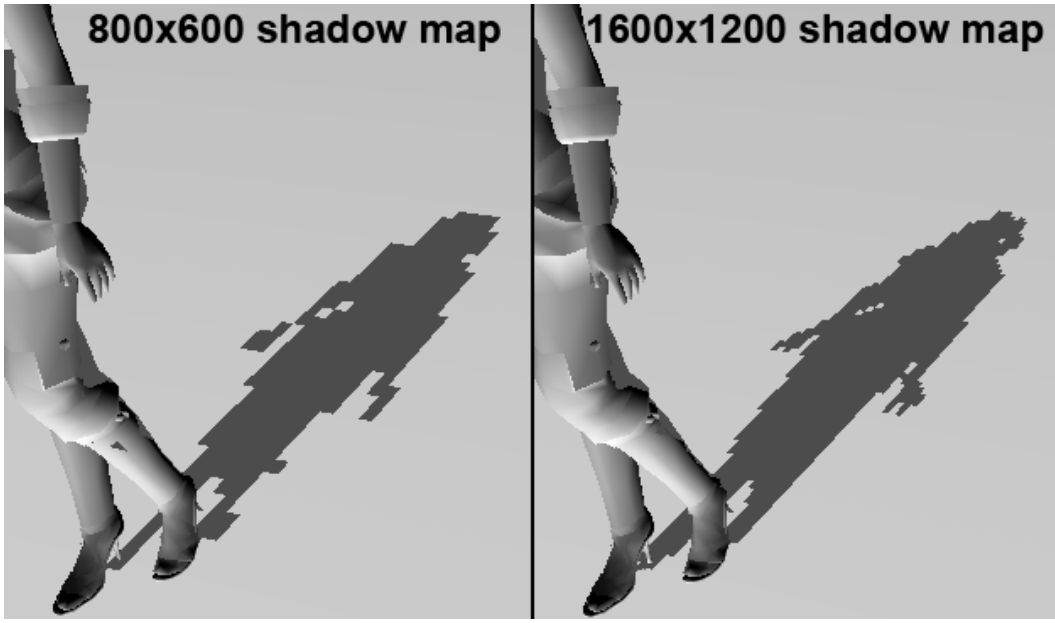
Shadow Map

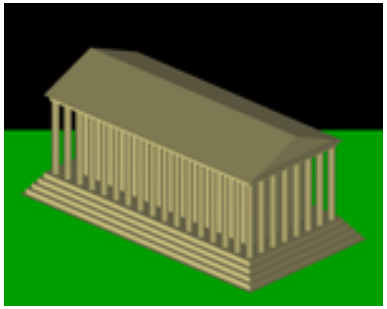
- Se a distância *pixel*-luz for maior que o valor armazenado no *shadow map* então o pixel está sombreado, pois existe um outro pixel com uma distância menor em relação a luz na frente dele projetando uma sombra; caso contrário, o pixel estará iluminado.



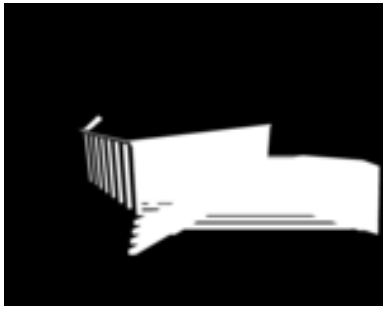


Shadow Map

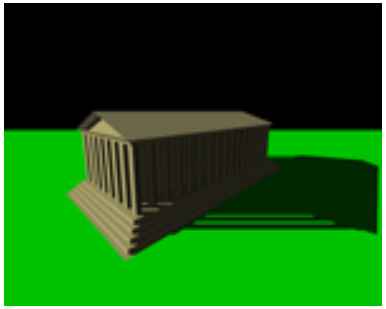




-



=



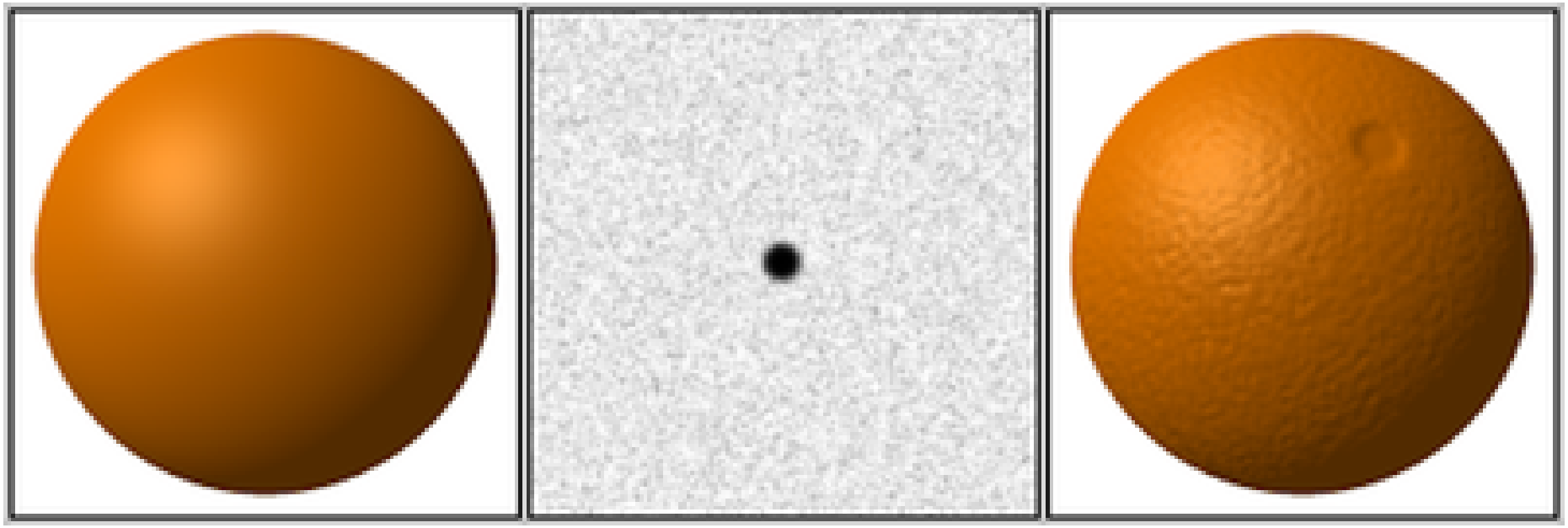
Bump Mapping

- É uma técnica criada por James Blinn em 1978 onde pega-se cada pixel do objeto que está sendo renderizado e é aplicada uma perturbação em sua superfície normal, baseada num mapa de altura, variando a intensidade de luz "refletida" por este pixel.
- A iluminação é aplicada após os cálculos dando a cada pixel seu respectivo brilho.
- O resultado é uma superfície renderizada com mais detalhes e imperfeições lembrando o mundo real.

Bump Mapping

- Esse mapa de altura é definido por uma imagem em escala de cinza onde preto é ausência de relevo e branco é relevo total.
- Pode ser criado em qualquer programa de manipulação de imagem.
- O defeito desse tipo de mapa é que quanto mais próximo a câmera da superfície, mais perceptível que aqui é falso, que o relevo não existe, porque *bump map* só simula altura.

Bump Mapping



Normal Mapping

- É uma variante da técnica *Bump Mapping*, desenvolvido por Krishnamurthy e Levoy em 1996.
- Ele é usado para adicionar detalhes sem usar mais polígonos, mas aumentando consideravelmente a aparência e detalhes de um modelo de baixo polígono.
- Ele é gerando por um mapa de normais, geralmente armazenados como imagens regulares RGB, em que os componentes RGB correspondem aos eixos X, Y, Z, respectivamente.

Normal Mapping

- Para calcular a iluminação lambertiana (difusa) de uma superfície, é combinado o vetor unitário a partir do ponto de sombreamento para a fonte de luz com o vetor unitário do mapa de normais. O resultado é a variação da intensidade da luz na superfície.
- O primeiro console de jogos a usar *normal mapping* foi o Sega Dreamcast. O Microsoft Xbox foi o primeiro console a utilizar amplamente o efeito em jogos de varejo. Jogos para o Xbox 360, PlayStation 3 e os atuais dependem fortemente de *normal mapping*.

Normal Mapping

- Para encontrar a perturbação na normal, a tangente deve ser calculada corretamente.
- Na maioria das vezes a normal é perturbada no *fragment shader* depois de se aplicar as matrizes de modelo (GL_MODELVIEW) e de visualização (GL_PROJECTION).

Normal Map

- Tipicamente, a geometria fornece uma normal e uma tangente. A tangente é parte do plano tangente e podem ser transformados de forma simples com a parte linear da matriz (3x3 superior).

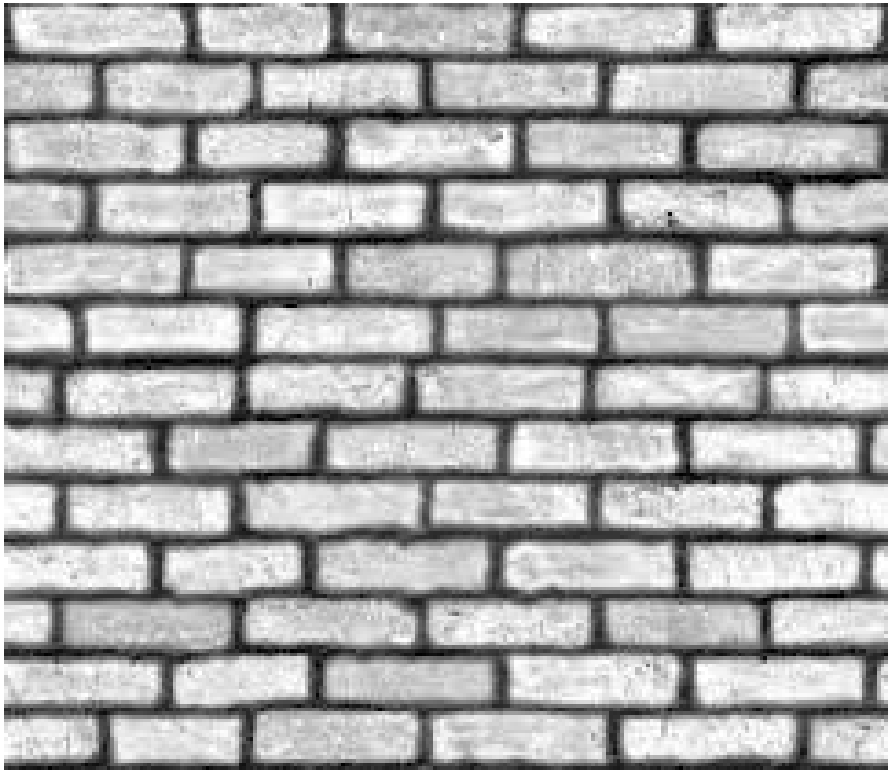
$$t' = t \times M_{3x3} \times V_{3x3}$$

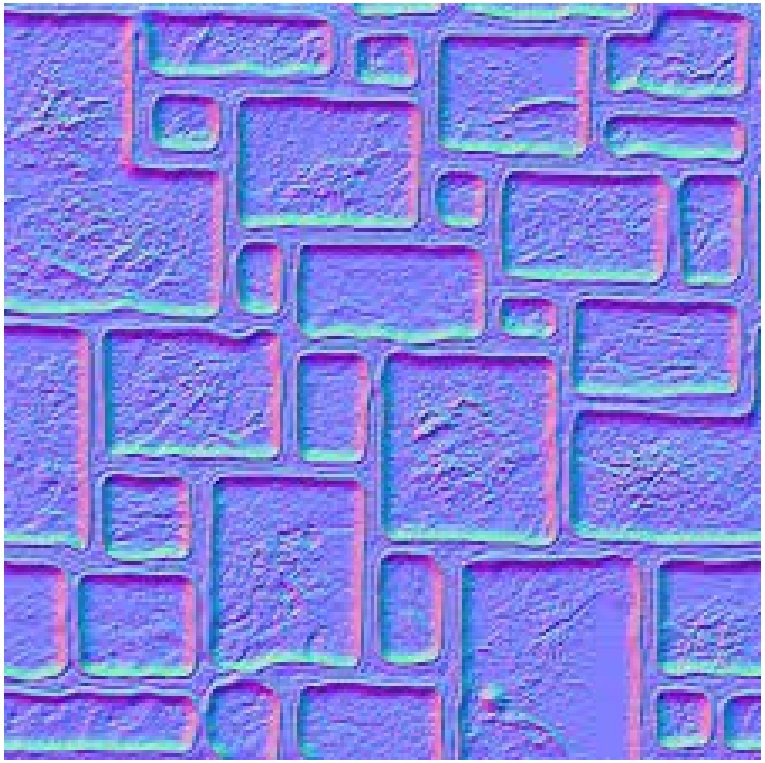
$$b' = b \times M_{3x3} \times V_{3x3}$$

- No entanto, as normais precisam ser transformadas pela inversa da transposta.

$$n' = n \times (M_{3x3} \times V_{3x3})^{-1T} = n \times M_{3x3}^{-1T} \times V_{3x3}^{-1T}$$

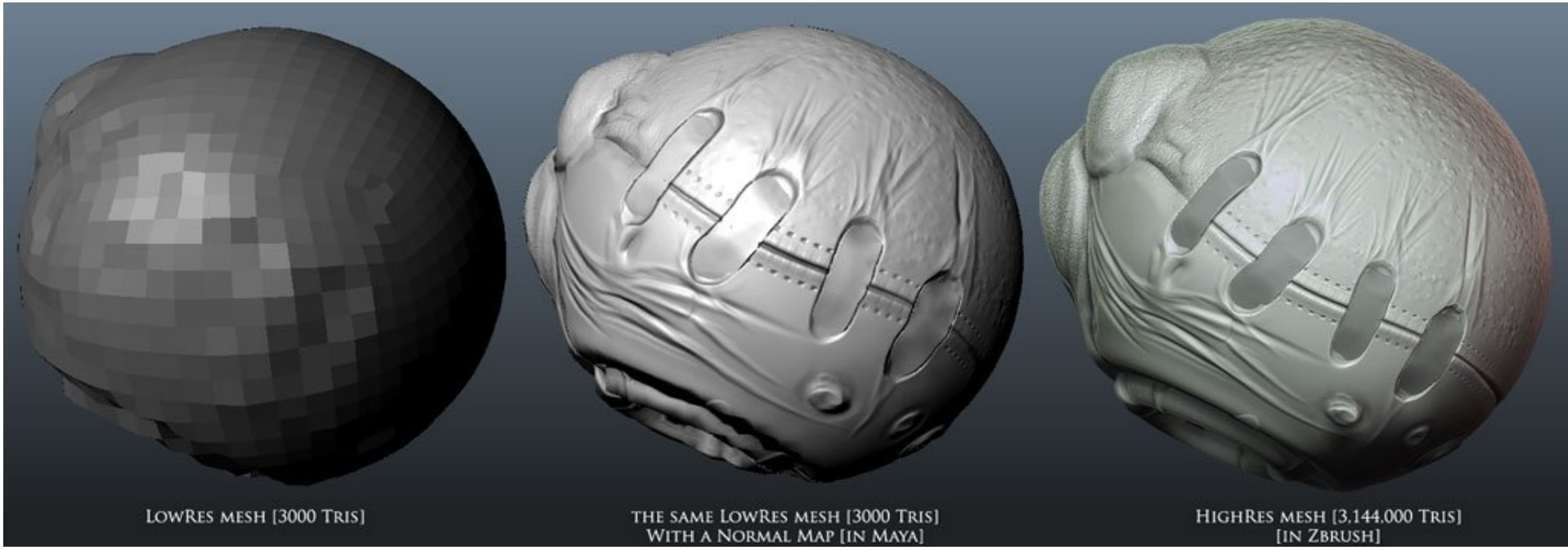
bump map x *normal map*





Normal Map

Normal Map


LowRes mesh [3000 Tris]
the same LowRes mesh [3000 Tris]
With a Normal Map [in Maya]
HighRes mesh [3,144,000 Tris]
[in ZBrush]

Displacement Mapping

- Foi introduzido por Tomomichi Kaneko et al. em 2001, sendo também chamado de *Parallax Mapping*. É um aprimoramento do *Bump Mapping* e *Normal Mapping* aplicadas a texturas em aplicações de renderização 3D, como vídeo games.
- Para o usuário final, isto significa que as texturas, tais como muros de pedra terão profundidade mais aparente e, consequentemente, um maior realismo com pouca influência no desempenho da simulação.

Displacement Mapping

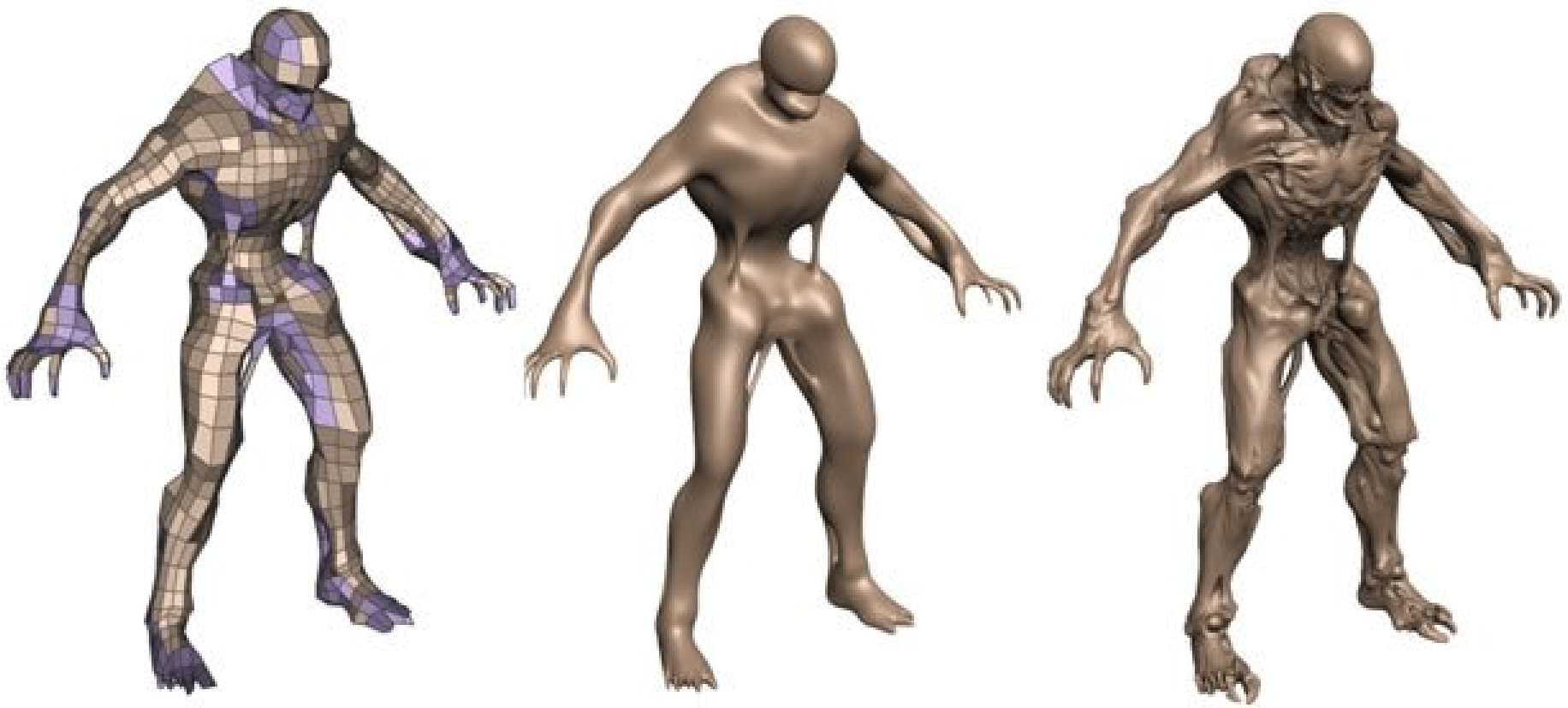
- É implementado através do deslocamento das coordenadas de textura num ponto no polígono renderizado em função do ângulo de visão no espaço tangente (ângulo em relação à superfície normal) e o valor do mapa de altura nesse ponto.
- Nos ângulos de visualização mais íngremes, as coordenadas de textura são mais deslocados, dando a ilusão de profundidade, devido aos efeitos de paralaxe como a mudança de ponto de vista.

Displacement Mapping

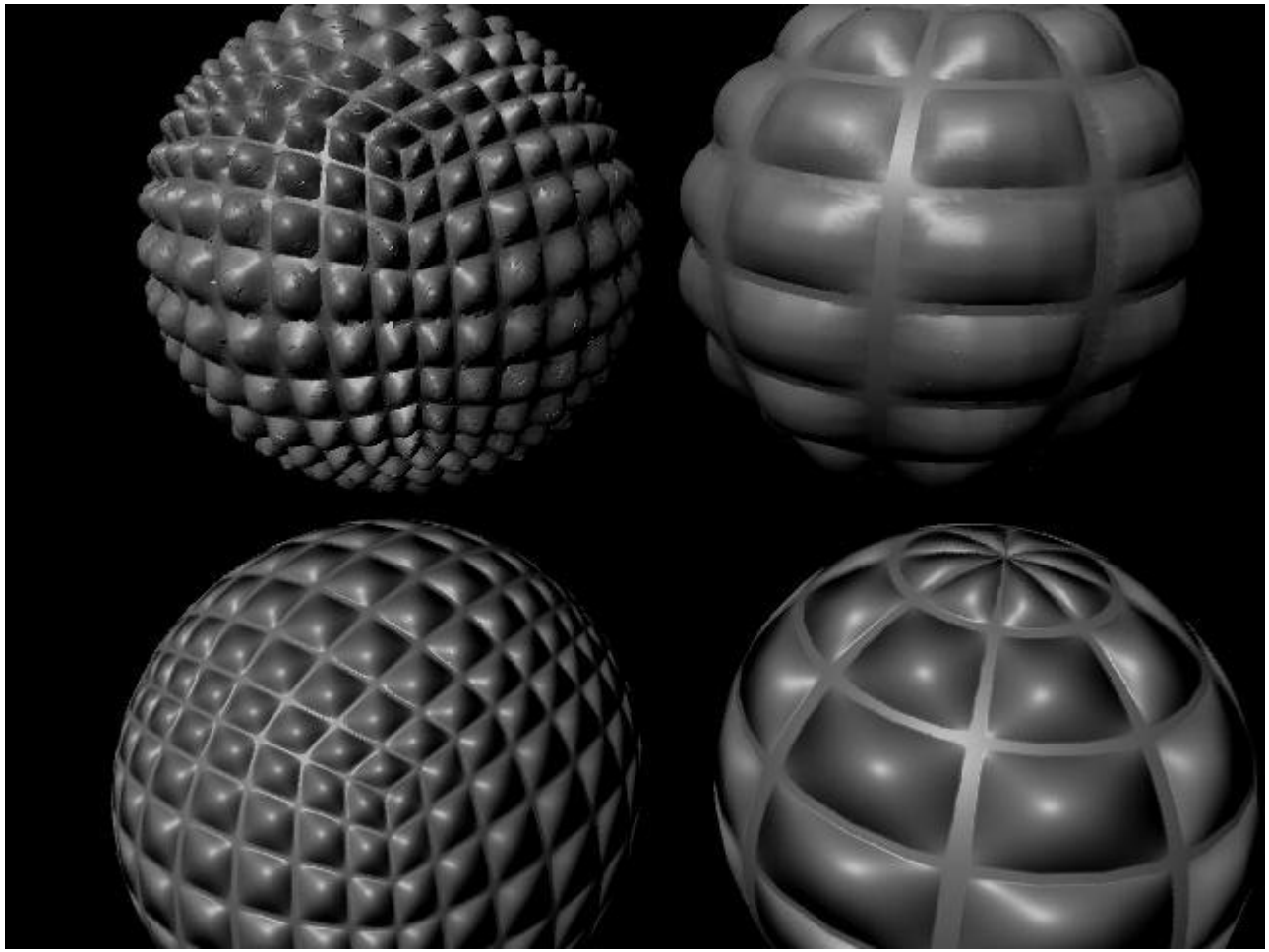
- O algoritmo descrito por Kaneko é um processo de único passo que não conta com oclusões.
- Melhorias posteriores foram feitas para o algoritmo incorporando abordagens iterativas para permitir a oclusão e renderização de silhueta mais precisas.

Displacement Mapping

Displacement Mapping



Displacement Mapping



Relief mapping

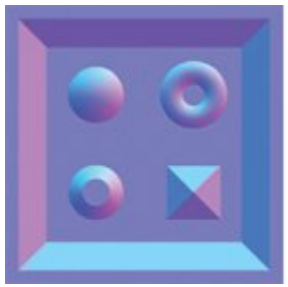
- Criado por Manuel M. Oliveira em 2000, simula a aparência de detalhes de superfícies geométricas detalhes pelo sombreamento de fragmentos individuais de acordo com alguma profundidade e de informações das normais da superfície, mapeadas sobre modelos poligonais.
- Um mapa de profundidade (normalizado entre [0,1]) representa detalhes geométricos que se encontram sob a superfície poligonal.

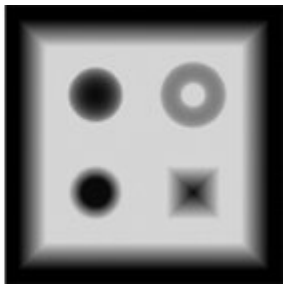
Relief mapping

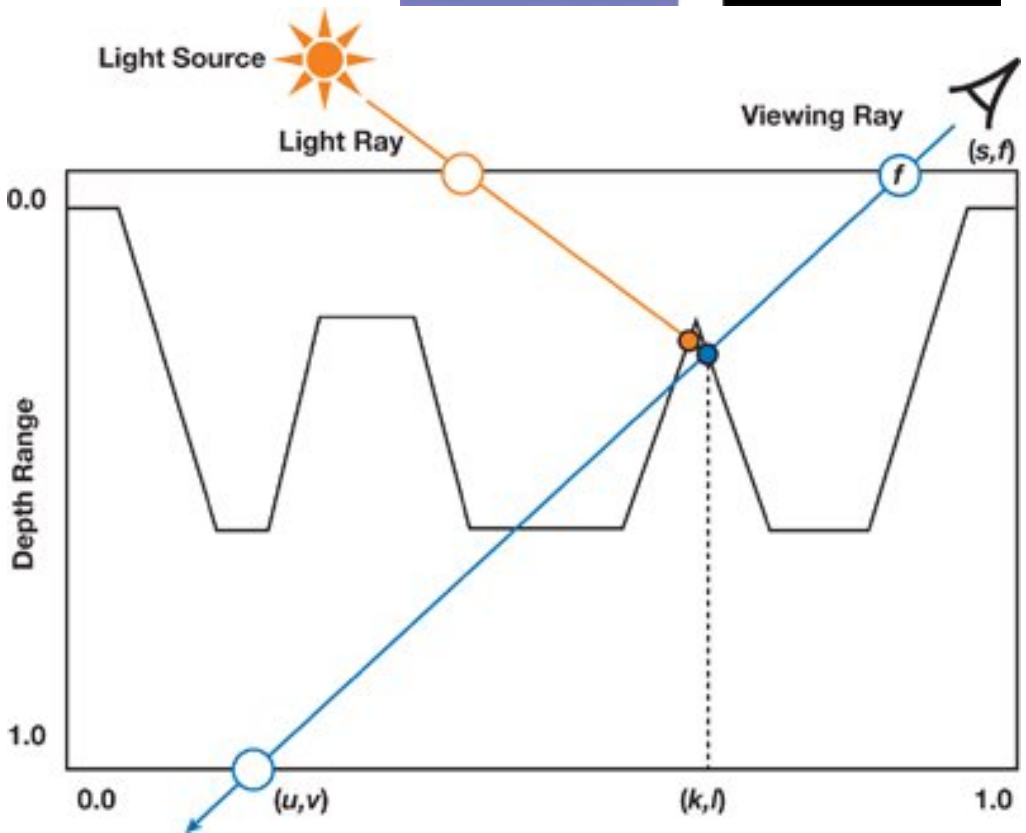
- Mapas de profundidade e de normais podem ser armazenados como uma única textura RGBA chamada de textura em relevo (Oliveira et al., 2000).
- Para melhor desempenho, é recomendável separar a profundidade e normais em duas texturas diferentes.

Relief mapping

- A figura acima mostra os mapas de normais e de profundidade cuja secção transversal é mostrado na figura abaixo.







Relief mapping

- O mapeamento de detalhes para um modelo poligonal é feito de forma convencional, através da atribuição de um par de coordenadas de textura para cada vértice do modelo.
- Durante o processamento, o mapa de profundidade pode ser redimensionado dinamicamente para conseguir diferentes efeitos.
- A oclusão correta é alcançada por meio da atualização do *z-buffer*.

Relief mapping

- O *rendering* é realizado inteiramente na GPU e pode ser dividido em três etapas:
  - Para cada fragmento *f* com coordenadas de textura (*s*, *t*), primeiro transforma a direção de visualização *V* para o espaço tangente de *f*.
  - Em seguida, encontra a interseção *P* do raio de visão transformado com o mapa de profundidade.
  - Finalmente, usa a posição correspondente de *P*, expressa em espaço de câmara e a normal armazenada em (*k*, *l*) para "pintar" *f*.

Relief mapping

- O auto-sombreamento pode ser aplicado verificando se o raio de luz atinge *P* antes de atingir qualquer outro ponto sobre o relevo.
- A oclusão adequada entre o *relief mapping* e outros objetos é obtida simplesmente através da atualização do *z-buffer* com a coordenada *z* de *P* (expresso em espaço de câmera e depois de projeção dividindo por *w*).
- Este *z-buffer* atualizado também suporta o uso combinado de *relief mapping* com *shadow mapping* (Williams 1978)

Relief mapping

Mais informações:

http://www.inf.ufrgs.br/~oliveira/RTM.html

https://vimeo.com/32003362

http://developer.download.nvidia.com/books/gpu_gems_3/samples/gems3_ch18.pdf