

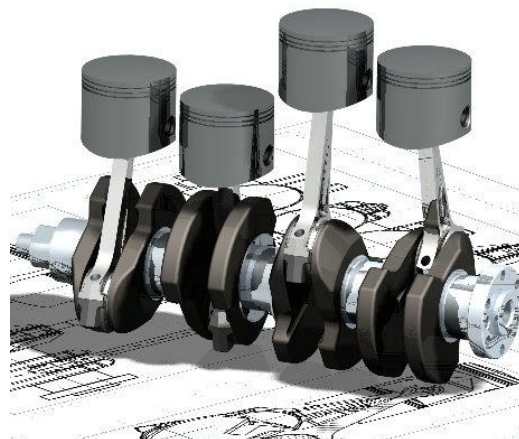
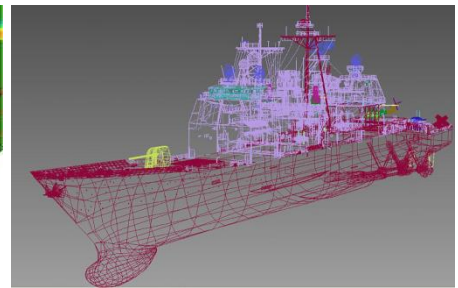
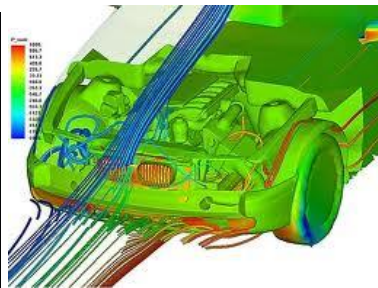
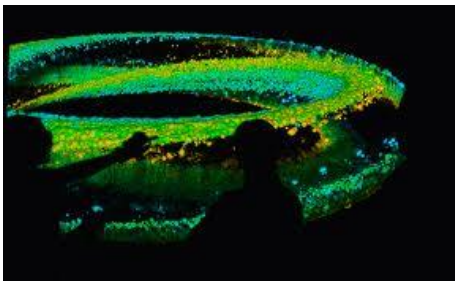
## Computação Gráfica 3D

### Definição

Faz uso de 3 coordenadas:  $x$ ,  $y$  e  $z$ .

### Aplicações

- Jogos 3D
- Visualização
- CAD
- Desenhos animados por computador
- Realidade Virtual (VR) e aumentada (AR)

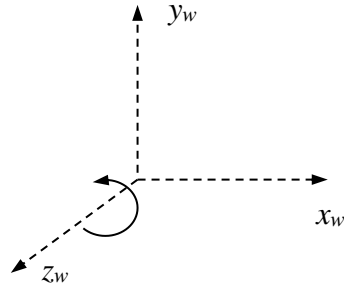


# 1 Sistemas de Coordenadas

Existem dois sistemas de coordenadas 3D: *left-handed* ou *right-handed*.

Em um sistema *right-handed*, fazendo-se a mão girar em torno do eixo  $z$ , do eixo  $x$  positivo para o eixo  $y$  positivo, tem-se o polegar apontando na direção positiva do eixo  $z$ . O sistema de coordenadas do mundo é definido neste sistema.

No sistema *left-handed* o polegar aponta para o  $z$  negativo. Geralmente utiliza-se o sistema *left-handed* em coordenadas de visualização.

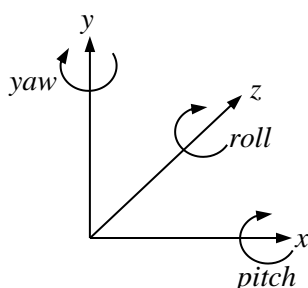


## 2 Transformações

**Rotação:** Para vetores em três dimensões, pode-se fazer a rotação sobre qualquer eixo de rotação, sendo principalmente sobre os eixos  $x$ ,  $y$  e  $z$ . Neste caso, usa-se as seguintes matrizes  $R$ .

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_y(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_z(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Pode-se utilizar ângulos de Euler para descrever a rotação de um objeto em um espaço euclidiano 3D. Para posicionar um objeto em uma dada orientação, basta aplicar uma sequência de rotações descrita pelos ângulos de Euler. Isso significa dizer que a matriz de rotação pode ser decomposta como o produto de três rotações sobre os eixos cartesianos  $x$ ,  $y$  e  $z$ . A rotação deve ocorrer na ordem  $R_z R_y R_x$ . Em aplicações aeroespaciais, costuma-se nomear os ângulos de Euler de modo que o ângulo no eixo  $x$  é chamado de *pitch*, no eixo  $y$  de *yaw* e no eixo  $z$  de *roll*. Essa nomenclatura varia entre referências bibliográficas.



[http://en.wikipedia.org/wiki/Yaw,\\_pitch,\\_and\\_roll](http://en.wikipedia.org/wiki/Yaw,_pitch,_and_roll)

## Translação

$$P' = P + T$$
$$P' = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} = \begin{bmatrix} x + T_x \\ y + T_y \\ z + T_z \end{bmatrix}$$
$$P' = TP = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + T_x \\ y + T_y \\ z + T_z \\ 1 \end{bmatrix}$$

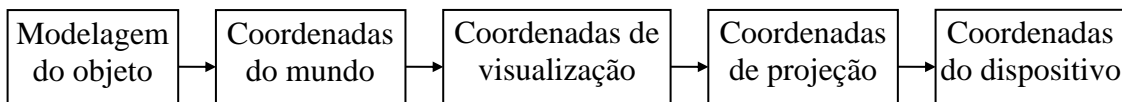
## Escala

$$P' = S \cdot P$$
$$P' = SP = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} xS_x \\ yS_y \\ zS_z \\ 1 \end{bmatrix}$$

## 3 Visualização 3D

→ O processo de visualização de uma cena 3D por computador é semelhante ao processo envolvido para se tirar uma fotografia. Deve-se posicionar a câmera no espaço e definir sua orientação (**6 graus de liberdade: 3 eixos de translação e 3 eixos rotação, semelhante a um spaceball**).

Como a maioria dos monitores (*displays*) existentes são bidimensionais, deve-se usar processos para converter objetos do espaço tridimensional para uma representação bidimensional. Este processo possui um termo conhecido como *three-dimensional pipeline*. As etapas deste pipeline são mostradas na seguinte figura, e incluem modelagem, visualização e conversão de diferentes tipos de coordenadas.



Como primeiro passo da etapa tem-se a definição de como será composto o cenário a ser modelado. Após sua definição, têm-se pontos tridimensionais no sistema de coordenadas do mundo que o definem.

As coordenadas de visualização (*view coordinates*) são as coordenadas do sistema de visualização, e não necessariamente precisam ser coincidentes com as coordenadas do mundo. Neste sistema de coordenadas é que estão localizados o observador e o plano de projeção da imagem.

A próxima etapa do processo é fazer a conversão das coordenadas de visualização para o plano de projeção, que será mapeado para o monitor. É nesta etapa do processo onde é realizada a conversão do espaço tridimensional para o bidimensional (tela do computador). Ao mapear a imagem para a tela, são realizadas operações para recortar partes do cenário que não estão dentro da área da janela de visualização (*viewport*), para posterior identificação de superfícies visíveis e aplicação de processos de renderização.

### Coordenadas de visualização

O *viewing system* é um sistema de coordenadas usado para dar ao observador um maior controle e mobilidade da posição em que se deseja observar um cenário tridimensional. Estes sistemas permitem que o plano de projeção esteja em qualquer lugar do espaço, ou seja, permite que se veja o cenário sob qualquer ângulo.

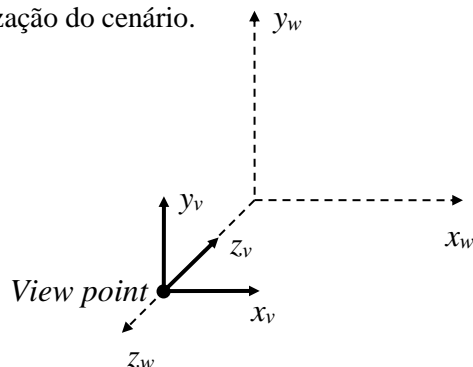
Um exemplo pode ser dado ao se observar um automóvel. Movendo-se ao seu redor, têm-se diferentes ângulos de visão. Abaixando-se pode-se vê-lo por baixo, usando-se uma escada pode-se vê-lo de cima; e obtêm-se uma visão interna entrando-se dentro do mesmo.

Tudo depende da forma como definimos o ponto onde o observador se encontra, que é chamado de centro de projeção (*view point* ou *view reference point*) e, associado a ele, existe o plano de projeção onde será projetada a imagem (Observe que o plano de projeção não faz parte das coordenadas de Visualização).

Além do *view point*, pode-se também definir uma orientação espacial do observador, ou seja, para onde ele está olhando (ângulo de elevação) e qual a inclinação de sua cabeça (ângulo de rotação). Os diferentes sistemas de visualização apresentados a seguir nos mostram mais características do que podemos simular em um computador.

## 1 Visualização sobre o eixo Z

A forma mais simples de se visualizar um cenário é localizar o observador sobre o eixo de projeção  $z$ , como mostrado na seguinte figura. Para este tipo de sistema, não é necessário nenhum processamento na etapa de geração de coordenadas de visualização do pipeline de visualização, pois o *view point*, assim como o plano de projeção paralelo ao plano  $x_w y_w$ , já estão localizados sobre o eixo  $z$ . Deste modo, necessita-se usar apenas a projeção para fazer a visualização do cenário.



*View point* (centro de projeção) localizado sobre o eixo  $z$ , em um sistema *left-handed*.

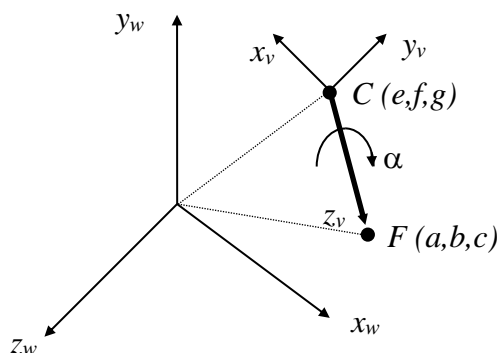
Deve-se apenas verificar se o sistema de coordenadas dos dois referenciais é o mesmo: *left-handed* ou *right-handed* e fazer ajustes se necessário (aplicar reflexão).

## 2 Visualização definida por um vetor arbitrário

Usando-se um vetor arbitrário no espaço tridimensional para definir a direção onde a câmera irá apontar dá ao observador um controle total de como observar um cenário sobre diferentes ângulos e posições. Com este sistema de visualização consegue-se gerar a maioria das aplicações gráficas, como simuladores, jogos tridimensionais, dentre outros.

Este sistema define basicamente dois pontos ( $C$  e  $F$ ), que definem a orientação da *reta de visualização* e um ângulo  $\alpha$  de rotação. Com a definição destas três variáveis pode-se posicionar a câmera em qualquer posição.

A câmera é então posicionada no ponto  $C$  (câmera) e direcionada no ponto  $F$  (foco). O plano de projeção é perpendicular ao eixo  $z$ , que deve ser coincidente com a *reta de visualização* definida pelos pontos  $C$  e  $F$ .



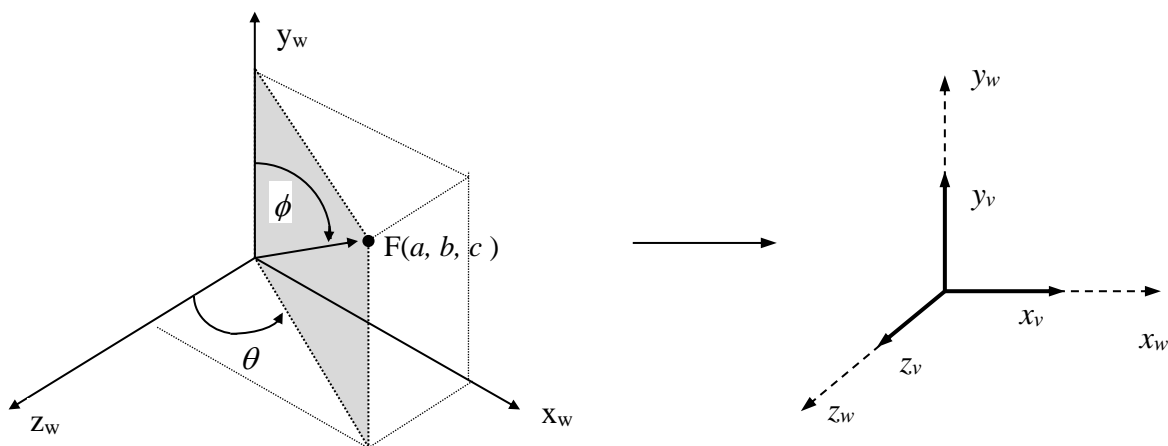
Antes de calcular as coordenadas de projeção deve-se tornar os dois sistemas de coordenadas coincidentes. Isso pode ser feito por uma sequência de transformações geométricas afins:

1. Translade o centro de projeção da câmera (*view point*) para a origem do sistema de coordenadas do mundo. (ou levar o mundo para o sistema de coordenadas da câmera).
2. Aplique rotações para alinhar os eixos  $x_v, y_v, z_v$  com os respectivos eixos do mundo:  $x_w, y_w, z_w$ .

A matriz de translação é dada por

$$T = \begin{bmatrix} 1 & 0 & 0 & -e \\ 0 & 1 & 0 & -f \\ 0 & 0 & 1 & -g \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Para o cálculo das rotações, devem-se encontrar ângulos  $\phi$  e  $\theta$  que representam a orientação da câmera em relação ao sistema de coordenadas do mundo.



Esta etapa pode requerer até 3 rotações sobre os eixos, dependendo do valor do ângulo  $\alpha$ . No caso mais geral, deve-se aplicar uma sequência de transformações  $R_z \cdot R_y \cdot R_x$ . Ou seja, primeiro rotaciona-se no eixo  $x$ , depois sobre  $y$  e finalmente sobre  $z$ .

Fazendo-se uma analogia a este processo, pode-se imaginar que a câmera esteja conectada aos objetos da cena por um eixo virtual. Cada operação aplicada à câmera para alinhar seus eixos com o sistema de coordenadas do mundo também é aplicada sobre os objetos da cena, visto que ambos estão unidos. Ao final do processo a câmera esta observando o mundo sob o eixo  $z$  e continua com a mesma visão dos objetos da cena que tinha antes de serem aplicadas as transformações.

A matriz resultante (rotações e translação) deve ser então aplicada sobre cada elemento do cenário, para então realizar a etapa de projeção. Este processo deve ser realizado para tornar a projeção mais simplificada.

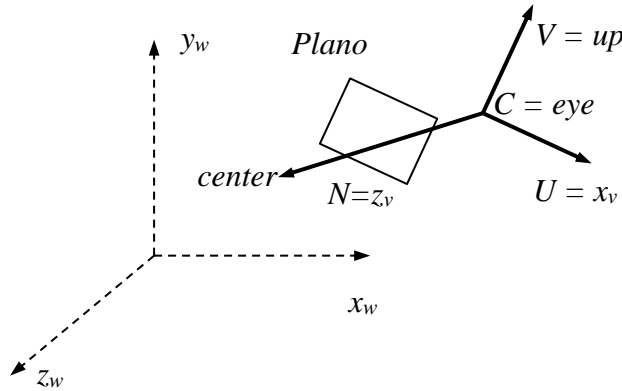
### 3 Visualização definida por dois vetores e um ponto (Ou dois pontos e um vetor - gluLookAt)

Dado o sistema anterior, pode-se definir a câmera por um vetor  $N$  que é normal ao plano de projeção (aponta para  $z_v$  positivo), e por um vetor  $V$  (chamado *up*), que orienta a câmera em relação à direção de visualização (rotação sobre o eixo  $z_v$ ).

Dados  $N$  e  $V$ , pode-se calcular o vetor  $U$  como sendo  $N \times V$ , que aponta na direção do eixo  $x_v$  positivo. A biblioteca OpenGL/Glut( `gluLookAt(eye, center, up)` ) utiliza este sistema de visualização (**utiliza dois pontos e um vetor**). Acesse os sites para maiores detalhes de implementação: <http://pyopengl.sourceforge.net/documentation/manual-3.0/gluLookAt.xhtml> ou <http://www.opengl.org/sdk/docs/man/xhtml/gluLookAt.xml>. **NOTA:** Nesse mesmo site podem ser encontradas formulações matemáticas para várias outras funções da API do OpenGL, que vão ser muito úteis

caso seja necessário programar na versão 3.2 ou superior desta API (Ex: gluPerspective, glFrustum, glRotate, glMultMatrix, glOrtho, etc).

A câmera está posicionada no ponto  $C(e, f, g)$ .



Como no sistema anterior, deve-se transladar o sistema de coordenadas da câmera para o mundo.

$$T = \begin{bmatrix} 1 & 0 & 0 & -e \\ 0 & 1 & 0 & -f \\ 0 & 0 & 1 & -g \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Dados os vetores  $N$  (expresso por **center – eye**) e  $V$  (vetor up), pode-se obter  $n$ ,  $u$  e  $v$  unitários por

$$n = \frac{N}{|N|} = (n_1, n_2, n_3)$$

$$u = \frac{V \times N}{|V \times N|} = (u_1, u_2, u_3)$$

$$v = n \times u = (v_1, v_2, v_3)$$

A matriz de **mudança de base** é dada por [4]. Esta matriz leva a cena para o espaço da câmara, sendo  $n$  representando o eixo  $z$ ,  $u$  o eixo  $x$  e  $v$  o eixo  $y$ . Deve-se observar que o objeto está modelado em relação a base canônica do  $\mathbb{R}^3$ .

$$R = \begin{bmatrix} u_1 & u_2 & u_3 & 0 \\ v_1 & v_2 & v_3 & 0 \\ n_1 & n_2 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

e a transformação completa por

$$M = R \cdot T$$

Para maiores detalhes de outras formas de visualização, ver [watt].

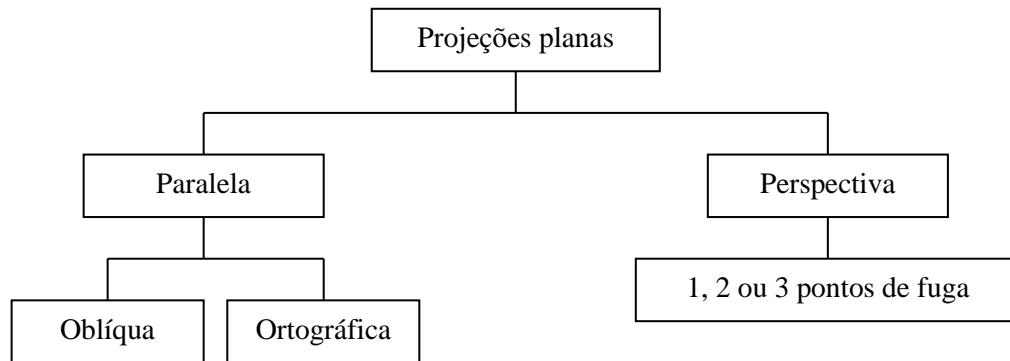
## Projeção

Para fazer a visualização de objetos 3D em um dispositivo 2D (o monitor, por exemplo), deve-se fazer a transformação do espaço 3D para o 2D, ou seja, converter as coordenadas que correspondam a uma visão do objeto de uma posição específica. Este processo é chamado de projeção.

Para fazer a projeção **deve-se definir o plano de projeção** e o centro de projeção. O plano de projeção é a superfície onde será projetado o objeto. No centro de projeção partem retas (raios de projeção) que passam pelo plano de projeção e tocam no objeto.

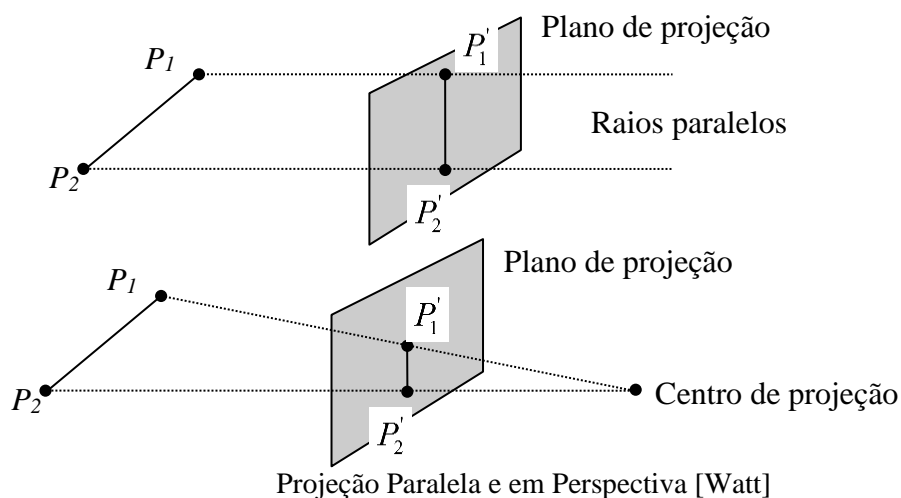
Existem vários tipos de projeção, que são dadas em função do centro de projeção, plano de projeção e direção dos raios de projeção

Em projeções paralelas, as coordenadas são transformadas para o plano de visualização seguindo linhas paralelas. O centro de projeção é localizado no infinito. Já, projeções em perspectiva, os objetos são transformados segundo linhas que não são paralelas, ou seja, linhas que se encontram no centro de projeção.

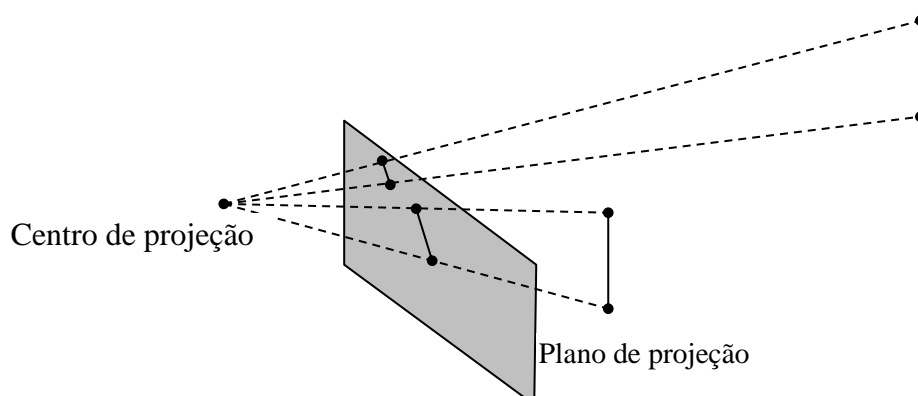


A diferença básica entre elas é que a primeira não deforma as características originais do objeto nem dá uma representação realista do objeto. A noção de profundidade (afastamento) não é apresentada. A segunda dá representações realista, mas para isso, deforma a forma original do objeto.

Em uma projeção ortográfica, as linhas de projeção são paralelas entre si e perpendiculares ao plano de projeção. As projeções oblíquas também são paralelas, porém inclinadas em relação ao plano de projeção.



Em uma projeção em perspectiva, linhas distantes são visualizadas menores que linhas mais próximas de mesmo comprimento.



## 1 Projeção Ortográfica

Em projeções paralelas são mantidos a forma e tamanho originais do objeto. Este tipo de projeção transforma o objeto tridimensional em bidimensional pela simples desconsideração de uma coordenada, ou seja, todos os valores em um certo eixo recebem valor zero, o que é equivalente a anular o eixo. Considerando-se uma projeção sobre o plano  $xy$ , tem-se a seguinte matriz de projeção

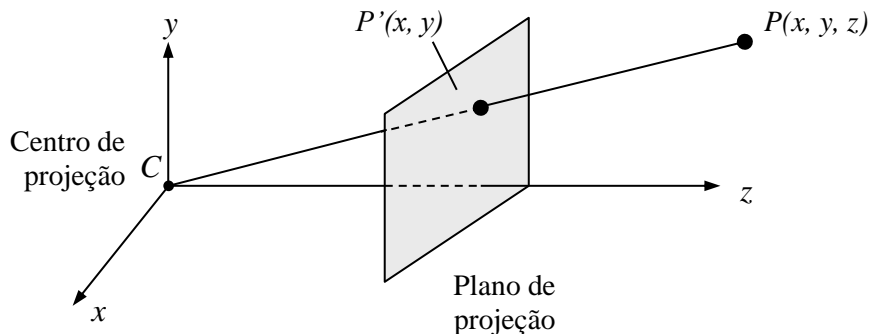
$$T_{ort} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [4-1]$$

Esta projeção é importante pois é usada como processo final das projeções em perspectiva, descritas nas próximas seções.

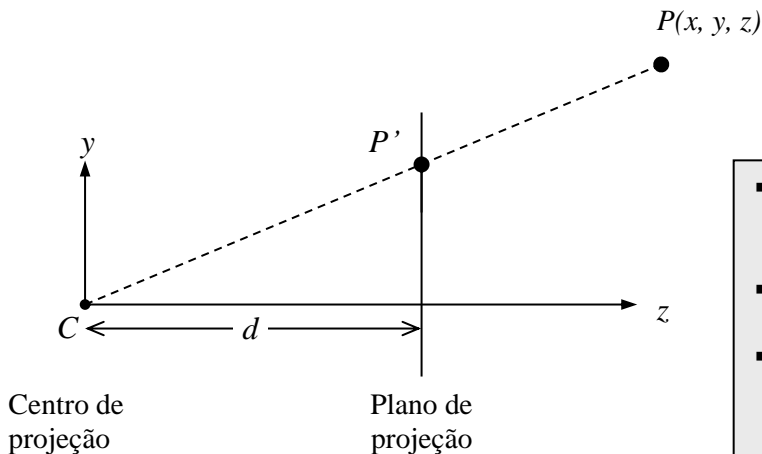
- Como seria a projeção ortográfica se a câmera não estivesse sobre o eixo  $z$ ? Com certeza não seria simples assim.
- Como seria  $T_{ort}^{-1}$ ?

## 2 Projeções em Perspectiva

O ponto  $P(x,y,z)$  é um ponto tridimensional no sistema de coordenadas de visualização. Este ponto deve ser projetado no plano  $xy$  (perpendicular ao eixo  $z$ ) posicionado a uma distância  $d$  da origem do sistema.  $P'(x',y')$  representa a projeção deste ponto no plano de projeção (tela). Fazendo-se uma analogia a uma câmera fotográfica, tem-se em  $C$  a posição da câmera (na posição  $(0,0,0)$ ) e em  $P'(x', y')$  a posição do filme fotográfico. Este modelo é chamado de **pinhole camera** (Furo de agulha).



Considerando-se a projeção apenas no eixo  $y$ , tem-se



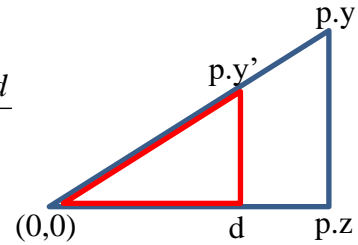
- Quanto menor for  $d$ , menor é a projeção. Se  $d=0$ , o objeto desaparece.
- Mudando  $d=100$  para  $d=-100$ , muda-se a orientação do objeto.
- Se  $d$  for igual a posição do objeto, não há alteração de tamanho.



Aplicando a relação de triângulos obtém-se

$$\frac{x'}{d} = \frac{x}{z} \quad \text{ou} \quad \frac{x'}{x} = \frac{d}{z} \quad x' = \frac{xd}{z}$$

$$\frac{y'}{d} = \frac{y}{z} \quad y' = \frac{yd}{z}$$



Fazendo-se  $r = 1/d$ , em representação matricial, esta transformação pode ser vista da seguinte maneira:

$$T_{pers} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & r & 0 \end{bmatrix}$$

Concatenando-se uma projeção ortográfica para anular a componente z tem-se

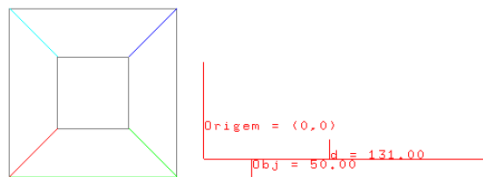
$$T_{pers} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & r & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & r & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 0 \\ rz \end{bmatrix}$$

Homogeneizando a matriz obtém-se a transformação de perspectiva

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 0 & 1 \\ rz & rz & 0 & 1 \end{bmatrix} = \begin{bmatrix} xd & yd & 0 & 1 \\ z & z & 0 & 1 \end{bmatrix}$$

**OBS:** Deve-se observar que se o objeto tiver alguma coordenada  $z=0$  vai gerar um erro de divisão por zero. Em termos práticos, o objeto deveria ser desenhado com tamanho infinito, visto que quanto mais próximo do centro de projeção estiver, maior é a sua projeção no plano de projeção.

**Exercício:** Implemente a visualização de um cubo, em **wireframe** (com o uso de 12 linhas em 2D), utilizando a projeção em perspectiva. O cubo deve ser modelado na origem e ter coordenadas variando entre  $[-1, 1]$ . O centro do cubo tem coordenadas  $(0,0,0)$ . Juntamente com a visualização, o programa deve permitir o ajuste do parâmetro  $d$ , e sua visualização gráfica como na figura. Deve permitir também ajustar a posição do cubo, e aplicar rotações e escalas.

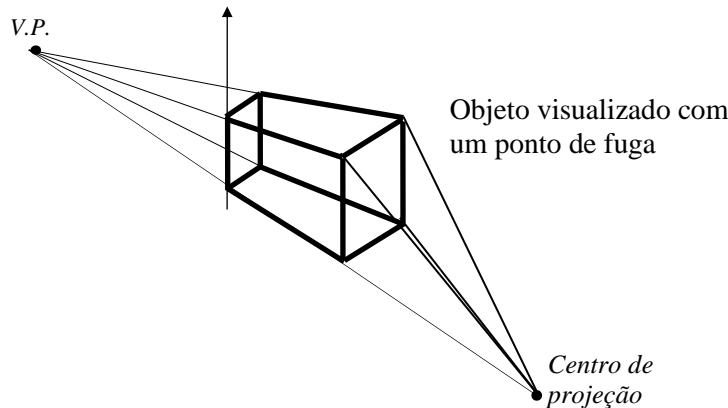


**Dicas de solução:**

```
Vector3 entrada[8] //8 vertices que definem o cubo
void render()
{
    Vector3 p;
    Vector2 saida[8]; //vetor 2D pois esta projetado e não tem z.
    Para i=0..7 //para um cubo são 8 vertices
        p = entrada[i]
        p = rotacionaY(p, ang )
        p = transladaZ(p, 3 ) //a câmera não pode estar dentro do objeto
        saida[i] = projeta(p, d) //faz a continha xd/z
    Desenha(saida) //para um cubo são 12 arestas
    ang += 0.01; //faz uma animação da rotacao
}
```

## 2.1 Projeção com um ponto de fuga

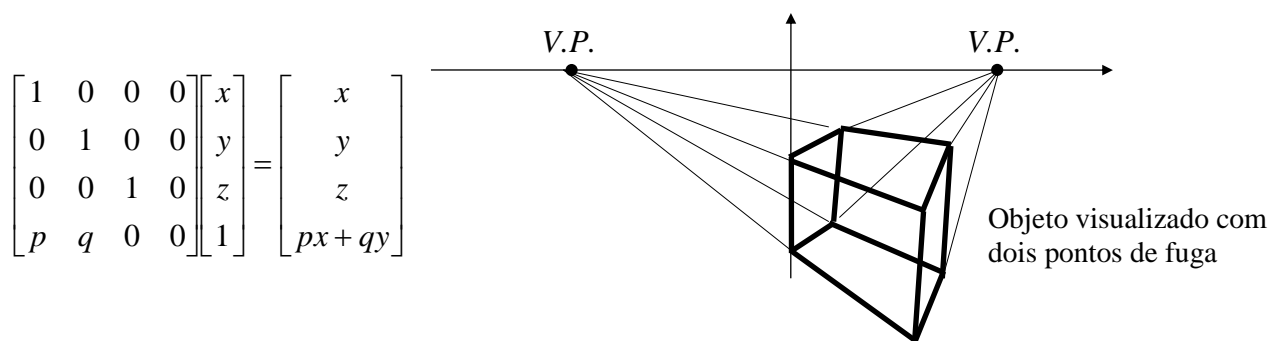
Um fator que caracteriza uma matriz de projeção é o valor dos elementos da quarta linha. O elemento  $A_{43}$  possui um valor  $r$ , e dos demais possuem valor zero. O número de elementos diferentes de zero irá determinar o número de pontos de fuga (*vanish point* - V.P.) presentes na projeção. Como mostrado na seguinte figura, um ponto de fuga é um ponto para onde as linhas que definem o objeto convergem. Existe uma relação entre o centro de projeção e o ponto de fuga. Se o centro de projeção estiver no ponto  $z = r$ , o ponto de fuga estará no ponto  $z = -r$ .



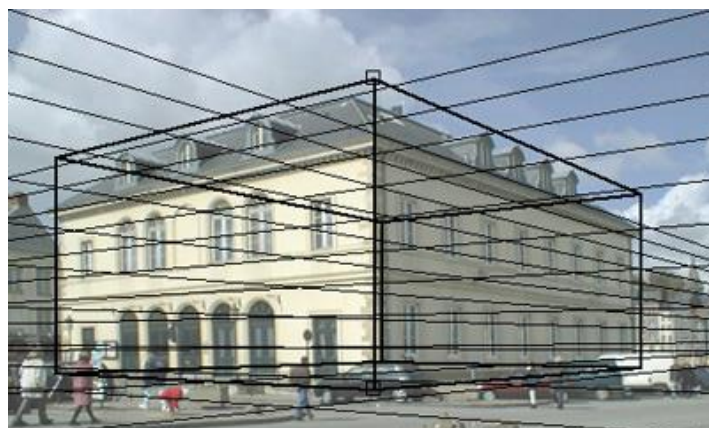
Nos exemplos até agora citados, o centro de projeção e o ponto de fuga sempre estavam localizados sobre o eixo  $z$ . Caso desejar-se colocar o centro de projeção sobre outro eixo, basta apenas mudar o valor de  $r$  da posição  $A_{43}$  e colocá-lo na posição  $A_{42}$  para se trabalhar sobre o eixo  $y$ , ou na posição  $A_{41}$  para trabalhar-se sobre o eixo  $x$ .

## 2.2 Projeção com dois pontos de fuga

Caso desejar-se dar mais realismo aos objetos, pode-se estender a técnica usada na seção anterior e, em vez de definir apenas um centro de projeção, definir dois e, por consequência, teremos também dois pontos de fuga. A seguinte figura mostra um objeto visualizado com dois pontos de fuga.

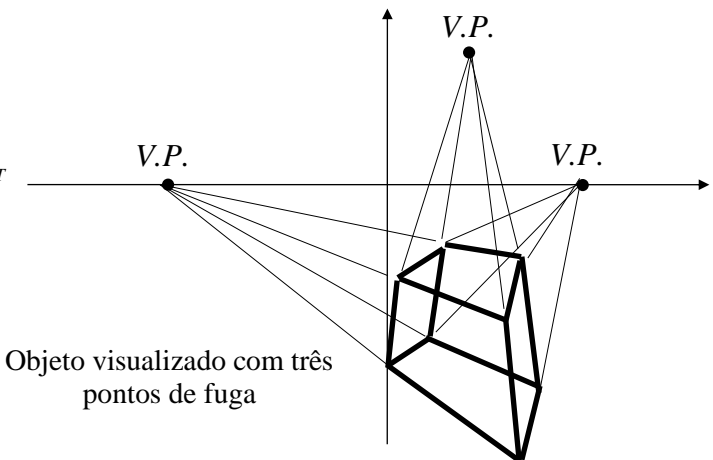


$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p & q & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ px + qy \end{bmatrix}$$



## 2.3 Projeção com três pontos de fuga

A projeção que mais dá noção da forma tridimensional de um objeto é com três centros de projeção. Neste caso teremos a matriz de transformação com todos os elementos da quarta coluna não nulos que terá a seguinte forma

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p & q & r & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x & y & z & (px + qy + rz) \end{bmatrix}^T$$


Objeto visualizado com três pontos de fuga

### Modelos de Câmera Sintética

O modelo de câmera sintética apresentado é um modelo chamado *pinhole*. O modelo *pinhole* é uma câmera sem a convencional lente de vidro. Faz uso de um pequeno orifício que projeta a luz proveniente da cena sob o filme. Para garantir que a imagem gerada esteja focada, deve-se garantir que esse orifício (*pinhole*) seja pequeno (na ordem de 0.5 mm). A dimensão deste orifício determina a **abertura** da câmera. Se esse orifício for muito grande, a imagem se tornará desfocada. Quanto menor for o orifício, maior deve ser o **tempo de exposição** da foto. Quanto menor for a abertura, maior será a **profundidade de campo** (*depth of field*). Quanto menor a abertura, maior deve ser o **tempo de exposição** da foto. Deve-se observar que deve-se ter uma quantidade adequada de luz para sensibilizar o filme (ou sensor CCD ou CMOS). O problema de se ter exposição longa é que se o fotógrafo ou o alvo se moverem durante a foto, obtém-se uma imagem borrada - não necessariamente fora de foco, mas com um efeito muito semelhante.

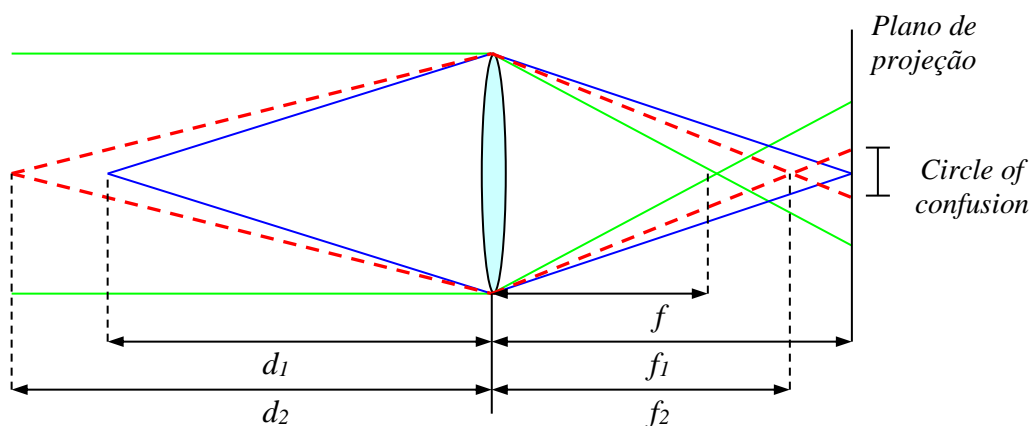
Deve-se observar que a dimensão do “orifício” usada em uma câmera sintética é zero. Isso garante que qualquer cena seja sempre renderizada em foco (Ver material de Síntese de Imagens), ou seja, a profundidade de campo é infinita. Por isso, tanto objetos próximos como objetos afastados se encontram sempre em foco. Isso não é possível de ser alcançado com uma máquina fotográfica convencional que faça uso de lentes.

A seguinte figura ilustra um exemplo de uma máquina fotográfica do tipo *pinhole*. Observe que a imagem gerada é sempre de invertida. Esse mesmo efeito ocorre em uma câmera sintética se o plano de projeção estiver atrás da câmera.

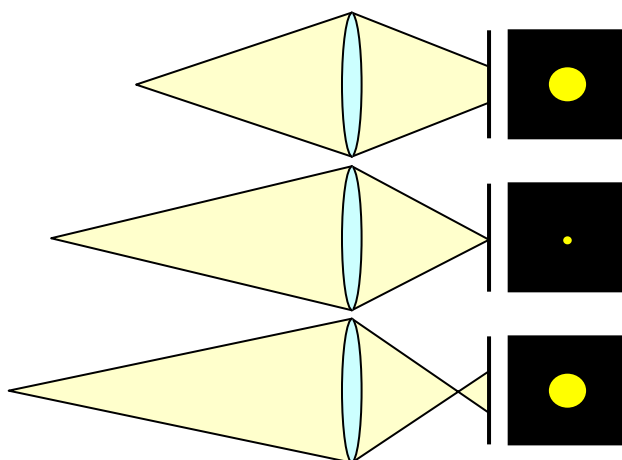


Câmeras com lente têm vantagens e desvantagens em relação ao modelo *pinhole*. A principal desvantagem é que as cenas podem sair fora de foco. A vantagem é que o tempo de exposição da foto pode ser muito menor (de 30 segundos até 1/8000 segundos). A principal diferença refere-se à abertura. Como a abertura é maior, mais luz entra na câmera e o tempo de exposição pode ser menor.

Uma câmera utiliza-se de lentes para fazer a projeção da imagem no filme, como mostrado na seguinte figura.



Deve-se observar que para uma dada configuração, apenas objetos a uma distância  $d$  da câmera vão estar em foco (linha azul – distância  $d_1$ ). O objeto posicionado na distância  $d_2$  vai estar fora de foco, como mostrado na seguinte figura.



O ponto amarelo caracteriza o **círculo de confusão** (*circle of confusion* - Disco de luz ou círculos luminosos da imagem, produzido pela objetiva quando o objeto a ser fotografado não está perfeitamente focado). Em óptica, também é conhecido como *disk of confusion*, *circle of indistinctness*, *blur circle*, etc. Este círculo é gerado por um cone de raios luminosos que não convergem para um único ponto. Quanto maior for este círculo, mais desfocada será a imagem.

### Especificação de lentes para máquinas fotográficas

Várias variáveis definem uma lente de uma máquina fotográfica (tanto para máquinas *Point-and-shoot* como para máquinas SLR). Nesta seção vamos analisar apenas a abertura e distância focal.

Para tirar uma foto adequada, deve-se determinar a quantidade de luz absorvida. Basicamente, a quantidade de luz pode ser definida como

$$\text{Número\_Fotons} = \text{ISO} * \text{Tempo\_exposição} * \text{Abertura} * \text{Luz\_ambiente}$$

ou seja, quanto maior o ISO, menor pode ser a exposição e abertura. Tendo-se abertura grande, pode-se reduzir o ISO ou o tempo de exposição, e assim por diante. Para um **local escuro** (baixa Luz\_ambiente), deve-se aumentar os outros parâmetros, geralmente o ISO e Tempo\_exposição. O parâmetro Abertura é físico (depende da lente) e tem um limite. O ISO também tem um limite, e quanto maior, maior o ruído. O ISO pode ser visto como um multiplicador (ganho). Ver referências abaixo.

Máquinas fotográficas profissionais permitem definir esses parâmetros.



No modo Auto, a máquina define ISO, abertura e tempo de exposição.  
 No modo P (Program), pode-se definir o ISO. A exposição e abertura são definidas pela máquina.  
 No modo A (Aperture), pode-se definir a abertura e ISO. A exposição é definida pela máquina.  
 No modo S (Shutter), pode-se definir o tempo de exposição e ISO. A abertura é definida pela máquina.  
 No modo M (Manual), pode-se definir todos os parâmetros.  
 Adicionalmente a todos os modos, pode-se definir o EV, para aumentar ou diminuir o brilho da imagem.

Para informações adicionais, consulte os seguintes textos:

<http://www.clarkvision.com/articles/iso/> - Muito bom.

<http://en.wikipedia.org/wiki/Gain>

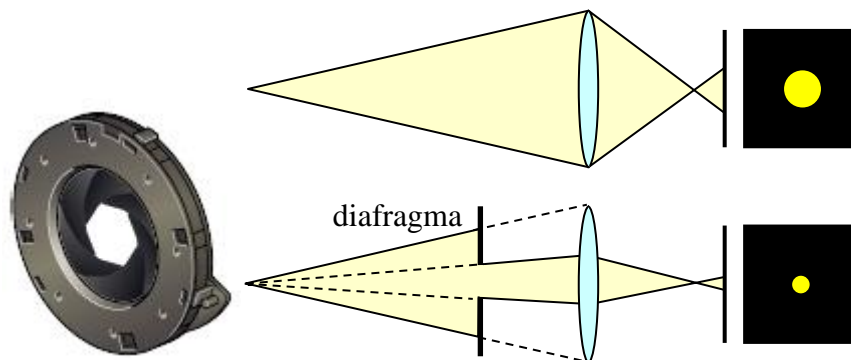
[http://www.astropix.com/HTML/I\\_ASTROP/HOW.HTM](http://www.astropix.com/HTML/I_ASTROP/HOW.HTM) - GAIN in digital sensors

<http://www.qsimaging.com/blog/understanding-gain-on-a-ccd-camera/>

<http://www.clarkvision.com/articles/digital.sensor.performance.summary/>

A abertura (<http://en.wikipedia.org/wiki/Aperture>) se refere à quantidade de luz que entra na máquina fotográfica. Na seguinte figura é ilustrado o efeito de **reduzir a abertura** da lente (por meio de um diafragma). Pode-se observar que o círculo de confusão tornou-se menor, o que caracteriza uma imagem com melhor foco e com maior profundidade de campo. Porém deve-se ter um **tempo de exposição** maior. Tendo-se um tempo de exposição maior, por sua vez, maior é a chance que a câmera se mova neste tempo, ocasionando borrões na imagem final. Por isso, é importante o uso de um tripé para assegurar que a câmera não irá se mover durante a exposição. Câmeras profissionais incorporam recursos de **estabilização de imagem**, que pode consistir de mover o sensor ou as lentes para compensar o movimento (tremor) da câmera.

[[http://www.canon.co.jp/Imaging/enjoydslr/p\\_2\\_005.html](http://www.canon.co.jp/Imaging/enjoydslr/p_2_005.html)]

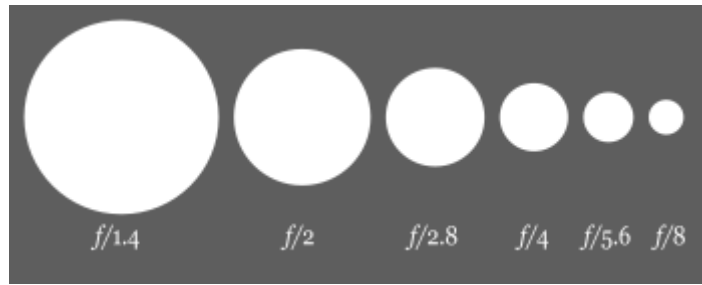


A abertura em câmeras fotográficas é dada pelo **f-number** ( $f/\#$ ), que expressa o diâmetro da entrada da luz em função da distância focal da lente

$$f/\# = \frac{f}{D}$$

onde D é o diâmetro da abertura [<http://en.wikipedia.org/wiki/F-number>].

Valores típicos de *f-number* (*f-stops*) são  $f/1, f/1.4, f/2, f/2.8, f/4, f/5.6, f/8, f/11, f/16, f/22$ , etc. A diferença entre cada valor corresponde a potências de  $\sqrt{2}$ , visto que a cada incremento do *f-number*, reduz-se pela metade a área de entrada da luz na câmera. Quanto maior for o *f-number*, menor é a abertura e maior a profundidade de campo.



Lentes convencionais com zoom óptico (por exemplo, a lente 18-200mm) têm uma abertura máxima em torno de  $f/3.5$  até  $f/5.6$  (sem e com zoom). A abertura máxima é reduzida à medida que se aumenta o zoom [<http://en.wikipedia.org/wiki/Aperture>]. Em lentes profissionais mais caras isso não ocorre (Ex: Nikon 14-24mm  $f/2.8$ ).



Abertura máxima:  $f/2.8$  (Wide)- $5.9$  (Tele)  
 Distância Focal: 4.7-18.8mm ou 26-105mm (equivalente 35mm)  
 Zoom: 4x ( $105/26 = 4$ )  
 Preço: US\$ 110,00



Abertura máxima:  $f/2.8$  (fixa)  
 Distância Focal: 14-24mm (equivalente 35mm) ou 21-36mm (sensor APS-C size)  
 Zoom: 1.7x  
 Peso: 969g  
 Preço: US\$ 2.000,00



Abertura máxima:  $f/3.5$  (W)- $5.6$  (T)  
 Distância Focal: 18-200mm (equivalente 35mm) ou 28-300mm (sensor APS-C size)  
 Zoom: 11.1x  
 Peso: 560g  
 Preço: US\$850,00

A distância focal ([http://en.wikipedia.org/wiki/Focal\\_length](http://en.wikipedia.org/wiki/Focal_length)) é a distância que os raios de luz são convergidos para um ponto focal. Quanto menor for, maior será o ângulo de visão da lente (FOV – *Field of view*). Muitas lentes têm distância focal variável, o que possibilita aproximar ou afastar os objetos. Isso é o que chamado de zoom óptico. A taxa de zoom é dada pela divisão entre a distância focal máxima e mínima (com apresentado nas figuras acima). As lentes que não tem zoom são chamadas de lentes de distância focal fixa ou *prime lenses*.



A medida da distância focal é dada em milímetros, em relação às antigas máquinas fotográficas de filme (padrão 35mm). Como hoje em dia o tamanho dos sensores das máquinas digitais é menor que esse padrão, geralmente os fabricantes colocam a distância focal proporcional ao sensor usado. Para se ter os dados reais da distância focal da lente, deve-se sempre procurar a distância equivalente ao filme 35mm.



Para máquinas Point-and-shoot, geralmente quanto menor for a distância focal mínima, mais cara e melhor é a máquina. Valores considerados razoáveis devem estar abaixo de 30mm. Existe também disponível no mercado (em 2011) máquinas com até 36x de zoom óptico, cuja distância focal varia dentre 4-144mm (22.5-810mm 35mm equivalente).



Abertura máxima: f/3.4 (W)-5.7 (T)  
 Distância Focal: 4-144mm ou 22.5-810mm (equivalente 35mm)  
 Zoom: 36x ( $144/4 = 36$ )  
 Preço: US\$ 400,00

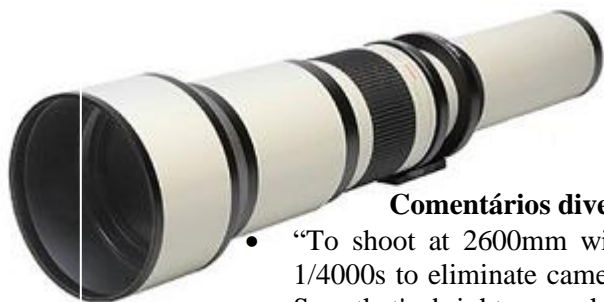
Para máquinas SLR, tem-se uma grande gama de lentes, com variações grandes de distância focal (de 10mm a 600mm) e abertura máxima (f/1.2 até f/5.6).



Abertura máxima: f/4 (fixa)  
 Distância Focal: 600mm (equivalente 35mm) ou 900mm (sensor APS-C size)  
 Zoom: 0x  
 Peso: 5kg  
 Preço: US\$ 10.000,00



Abertura máxima: f/5.6 (fixa)  
 Distância Focal: 1200mm (equivalente 35mm)  
 Zoom: 0x  
 Preço: US\$ 120.000,00 (somente a lente)  
 Peso: 16.5kg  
 (fora de produção)

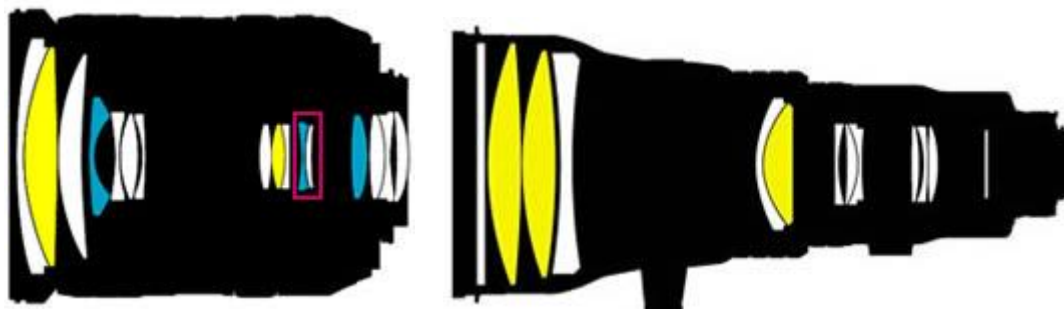


Abertura máxima: f/8-16  
 Distância Focal: 650-1300mm + Teleconverter 2x  
 Zoom: 2x  
 Preço: US\$ 270,00  
 Peso: 1.8 kg

**Comentários divertidos encontrado na internet da lente Phoenix:**

- “To shoot at 2600mm with a crop factor of 1.5 you'll want to shoot at, at most, 1/4000s to eliminate camera shake. I'm wondering if there's anything closer than the Sun that's bright enough to be photographed at f/32, 1/4000s”. Relembrando a fórmula da Luz, temos uma abertura ínfima (que não caso é a abertura máxima da lente) e tempo de exposição mínimo. Por isso, a cena deve estar **beeeeeem iluminada**.
- “Well, I bought this for fun to compare to my canon 600mm lens. This lens is just cheap, and it is. You get what you pay for, all I need to say about this product is that it will have all the optical qualities of a coke bottle, but you don't get the free fizzy drink with it”.

Na seguinte figura são apresentados modelos de lentes de câmeras SLR profissionais. Deve-se observar a quantidade de lentes usadas para garantir uma imagem em foco, juntamente com recursos de zoom óptico.



[<http://nikonimaging.com/global/products/lens/index.htm>]



Para os que tiverem interesse no processo de construção de lentes para máquinas SLR (excluindo a parte de especificação e projeto), os seguintes vídeos fazem uma boa explanação sobre o assunto, justificando o seu alto preço:

Busca no Youtube: “Canon Lens Production”

<http://www.youtube.com/watch?v=MKNFW0YwDYw&NR=1>

<http://www.youtube.com/watch?v=qzpt49qq6v4&feature=related>

<http://www.youtube.com/watch?v=6bQ3-DWh-rA&feature=related>

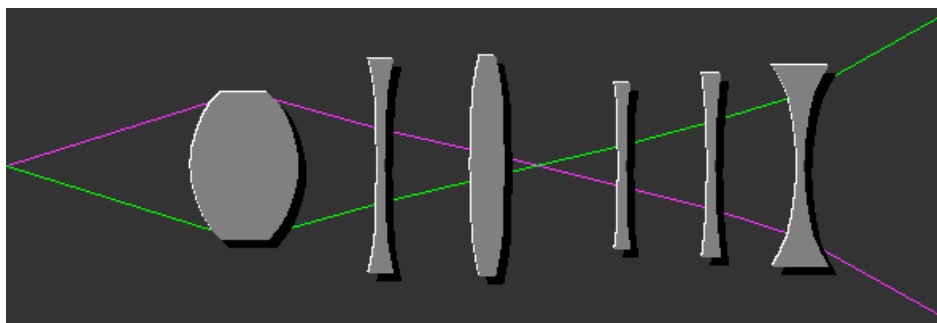
Busca no Youtube: “how its made s8 ep4- optical lenses”

<http://www.youtube.com/watch?v=F7-IfLe3-oo&feature=related>

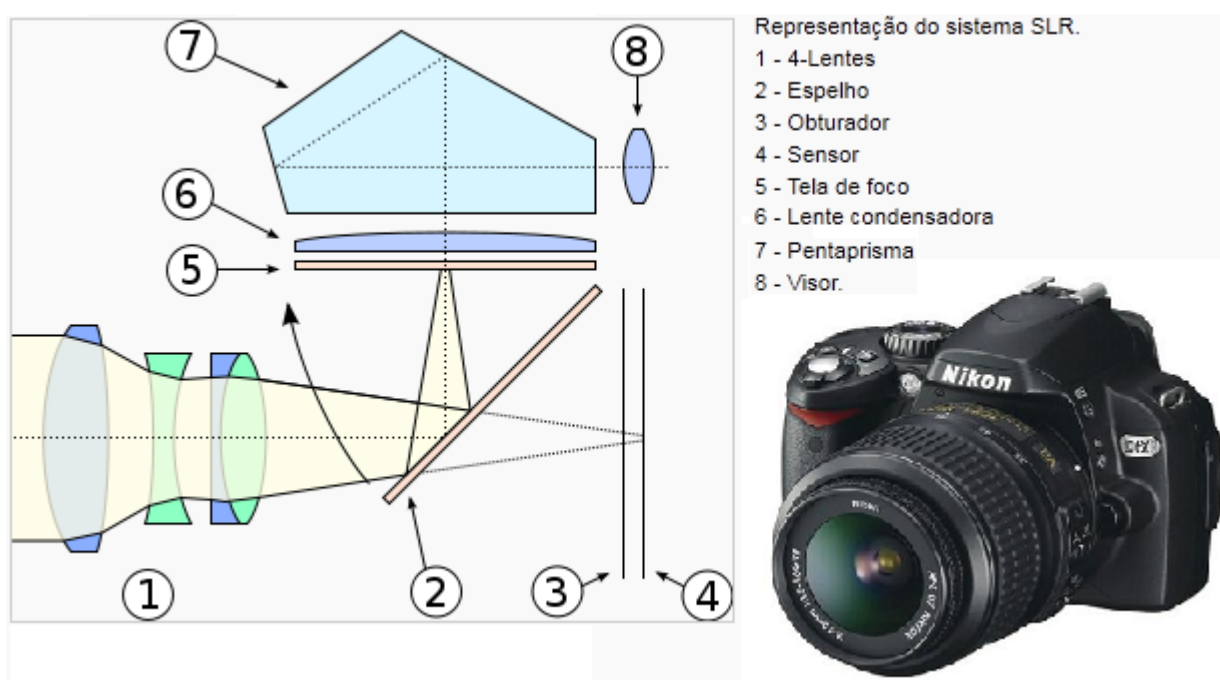
<http://www.youtube.com/watch?v=r5XMI7-xFow&feature=related>



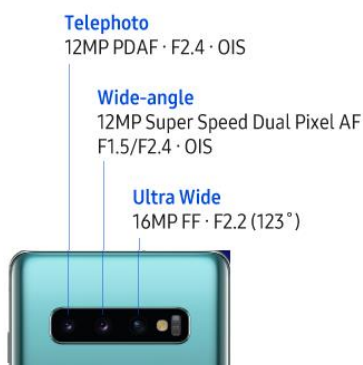
No site do pesquisador Ken Perlin (<http://mrl.nyu.edu/~perlin/experiments/demos/OpticalBench.html>) tem um applet onde se pode fazer simulações simples dos princípios de lentes côncavas e convexas.



A seguinte figura ilustra o funcionamento de uma máquina SLR.



Com a evolução dos smartphones e suas lentes, as máquinas fotográficas Point-and-shoot praticamente desapareceram. As máquinas DSLR ainda continuam sendo vendidas, principalmente para profissionais, mas em menor quantidade. Devido a limitação de espaço do smartphone, a tendência atual consiste em adicionar várias lentes ao smartphone, cada uma específica para cada tipo de fotográfica. Temos como exemplos o Galaxy S10 que possui 3 câmeras traseiras, cada uma com diferente distância focal (13mm, 26mm e 52mm). Para atingir maior distância focal, alguns fabricantes utilizam periscópios, como no caso do Oppo e do Huawei P30 (com lente de 135 mm).



Mesmo com tamanho super reduzido, a qualidade das câmeras dos smartphones continua a surpreender. O modelo Galaxy S10 conta com lente com abertura máxima f/1.5 na câmera principal, o que é um diferencial para fotografias noturnas. Deve-se observar que em máquinas profissionais, uma abertura f/2.8 já é considerada grande e torna a lente muito cara. Entretanto, como o sensor CMOS de uma DSRL é grande, pode-se aumentar muito o ISO, que no geral é muitas vezes maior que o ISO máximo disponibilizado em smartphones (800 no Galaxy S10 e 25.600 na Nikon D850, podendo chegar a 102,400). Dobrar o ISO equivale a reduzir pela metade a abertura.

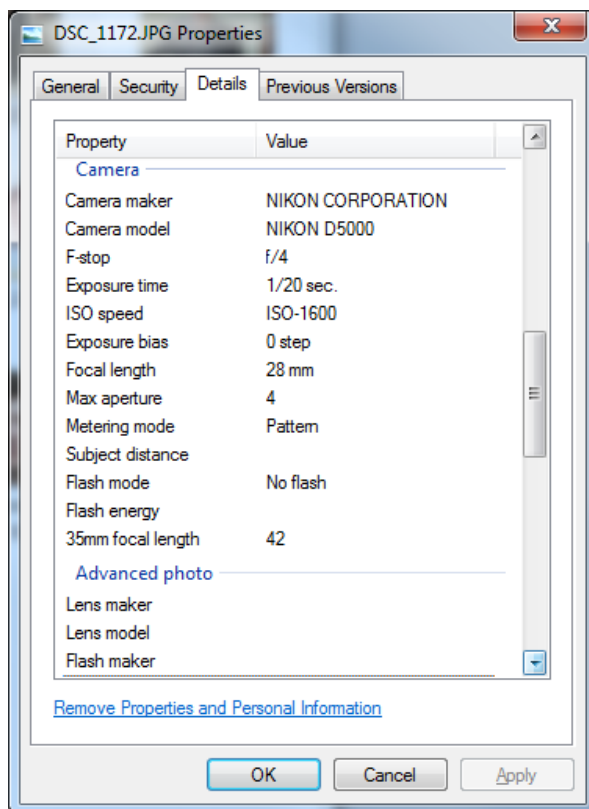
O site <https://www.dxomark.com/> faz uma análise bem detalhada da qualidade das câmeras dos smartphones e máquinas convencionais.

Rank	Device	Launch Price	Launch Date	CAMERA	SELFIE	AUDIO	DISPLAY	BATTERY
1.	Honor Magic4 Ultimate	\$1211	Mar 2022	146	-	-	95	-
2.	Huawei P50 Pro	\$907	Jul 2021	144	106	-	93	79
3.	Xiaomi Mi 11 Ultra	\$1200	Mar 2021	143	94	71	87	69
4.	Huawei Mate 40 Pro+	\$1363	Sep 2020	139	-	-	-	-
5.	Apple iPhone 13 Pro	\$999	Sep 2021	137	99	75	98	76
=	Apple iPhone 13 Pro Max	\$1099	Sep 2021	137	99	75	99	89
7.	Huawei Mate 40 Pro	\$1199	Sep 2020	136	104	-	-	-
8.	Google Pixel 6 Pro	\$899	Oct 2021	135	102	71	90	49

Rank	Device	Launch Price	Launch Date	OVERALL
1.	Hasselblad X1D-50c	\$8995	Jun 2016	102
2.	Pentax 645Z	\$8499	Apr 2014	101
3.	Leica M11	\$8350	Jan 2022	100
=	Nikon Z7II	\$3399	Oct 2020	100
=	Nikon D850	\$3300	Aug 2017	100
=	Panasonic Lumix DC-S1R	\$3700	Jan 2019	100
=	Sony A7R III	\$3200	Oct 2017	100
8.	Nikon Z7	\$3400	Aug 2018	99

O formato JPEG armazena detalhes dos parâmetros utilizados quando a fotografia foi tirada, como pode ser visto na seguinte figura:

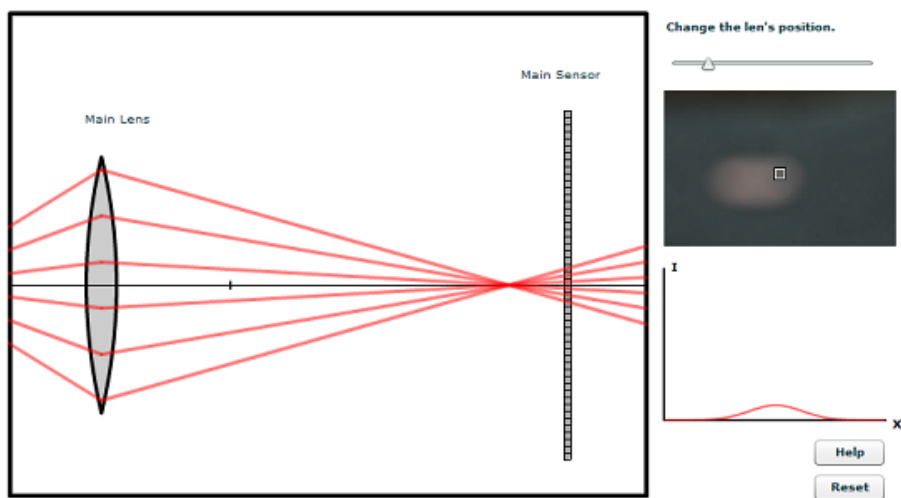


## Autofoco em máquinas Fotográficas

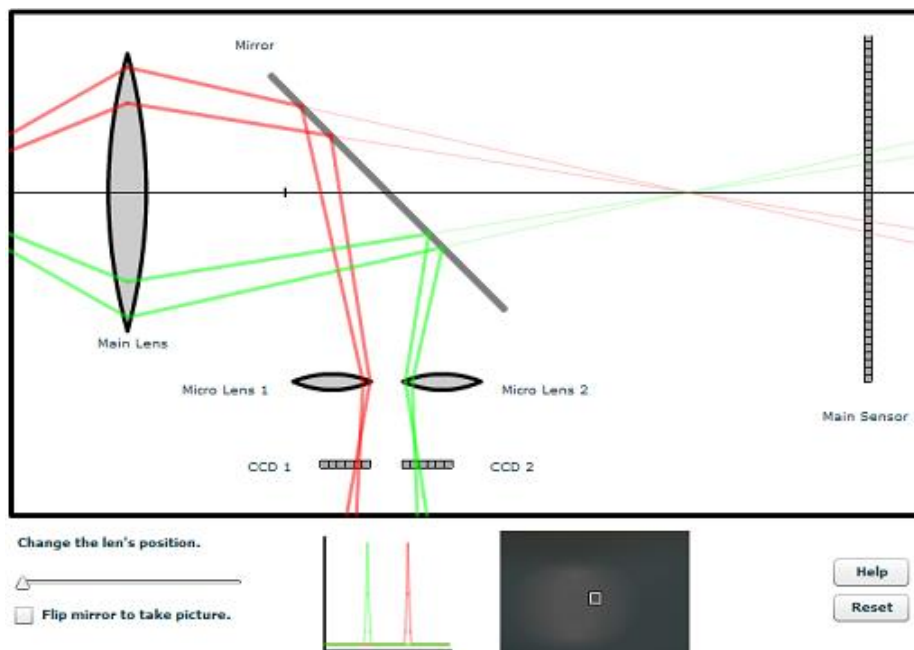
O foco automático em modo passivo, o mais utilizado atualmente, pode ser de dois tipos: *Phase detection* (máquinas SLR utilizando a tecnologia TTL (*Through-the-lens*) [<http://en.wikipedia.org/wiki/Through-the-lens>]) ou *Contrast detection* (usado em máquinas Point-and-shoot). Essas duas tecnologias são apresentadas nas próximas figuras (aplets interativos).

Por detecção de contraste, a máquina utiliza o próprio sensor para determinar o ponto máximo de contraste entre pixels vizinhos. É um processo mais demorado, mas ao mesmo tempo super preciso, pois deve mover a lente em ambas as direções (frente e para trás) para achar o ponto de contraste máximo.

A detecção por fase é mais rápida, pois analisando o sinal capturado por sensores em formato de fileiras de pixels, pode-se determinar em que direção e passo a lente deve se mover para ajustar o foco. Os valores de movimento são pré-estabelecidos pelo fabricante. Se estiverem um pouco errados ou se houver alguma desregulagem ou desgaste, a máquina perde a precisão do foco.



<http://graphics.stanford.edu/courses/cs178/applets/autofocusCD.html>



<http://graphics.stanford.edu/courses/cs178/applets/autofocusPD.html>

Em 2011, a Sony lançou dois modelos (alpha 33 e alpha 55) de máquina SLR-like, que com o uso de um espelho translúcido consegue ter as vantagens do autofocus por detecção de fase e ao mesmo tempo modo Liveview. Porém, esse modelo não tem viewfinder óptico.



## 4 Preenchimento de Polígonos

Uma vez que os polígonos já passaram pelo estágio de visualização (transformação de câmera), estes devem ser pintados na tela. Para isso, o algoritmo z-buffer é o mais utilizado (inclusive na API OpenGL). O trabalho consiste em mostrar na tela somente os polígonos visíveis (ou parte deles).

O algoritmo itera sobre cada polígono individualmente, um após o outro, até que todos os polígonos dentro do volume de visão da câmera sejam processados. Para cada polígono, com o uso de um algoritmo de scan-line, determina-se todos os pixels que fazem parte do polígono. Se o pixel for mais próximo da câmera (menor z) que o z atual, este deve sobrepor o antigo, por isso o algoritmo se chama z-buffer. O algoritmo utiliza dois buffers: um para cor de cada pixel e um para a profundidade de cada pixel.

O seguinte algoritmo ilustra o processo. No algoritmo, x e y representam a dimensão da viewport.

```

Lista de polígonos {P1, P2, ..., Pn}
Matriz Z-buffer[x,y] inicializada com ∞
Matriz Cor[x,y]
para cada polígono P na lista de polígonos faça
{
  para cada pixel (x,y) na projeção de P faça
  {
    calcule prof-z de P na posição (x,y)
    se prof-z < Z-buffer[x,y] então
    {
      Cor[x,y] = cor de P em (x,y)
      Z-buffer[x,y] = prof-z
    }
  }
}

```

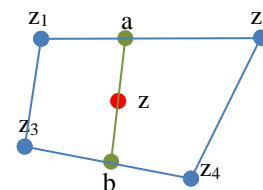
### Interpolação Bilinear

Dados 4 vértices em 3D, um ponto z é dado por:

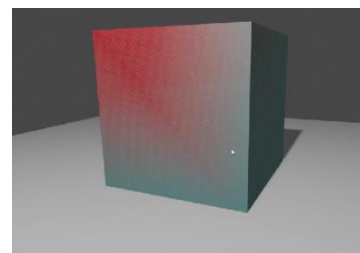
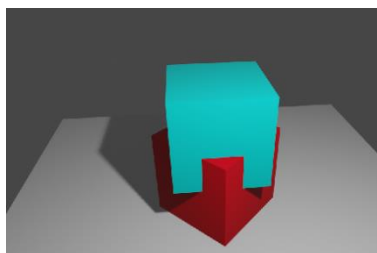
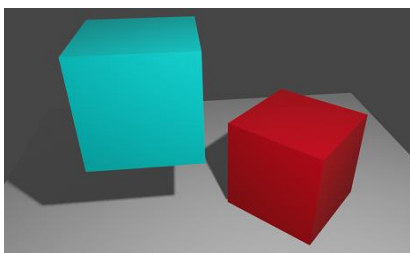
$$a = z_1 * (1-t) + z_2 * t$$

$$b = z_3 * (1-t) + z_4 * t$$

$$z = a * (1-s) + b * s$$



Deve-se observar que o algoritmo torna-se mais eficiente se os polígonos mais próximos da câmera forem processados primeiro. Abaixo, na figura a direita, o problema de z-fighting.



# 5 Representação e Modelagem

## Níveis de realismo de representação

- Pontos
- Linhas (*wireframe*)
- Faces (preenchimento)
  - Iluminação
  - Textura
  - Bump mapping
  - Environment Mapping

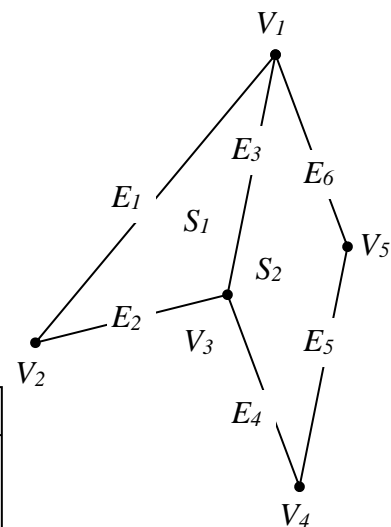
### 1. Faces (Polígono)

- Tabela de vértices (*vertex table*) [Hearn, pg 307]
- tabela de arestas (*edge table*)
- Tabela de superfície (*polygon-surface table*)

Vertex Table
$V_1: x_1, y_1, z_1$
$V_2: x_2, y_2, z_2$
$V_3: x_3, y_3, z_3$
$V_4: x_4, y_4, z_4$
$V_5: x_5, y_5, z_5$

Edge Table
$E_1: V_1, V_2$
$E_2: V_2, V_3$
$E_3: V_3, V_1$
$E_4: V_3, V_4$
$E_5: V_4, V_5$
$E_6: V_5, V_1$

Face Table
$S_1: E_1, E_2, E_3$
$S_2: E_3, E_4, E_5, E_6$



#### Vantagens:

- Esta organização permite que o objeto seja visualizado como pontos, linhas ou faces.
- Evita a duplicação de vértices no caso do uso de uma única tabela

#### Observações:

- Para se trabalhar com faces deve-se fazer uso de algoritmos de preenchimento de polígonos, como *z-buffer* ou *scan-line*.
- Para se obter superfícies com realismo (no mínimo iluminação), deve-se dispor de algoritmos que façam cálculos de iluminação, aplicados sobre cada pixel de cada face.

Outra abordagem: Armazenar na tabela de arestas as faces que a aresta é comum. Isso traz vantagens na aplicação de algoritmos de iluminação (renderização), onde deve haver uma transição suave de cor na transição entre faces que possuem arestas em comum.

Edge Table
$E_1: V_1, V_2, S_1$
$E_2: V_2, V_3, S_1$
$E_3: V_3, V_1, S_1, S_2$
$E_4: V_3, V_4, S_2$
$E_5: V_4, V_5, S_2$
$E_6: V_5, V_1, S_2$

## Técnicas de Geração de objetos 3D

As técnicas apresentadas a seguir são usadas para gerar primitivas (vértices) que definem objetos 3D, que podem ser simples esferas bem como complexas superfícies.

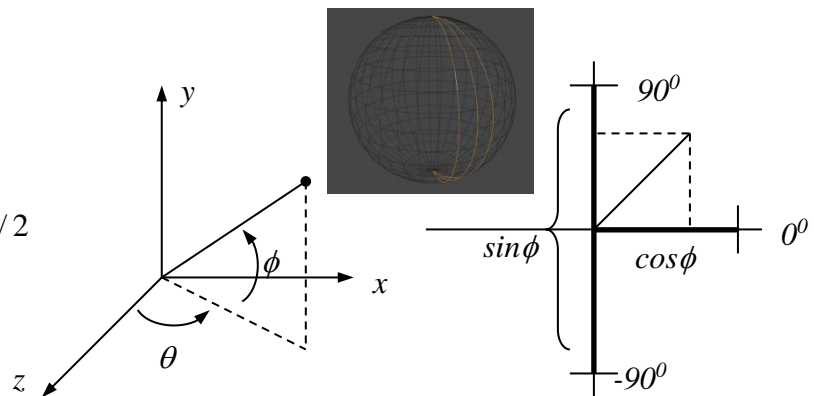
Deve-se observar que a partir da malha de pontos gerada, deve-se definir estruturas de arestas e faces, com mostrado na técnica anterior, para possibilitar a visualização do objeto de forma mais realista. Caso contrário, este poderia ser observado somente como uma coleção de pontos, que em muitos casos não é suficiente para descrever a geometria real do objeto.

Cabe como exercício ao aluno implementar algoritmos de geração da malha para cada modelo apresentado.

### Esfera

Em coordenadas esféricas temos

$$\begin{aligned} x &= r \cos \theta \cos \phi & -\pi/2 \leq \phi \leq \pi/2 \\ z &= r \sin \theta \cos \phi & -\pi \leq \theta \leq \pi \\ y &= r \sin \theta \sin \phi \end{aligned}$$



onde

- $\sin \phi$ , no intervalo especificado, varia de -1 a 1
- $\cos \phi$ , no intervalo especificado, varia de 0 a 1
- o termo  $\cos \phi$ , nas equações de  $x$  e  $z$ , altera o raio de cada círculo concêntrico que define a esfera.

Deve-se observar que este algoritmo faz uso de duas variáveis paramétricas, logo o algoritmo de renderização deve ter a seguinte estrutura.

```
for( $\theta = 0; \theta \leq 2\pi; \theta \pm step$ )
```

```
{
```

```
    for( $\phi = 0; \phi \leq \pi; \phi \pm step$ )
```

```
    {
```

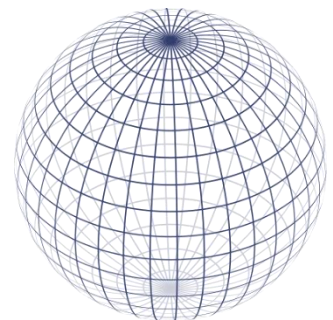
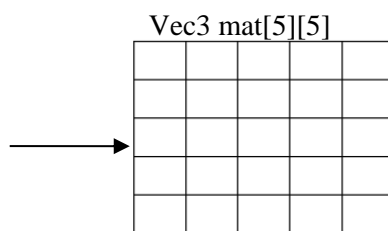
```
        x =
```

```
        y =
```

```
        z =
```

```
    }
```

```
}
```



Este algoritmo alimenta uma matriz quadrada (reticulado) de dimensão  $M \times N$ , em função do parâmetro *step*. Quanto menor for o *step*, maior deverá ser a dimensão da matriz. Cada coluna desta matriz contém uma sequência de pontos que representa uma “fatia” do objeto construído. Cada elemento da matriz contém as 3 coordenadas ( $x, y, z$ ) de cada vértice do objeto gerado. Esta abordagem é muito mais simples do que criar listas de vértices, arestas e faces.

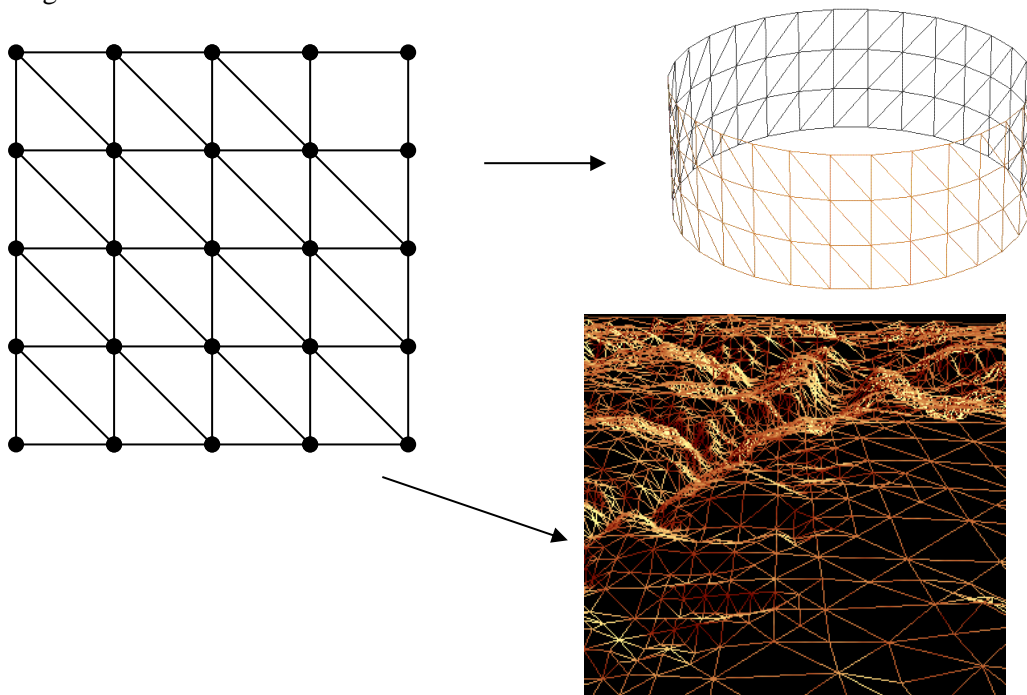
Uma vez construída a matriz, pode-se fazer a exibição do objeto da tela. Para isso necessita-se de um algoritmo de triangulação, de modo a gerar uma malha triangular para os pontos gerados e assim definir a geometria do objeto. Vamos considerar aqui um caso simples, onde esta matriz representa um terreno, para evitar o tratamento de ligações circulares. O seguinte pseudo-algoritmo trata deste problema

```

for(l=0; l<M; l++)
{
    for(c=0; c<N; c++)
    {
        line(M[l,c], M[l+1,c]); //linha horizontal
        line(M[l,c], M[l,c+1]); //linha vertical
        line(m[l,c], M[l+1,c+1]); //linha diagonal
    }
}

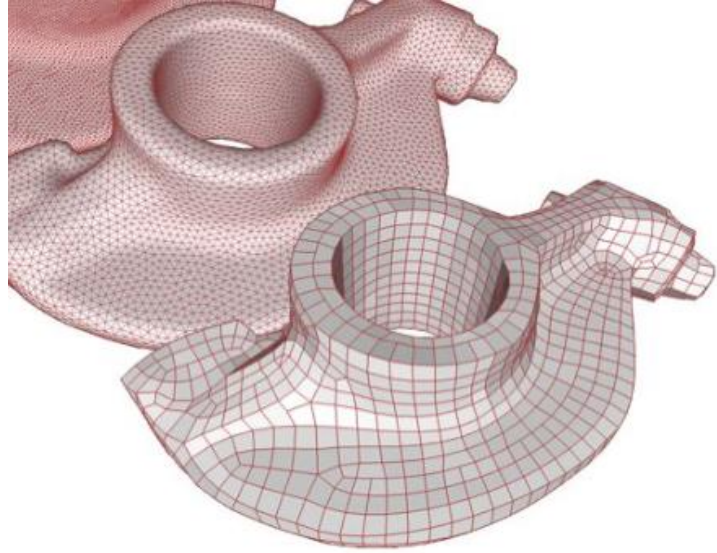
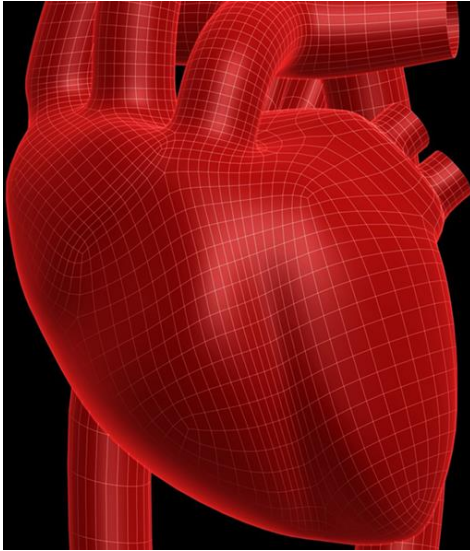
```

O resultado do processo é uma malha triangular como mostrado na seguinte figura. Fazendo-se uso de uma câmera sintética, pode-se fazer a visualização do objeto sob qualquer ângulo e se desejar, em perspectiva. Essa mesma abordagem de criação de uma malha triangular pode ser usado na renderização de superfícies paramétricas e também na técnica de *sweep*, apresentadas na sequência, de forma transparente. O terreno mostrado apresentado uma estrutura mais complexa do que o cilindro, mas mesmo assim possui uma geometria bastante regular.



Existem diversas aplicações que necessitam de modelos mais genéricos, que não podem ser representados por **reticulados**, como mostrado nas seguintes imagens. Para isso, o uso de técnicas de modelagem como half-edge (ver última seção) faz-se necessário.





## Elipsóide

Para a elipsóide, pode-se ter até 3 raios diferentes.

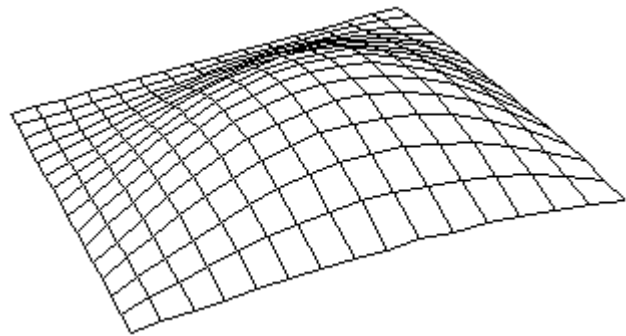
$$\begin{aligned}x &= r_x \cos \phi \cos \theta & -\pi/2 \leq \phi \leq \pi/2 \\z &= r_z \cos \phi \sin \theta & -\pi \leq \theta \leq \pi \\y &= r_y \sin \phi\end{aligned}$$

## Super Elipsóide

$$\begin{aligned}x &= r_x \cos^{s_1} \phi \cos^{s_2} \theta & -\pi/2 \leq \phi \leq \pi/2 \\z &= r_z \cos^{s_1} \phi \sin^{s_2} \theta & -\pi \leq \theta \leq \pi \\y &= r_y \sin^{s_1} \phi\end{aligned}$$

## Superfícies biparamétricas

Fazem uso de 16 pontos de controle para a definição de um *patch*.



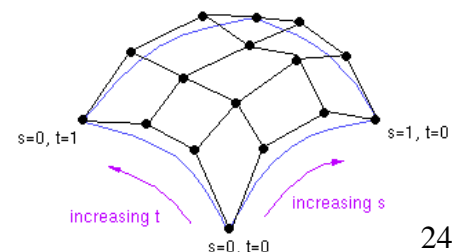
## Superfície Bézier (*Bézier surface patch*)

$$P(s, t) = \sum_{i=0}^m \sum_{j=0}^n B_{ij} J_{i,n}(s) J_{j,m}(t), \quad 0 \leq s, t \leq 1$$

$$J_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}$$

Para polinômios de grau 3, tem-se as seguintes funções base:



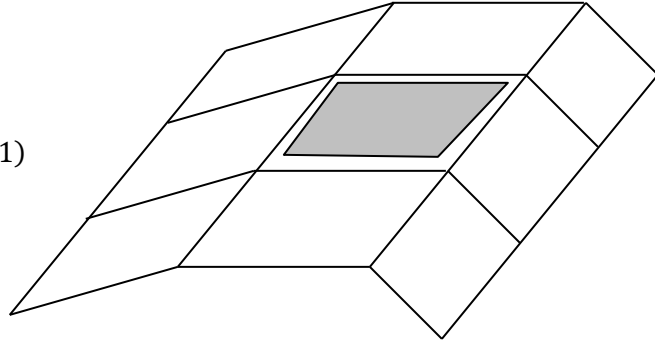


$$\begin{aligned}
(1-t)^3 & \text{ (valor máximo em } t=0) \\
3t(1-t)^2 & \text{ (valor máximo em } t=1/3) \\
3t^2(1-t) & \text{ (valor máximo em } t=2/3) \\
t^3 & \text{ (valor máximo em } t=1)
\end{aligned}$$

### Superfície Spline (B-Spline Surface Patch)

$$Q(s, t) = \sum_{i=0}^3 \sum_{j=0}^3 P_{ij} B_i(s) B_j(t), \quad \sum_{i=0}^m B_i(t) = 1$$

$$\begin{aligned}
B_0 &= \frac{1}{6} (1 - t^3) \\
B_1 &= \frac{1}{6} (3t^3 - 6t^2 + 4) \\
B_2 &= \frac{1}{6} (-3t^3 + 3t^2 + 3t + 1) \\
B_3 &= \frac{1}{6} t^3
\end{aligned}$$



No caso de superfícies *spline*, apenas uma fração da área definida pelos pontos de controle é gerada, e nenhum dos 16 pontos de controle é interpolado.

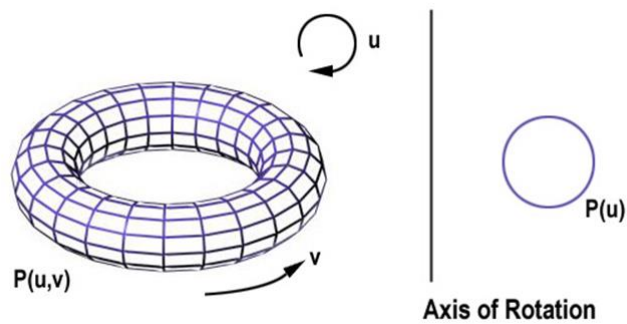
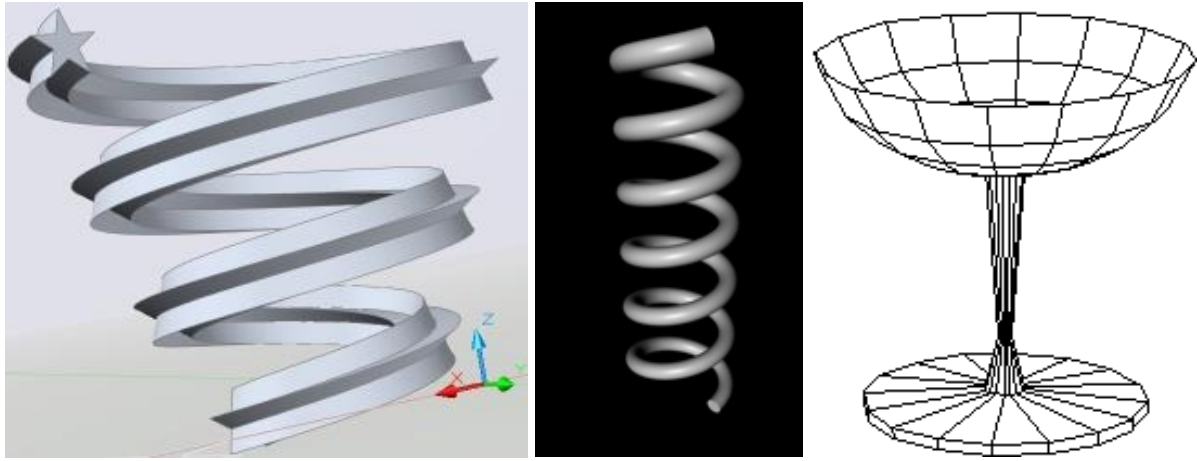
### Renderização de superfícies paramétricas curvas [watt pg 197]

- Renderizar diretamente a partir da descrição paramétrica;
- Podem ser representadas por um reticulado de pontos 3D
- Aproximar a superfície por uma malha poligonal e utilizar algoritmos de preenchimento de polígonos para renderizar esta aproximação.
  - Abordagem mais simples e
  - Mais utilizada atualmente.

### Sweep (Varredura)

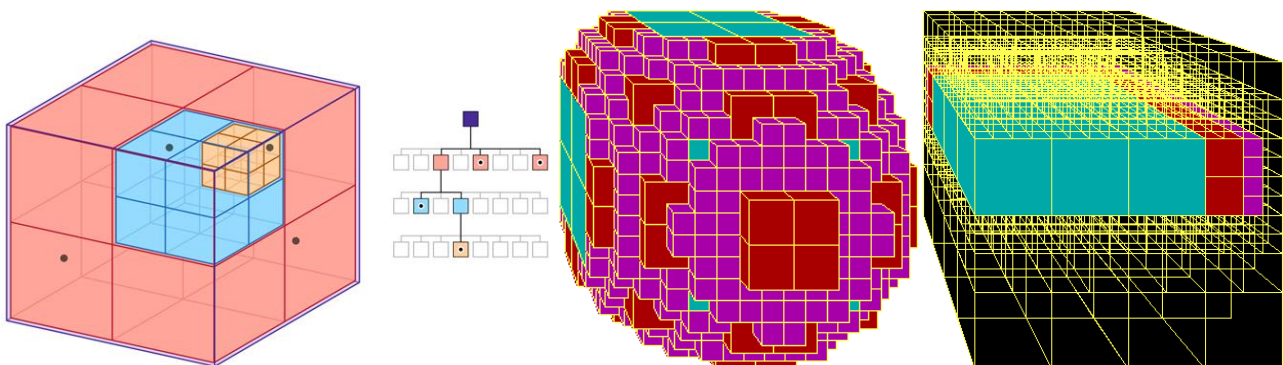
Definidos a partir de uma figura 2D (*shape*) e uma varredura que move a primitiva por uma região no espaço. Pode consistir de rotação, translação e objetos gerados por esta técnica podem ser representados por um reticulado de pontos 3D.

- *Sweep Transacional*: Pode-se utilizar um *path* para determinar a trajetória da figura base.
  - a. Pode gerar cilindros, por exemplo.
- *Sweep Rotacional*: Aplica sobre a forma base rotações sobre eixos.
  - a. Pode gerar formas cilíndricas como anéis, copos, vasos, etc. **OBS: a equação de geração da esfera pode ser vista como um *sweep* rotacional.**
- *Combinação dos dois*: Pode-se usar translação e rotação para definir o objeto.
  - a. Usado para gerar molas.

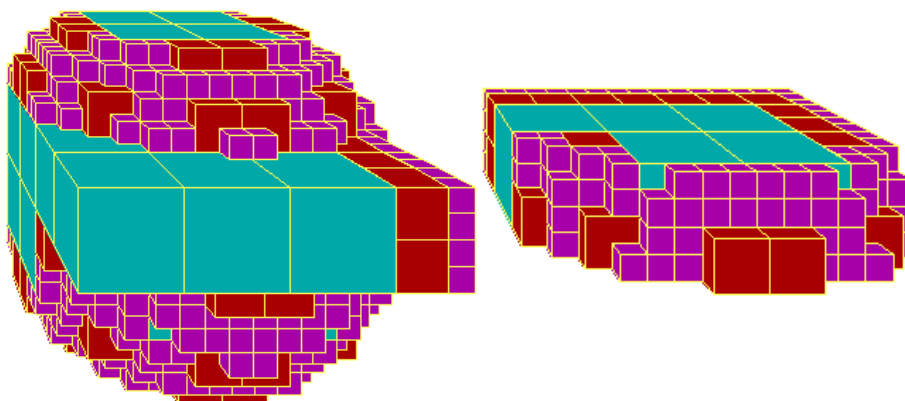


## Octree

- Consiste na subdivisão de um sólido em 8 quadrantes.
- Como o nome sugere, faz uso de uma árvore onde cada nó tem 8 ponteiros (1 para cada quadrante). Cada quadrante pode ser dividido recursivamente até que se chegue na resolução desejada.
- Cada nó na árvore poder ser *full*, *empty* ou *partial*. Essa informação pode ser utilizada para determinar o nível de refinamento de um ramo da árvore.
- Utiliza-se a palavra **voxel** para descrever cada elemento volumétrico que define o objeto em uma octree. Tem o mesmo significado que um pixel em uma imagem.



- Pode-se facilmente aplicar operações booleanas sobre dois objetos definidos por uma octree.



## CSG

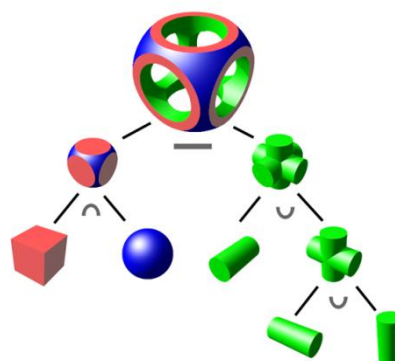
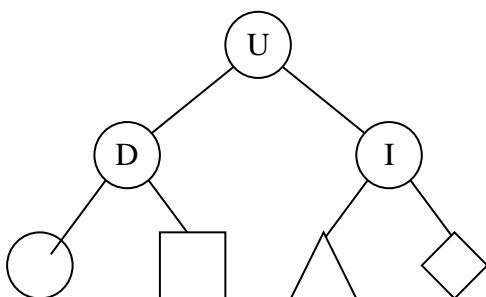
A representação de objetos sólidos pode ser obtida por várias técnicas, como por octree, CSG, dentre outras. Uma forma de classificar a técnica é pela forma como os objetos são representados. Com o uso de *octrees*, faz-se uma decomposição do objeto em primitivas básicas, como o cubo, e por isso, operações booleanas entre objetos são simplificadas. Entretanto, esta representação não oferece alto grau de precisão, principalmente quando aplicada a objetos curvos.

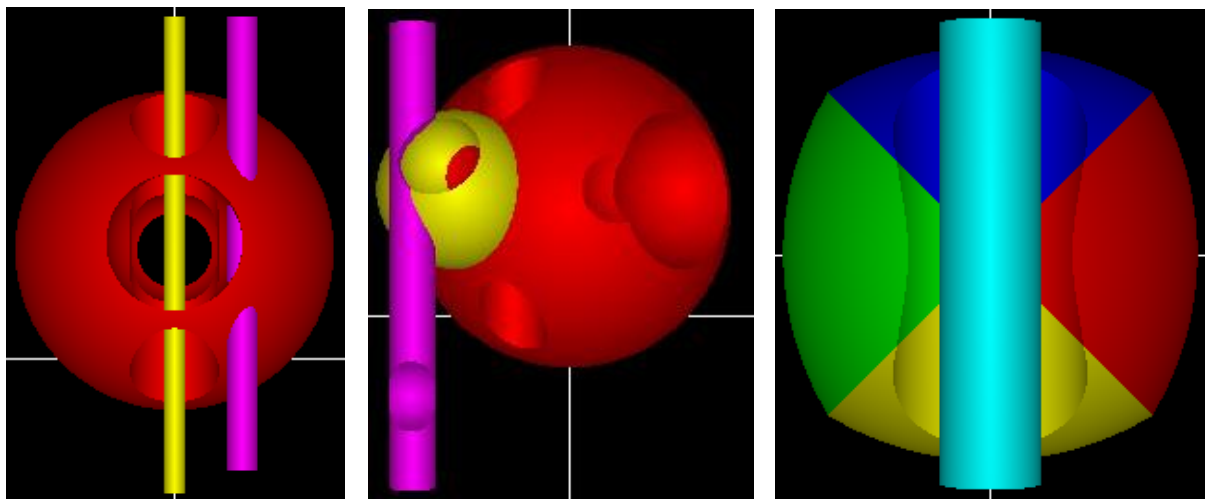
A representação de sólidos por CSG (*Constructive Solid Geometry*), que é um modelo construtivo, se caracteriza por uma representação com alta precisão, pois os objetos não são mais definidos por primitivas (cubos) e sim por equações. O fato de se trabalhar com a descrição precisa do objeto implica em um aumento de complexidade, tanto na realização das operações booleanas, bem como na visualização do mesmo.

O termo construtivo vem do fato que objetos complexos são obtidos por operações booleanas entre objetos mais simples, como esferas, cubos, cilindros, etc. Estes objetos simples são armazenados em uma árvore, que também pode conter transformações, como escala, rotação, translação e que oferece ao usuário a capacidade de edição do sólido que será gerado. A representação da CSG exige uma estrutura de dados que possam organizar, de forma hierárquica, as primitivas e seus relacionamentos. Esta hierarquia é definida por uma árvore binária onde as primitivas só podem ser armazenadas nas folhas e as operações nos demais nós.

A árvore CSG define um conjunto de primitivas de forma hierárquica, onde a cada nível da árvore, operações booleanas e/ou transformações como rotação e escala podem ser aplicadas às primitivas para que sua combinação possa gerar o sólido desejado.

Para a obtenção do sólido gerado, deve-se traçar retas que interceptam os elementos definidos na árvore, bem como operações de união, interseção e diferença.

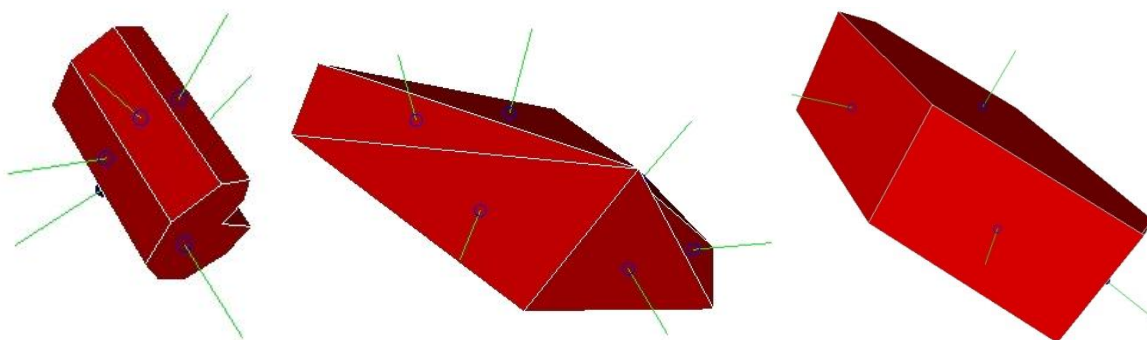




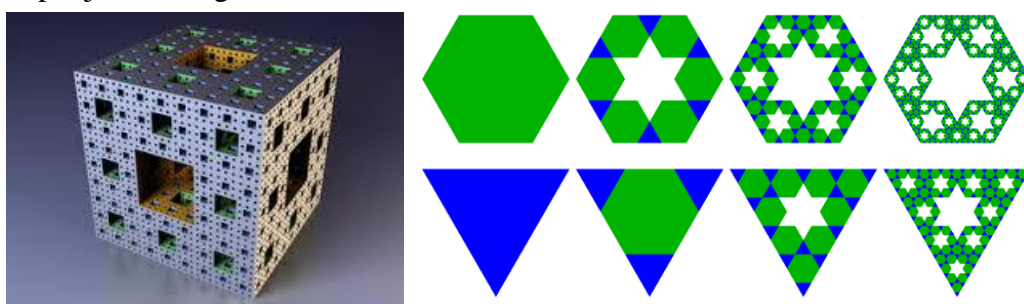
### Outras técnicas de modelagem de objetos

- Winged-edge
- Half-edge
- Fractal
- Sistemas de Partículas

### Half-Edge



### Esponja de Menger



## Referências Bibliográficas

- [1] Hearn, D., Baker, M. P. **Computer Graphics, C Version** (2<sup>nd</sup> Edition), Prentice Hall, New Jersey, 1997
- [2] Watt, A. **3D Computer Graphics**, Addison Wesley; 3 edition (December 6, 1999)
- [3] Rogers, D., Adams, J. **Mathematical Elements for Computer Graphics**, 2<sup>nd</sup> Edition. Mc Graw Hill, 1990.
- [4] Fiume, E. L. **The Mathematical Structure of Raster Graphics**. San Diego, CA. Academic Press, 1989.