1 *Pipeline* de Visualização Tridimensional

1.1 Introdução

O processo básico de geração de imagens tridimensionais por computador para aplicações em tempo real ou interactivas, amplamente conhecido por ***pipeline* de visualização**, encontra-se perfeitamente normalizado [Foley90]. Trata-se de uma estratégia de visualização que proporciona um compromisso razoável velocidade de desenho versus realismo. O propósito desta técnica consiste em criar uma **projecção 2D** da cena tridimensional, tomando em linha de conta a iluminação e a perspectiva de visualização. O processo assume como entrada uma cena tridimensional, consistindo de objectos descritos por polígonos[1], fontes de luz e parâmetros de visualização, e à qual vai ser aplicada uma sequência de andares de cálculo:

1. Atravessar a base de dados que contém o modelo matemático da cena;
2. Aplicar o modelo de reflexão local de acordo com o algoritmo de sombreamento;
3. Aplicar as transformações geométricas de acordo com a posição e a direcção de um observador
4. Eliminar as faces escondidas (operação designada por *back-faces culling*);
5. Executar o recorte e opcionalmente a transformação perspectiva;
6. Calcular os pixéis que irão ser activados (*scan-conversion*);

[1] Na realidade, a maioria das APIs gráficas ou programas modeladores possibilitam a modelação de cenas utilizando primitivas gráficas de alto nível (ex: sólidos 3D) o que implica a existência de um andar de pré-processamento responsável pela sua decomposição em polígonos.

7. Calcular a cor de cada um dos pixéis (algoritmo incremental de sombreamento);
8. Remover as superfícies ocultas;
9. Implementar eventuais efeitos visuais (ex: mapeamento de texturas, transparências, sombras);
10. Filtrar os ruídos (*anti-aliasing*);
11. Escrever a informação respeitante aos pixels na memória de imagem para visualização (implementa implicitamente a projecção 2D).

O fluxo de actividade entre o processo de atravessamento da base de dados da cena e a visualização da imagem num ecrã pode ser agrupado em dois grandes blocos funcionais distintos: o bloco dos **cálculos geométricos** (passos 2, 3, 4 e 5) que fundamentalmente converte as primitivas gráficas descritas em coordenadas do Mundo em primitivas descritas em coordenadas de ecrã, e o bloco de **rasterização** (passos 6, 7, 8, 9, 10) que converte informação geométrica em pixels no ecrã com informação de cor. O processamento geométrico opera no Espaço-Objecto e a rasterização opera no Espaço-Imagem.

Como se referiu anteriormente, a visualização de cenas tridimensionais num ecrã bidimensional implica a execução de uma projecção. Saliente-se que a execução dos passos 1 a 10 envolve operações tridimensionais, ou seja, operações onde é necessário conhecer obrigatoriamente as três coordenadas, como, por exemplo, a profundidade, para poder resolver o problema da visibilidade. Deste modo, a projecção é realizada implicitamente apenas no passo 11 quando se guarda a informação de cor dos pixéis, pois só aqui se acede à informação em x e y, através do acesso à memória de imagem.

São diversos os sistemas de visualização comerciais que recorrem a esta técnica de síntese de imagem baseada no *pipeline* 3D: OpenGL, Direct3D, Renderware, apenas para referir os mais conhecidos actualmente. De um modo geral, qualquer destes sistemas, frequentemente apelidados de APIs (*Application Progamming Interface*) Gráficas, disponibiliza uma biblioteca gráfica que constitui um ambiente bastante versátil para a modelação e visualização de objectos complexos.

Geralmente, as bases de dados de cenas para os programas de visualização em tempo real ou interactivos contêm a especificação dos parâmetros de visualização, das fontes de luz, dos atributos físicos das superfícies (cor, índices de reflexão, etc.) e da informação geométrica dos polígonos. Quanto a este último aspecto refira-se que, na busca de um maior realismo, esses programas suportam ainda primitivas de alto nível na descrição da cena, ou seja, formas geométricas mais complexas que incluem, entre outras, arcos de curvas, superfícies curvas de diversos tipos e superfícies quadráticas.

Por isso, as bibliotecas gráficas incluem tradicionalmente funções específicas para a modelação de cenas: definição de primitivas gráficas de alto nível (cubo, prisma, esfera, elipsóide, cilindro, cone, *torus*, etc.) e a especificação de parâmetros de visualização, de fontes luminosas e características de superfícies. A utilização de primitivas de alto nível em aplicações interactivas pressupõe, na maioria das vezes, que estas sejam primeiramente convertidas em polígonos ou malhas de polígonos - operação designada vulgarmente por *tesselation* [Foley90] - e, depois, injectadas no *pipeline* de visualização.

Outra questão relacionada com a base de dados prende-se com o modo como a informação se encontra estruturada. A maioria das *interfaces* suporta uma modelação

hierárquica na descrição da cena [Foley90]. Cada nó da árvore[2] possui uma especificação das primitivas (por exemplo, polígonos), atributos que definem o aspecto das primitivas (por exemplo, cor), matrizes de transformação que posicionam as primitivas no espaço 3D e, eventualmente, referências para outros nós. Os atributos e as transformações são herdados pelos nós inferiores. Quando a árvore se mantém entre as diferentes tramas (ou quadros), diz-se que a interface suporta o **modo de retenção** ou *display-list*. Quando a árvore é definida pela execução de um programa em *stack*, e não se mantém entre as tramas, então a interface suporta o **modo imediato**.

A obtenção de desempenhos elevados nas arquitecturas *raster* é condicionada fundamentalmente por três parâmetros: o número de cálculos em vírgula-flutuante no bloco geométrico, as operações associadas com a computação dos valores dos pixéis (operações envolvendo arimética inteira) e o número de acessos à memória de imagem durante o processo de rasterização.

Partindo do pressuposto de que uma cena "típica" contém cerca de 10000 polígonos, [Molnar90] estima o número de operações efectuadas em cada um dos blocos do *pipeline* de visualização. A título de exemplo, um sistema *raster*, para ser capaz de gerar imagens de resolução 1280 x 1024 a uma taxa de 25 tramas/segundo, teria que realizar cerca de 92 milhões de flops (*floating-point operations per second*) no bloco geométrico, e 106 milhões de operações inteiras e 127 milhões de acessos à memória de imagem por segundo no bloco de rasterização. Sublinhe-se que estes valores respeitam uma cena modesta com apenas 10000 polígonos.

1.2 Arquitectura global

O diagrama de blocos que define a arquitectura global de um sistema gráfico típico está ilustrado na figura 1.1. Refira-se que algumas das operações poderão ser substituídas por outras, como por exemplo a Transformação Perspectiva, ou poderão não ser realizadas, como por exemplo a triangulação.

[2]A hierarquia utilizada na modelação das cenas costuma ser simbolizada por um grafo denominado de DAG (*Directed Acyclic Graph*) [Foley90].

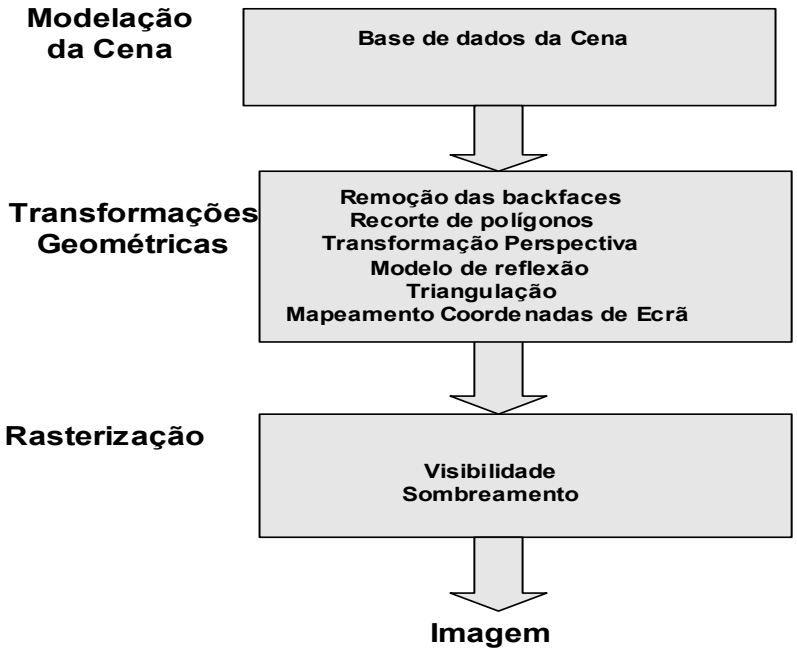


Figura 1.1 - Diagrama de blocos dum sistema gráfico típico.

Como foi referido anteriormente, o tradicional sistema gráfico baseado num pipeline de visualização é constituído por três grandes blocos:

- as operações de construção da cena;
- as operações geométricas que basicamente actuam sobre os vértices dos polígonos e que incluem as operações de pré-remoção de polígonos não visíveis (*Culling* ou *Backface Polygon Removal*), o recorte de polígonos (*Clipping*), a aplicação da transformação perspectiva, a determinação da luminosidade por aplicação do modelo de iluminação, a segmentação opcional (*Polygon Splitting*) de polígonos em triângulos (ou triangulação), e a operação de mapeamento nas coordenadas físicas (inteiras) do dispositivo de visualização (*Device Dependent Coordinates*);
- a rasterização responsável pelo cálculo dos pixéis, remoção de superfícies ocultas (*Hidden Surface Removal*), e sombreamento (*Shading*);

Os sistemas gráficos facultam ainda algumas características adicionais que podem ser essenciais na sua função de plataforma genérica para suporte de visualização de polígonos. Nomeadamente:

- juntar aos métodos de sombreamento *Phong* e *Gouraud* a possibilidade de visualizar os objectos com sombreamento constante em cada polígono *(Flat Shading)* ou simplesmente em modelo de "arame";
- aplicação de texturas;
- implementação de efeitos sombra e de transparência;
- execução de algoritmos de *anti-aliasing*.

1.3 Sistemas Coordenados

Apesar de a maioria das APIs gráficas facultarem a um utilizador a capacidade de descrição de cenas de complexidade elevada através do recurso a primitivas gráficas de alto nível, estas são "pré-processadas" pelo sistema, de forma a serem modeladas em termos de polígonos. O sistema permite exclusivamente o processamento de objectos descritos sob a forma de conjuntos de polígonos planares, convexos ou côncavos e sem buracos. Este processamento implica, em primeiro lugar, a aplicação de um conjunto de transformações geométricas às quais se encontram associados os respectivos sistemas de coordenadas.

De forma a facilitar a modelação de um determinado objecto, opta-se geralmente por descrever os seus diversos componentes (isto é, especificar os vértices dos polígonos que o constituem) em função de um ponto localizado dentro ou perto do próprio objecto. Nesta situação, os objectos dizem-se descritos em termos de um **Sistema de Coordenadas Local** (ou *Local Coordinate System).*

Após a modelação, o passo seguinte consiste em colocar o objecto na **cena** que se pretende visualizar. Cada objecto que constitui a cena tem o seu referencial local. Todos os objectos são transformados de forma a poderem ser representados em função de um referencial único, o que permite especificar o seu posicionamento relativo. Este referencial comum é denominado **Sistema de Coordenadas do Mundo** (ou *World Coordinate System*). Ainda neste sistema de coordenadas, a descrição da cena é complementada com a especificação das posições das fontes luminosas e da posição da câmara, em particular, do ponto de observação e da direcção de observação.

Em Computação Gráfica, recorre-se normalmente ao conceito de **câmara virtual** [Watt93] para definir um novo referencial, denominado **Sistema de Coordenadas da Câmara** *(Eye or Camera Coordinate System*), cuja origem coincide com o centro de projecção (posição da câmara) e cujo eixo dos ZZ é normal ao plano de visualização (*View Plane*) e representa a direcção de observação.

Por fim, o último espaço de coordenadas em que os objectos são transformados (talvez o menos intuitivo) está relacionado com as técnicas de projecção dos objectos da cena no plano de visualização. Define-se, nesse sentido, um **volume de visualização** *(Viewing Frustum)* que delimita o volume do espaço que será visualizado.

A topologia do volume de visualização depende do tipo de projecção que o utilizador pretende. No caso de se optar pela projecção perspectiva, utilizar-se-á uma operação com carácter projectivo que transforma um espaço 3D noutro espaço 3D, a **Transformação Perspectiva**. Os objectos resultantes são expressos em termos de Coordenadas de Ecrã *(Screen Coordinates).* Em rigor, as coordenadas resultantes desta transformação são normalizadas pelo que se impõe um mapeamento destas no referencial determinado pelo dispositivo físico de visualização *(Device Dependent Coordinates).*

1.4 Modelação da cena

Uma cena é composta por objectos, parâmetros de visualização e fontes luminosas. Um determinado objecto complexo pode ser modelado recorrendo a outros objectos de maior simplicidade, ou seja, um objecto pode referenciar sub-objectos. A forma natural de armazenar a informação respeitante aos objectos que compõem uma determinada cena é sob a forma de uma estrutura de dados em árvore, em que cada nó representa um objecto.

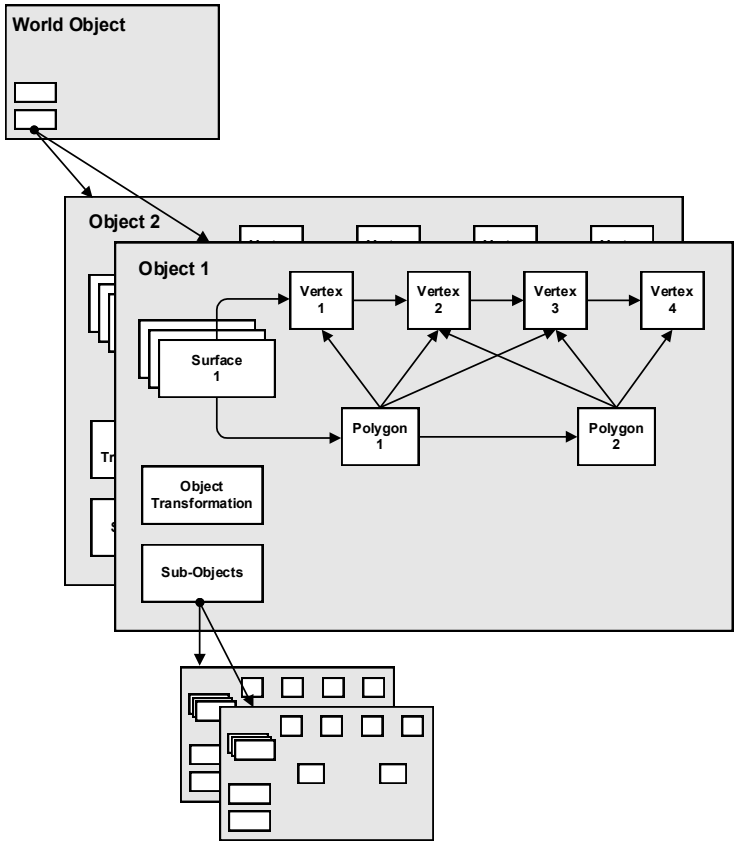


Figura 1.2 - Estrutura hierárquica dos objectos na cena.

Cada objecto é por sua vez definido por uma lista de superfícies que, para além das suas características físicas (côr, opacidade, parâmetros de iluminação, etc.), contém uma lista de polígonos e uma lista de vértices. A estrutura de dados é optimizada de modo a permitir a partilha de vértices de polígonos de uma mesma superfície. Deste facto, advêm economias óbvias, no que respeita ao volume de informação a armazenar.

Com o objectivo de facilitar as tarefas de modelação da cena, os objectos são definidos em termos de coordenadas locais, aos quais se associam transformações geométricas (translações, rotações, etc.) que permitem definir o seu posicionamento relativamente ao referencial universal. Estas transformações, representadas internamente em termos de matrizes, são conjugadas e armazenadas como parte

integrante da estrutura de dados do objecto, sendo aplicadas posteriormente aquando do processamento geométrico do objecto. Cada objecto mantém na sua estrutura de dados uma matriz de transformação relativamente ao seu objecto superior em termos de hierarquia. Qualquer transformação geométrica aplicada a um objecto propaga-se de forma recursiva a todos os seus sub-objectos. Neste sentido, a estrutura hierárquica de objectos é percorrida e são aplicadas as transformações geométricas, associadas a cada objecto, que permitem converter as coordenadas locais nas coordenadas do Mundo da cena. É de realçar, novamente, que se trata de um processo recursivo já que as transformações de coordenadas efectuadas a nível dos objectos de topo afectam os respectivos sub-objectos.

Refira-se que os polígonos são caracterizados por possuírem duas faces, a frontal e a traseira (*Front Face* e *Back Face*). A ordem pela qual são definidos (ou enumerados) os respectivos vértices permite identificar qual é a face frontal e qual a traseira.

A figura 1.2 representa, de forma esquemática, a estrutura interna de dados utilizada no armazenamento da descrição dos objectos que compõem a cena. Neste modelo, cada polígono é descrito internamente com base em:

- uma lista de estruturas em que cada uma contém a informação de um vértice, em coordenadas locais, relativa à sua posição espacial e à sua normal (o conceito de normal ao vértice será explicado no capítulo dedicado ao Sombreamento);
- a normal ao plano que contém o polígono, definido também no espaço de coordenadas locais. Este vector é utilizado, nomeadamente, durante a operação de remoção de faces traseiras.

1.5 Interface de visualização

O tipo de *interface* que uma API gráfica disponibiliza ao utilizador é um aspecto fundamental pois ele é que define o modo de visualização de uma determinada cena.

Um sistema de Visualização consiste num conjunto de parâmetros através dos quais um utilizador pode estabelecer uma transformação que conduza ao mapeamento de pontos definidos em coordenadas do Mundo (WCS: *World Coordinates System* - 3D), em pontos sobre uma superfície de visualização (2D).

Desse modo, o utilizador tem que definir a posição da câmara, a posição e a orientação do Plano de Visualização no sistema de coordenadas WCS e o volume de visualização dentro do qual reside a porção da cena a visualizar.

Para isso, o utilizador tem que introduzir os seguintes parâmetros (veja-se a figura 1.3)

- Posição da câmara ou *eyepoint*: **VRP** (*View Reference Point*) em WCS
- Direcção para onde a câmara aponta: **VPN** (*View Plane Normal*) ou seja, a direcção de visualização -define imediatamente o plano de visualização
- Vertical: **VUV** (*View Up Vector*) - define o ângulo de rotação em torno do VPN
- Distância do plano de visualização ao VRP, sobre a direcção VPN: **D**

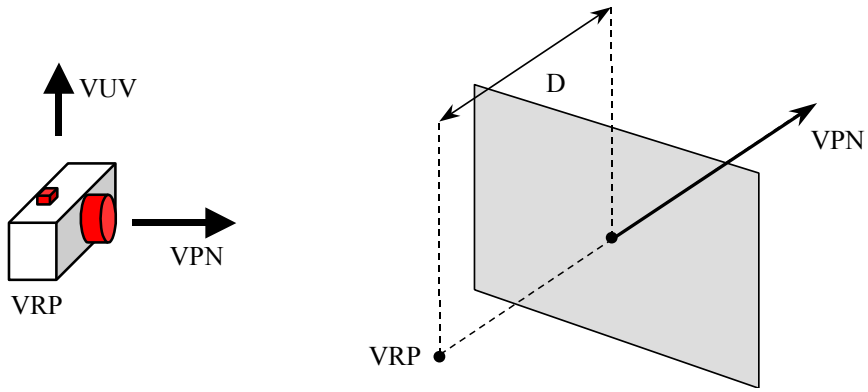


Figura 1.3 – Parâmetros de posição e orientação de um observador.

Nalguns sistemas gráficos, a fim de garantir uma maior flexibilidade, o utilizador tem a possibilidade de especificar um vector "aproximado" VUV', cuja única restrição é não ser paralelo a VPN. O sistema encarrega-se de, a partir desse vector, calcular o vector VUV através da projecção de VUV', paralelamente a VPN, no plano de visualização (ver figura 1.4).

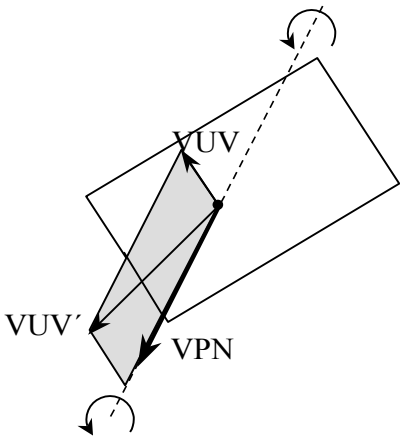


Figura 1.4 – Relação entre os vectores VUV' e VUV.

Saliente-se novamente que os parâmetros VRP, VPN e VUV' são definidos em coordenadas do Mundo. Com base nesta informação, o sistema gráfico define então o **sistema de coordenadas da câmara** cuja origem é VRP. Este referencial é ainda conhecido por diversas designações: VRC (*Viewing Reference Coordinates*), *Eye Coordinates System* e ainda **referencial uvn**. Esta última designação justifica-se pelo facto de o sistema definir esse referencial à custa de uma base ortonormada formada pelos versores u, v e n e orientada segundo a regra da mão esquerda (veja-se a figura 1.5). O versor n é calculado a partir de VPN e o versor v é extraído a partir de VUV.

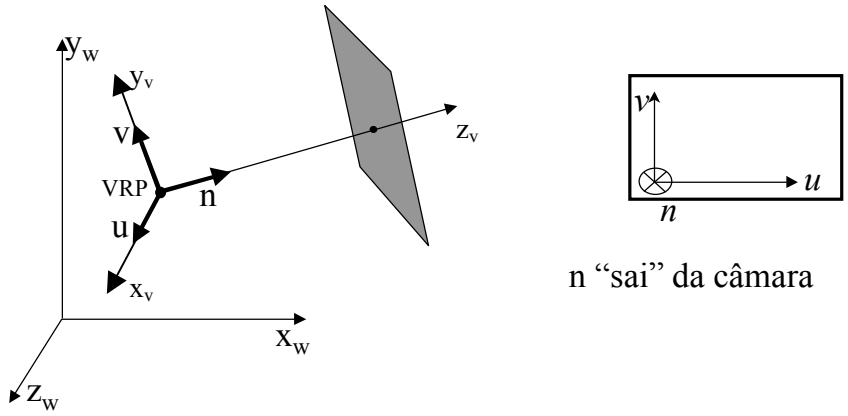


Figura 1.5 – Referencial da câmara ou uvn.

A sequência de passos a serem executados para o cálculo dos versores u, v e n são os seguintes

1) $$n = \frac{VPN}{\|VPN\|}$$

2) $$VUV = VUV' - VPN\left(VPN \bullet VUV'\right)$$

3) $$v = \frac{VUV}{\|VUV\|} \qquad u = n \times v$$

Assim, obtém-se o referencial da câmara, em que o eixo dos ZZ (denominado de $\mathbf{z_v}$) é definido pelo versor n, o eixo dos XX (denominado de $\mathbf{x_v}$) é definido pelo versor u e o eixo dos YY (denominado de $\mathbf{y_v}$) é definido pelo versor v.

Em seguida, o utilizador tem que introduzir informação acerca das dimensões de uma área rectangular, no plano de visualização, aonde irá ser projectada a cena. Esta área rectangular tem o nome de **Janela de Visualização** e no modelo de câmara virtual simples está centrada simetricamente relativamente ao VRP (veja-se a figura 1.6). Introduzem-se deste modo as dimensões 2h (altura) e 2w (largura) da Janela de Visualização. No referencial da câmara, as coordenadas do centro da janela (CW) são (0, 0, D).

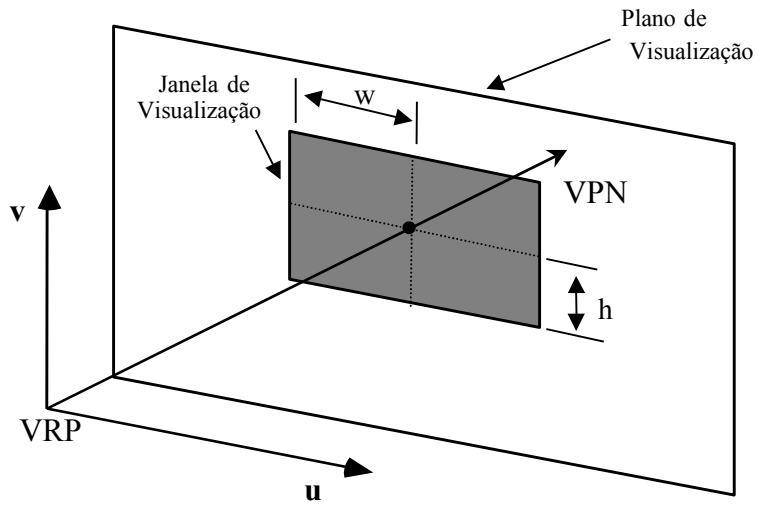


Figura 1.6 – Definição da janela de visualização.

A execução do algoritmo de recorte passa, em primeiro lugar, pela definição do volume de visualização que contém a parte da cena que está visível na direcção do observador, ou seja, tudo aquilo que a câmara "vê". A topologia do volume de visualização depende do tipo de projecção utilizada. O utilizador, no modelo de câmara virtual simples, pode especificar dois tipos de projecção: projecção perspectiva e projecção ortogonal.

No caso de o utilizador ter especificado projecção perspectiva, o volume de visualização é definido por um tronco de pirâmide infinito cujo ápice se localiza no VRP e lados sobre a janela de visualização (figura 1.7). Uma das principais características do modelo de câmara virtual simples reside no facto de, no caso deste tipo de projecção, o VRP coincidir com o centro de projecção originando, deste modo que o vértice de pirâmide seja o VRP. O eixo central da pirâmide coincide com o eixo z_v. Fazendo o recorte da cena sobre este volume, não ficam projectados objectos que se encontrem atrás do centro de projecção.

No caso de o utilizador ter especificado projecção ortognal, o volume de visualização é definido por um paralelipípedo infinito passando pelos lados da janela de visualização e de arestas paralelas à direcção VPN (figura 1.8). O eixo central do volume, à semelhança do que acontecia com o caso anterior, coincide com o eixo z_v. Este facto é consequência da adopção do modelo de câmara virtual simples, em que, no caso da projecção ortogonal, os raios projectores são paralelos à direcção VPN.

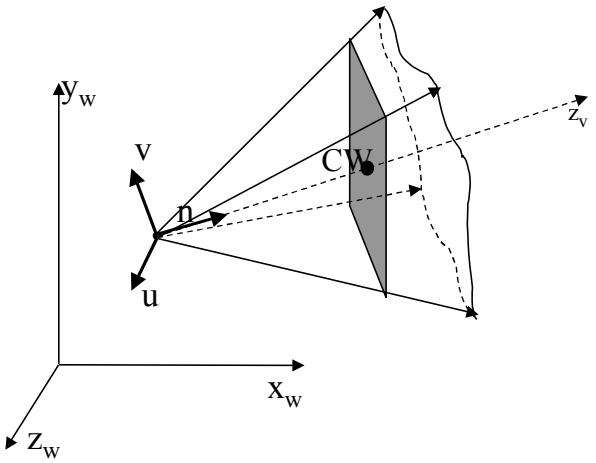


Figura 1.7 Volume de visualização perspectiva.

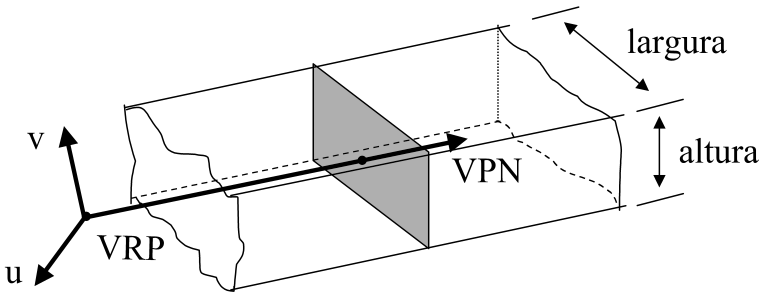


Figura 1.8 – Volume de visualização ortogonal.

Na figura 1.9 estão representadas duas vistas, a de topo e a lateral, do volume de visualização perspectivo que explicam o significado geométrico e analítico do conceito de ***Field of View*** (FOV), ou de **abertura perspectiva**, que muitas APIs gráficas geralmente disponibilizam. Deste modo, os utilizadores, em vez de especificarem a altura e a largura da janela de visualização, introduzem as aberturas horizontal e vertical com que pretendem dotar o volume de visualização perspectivo.

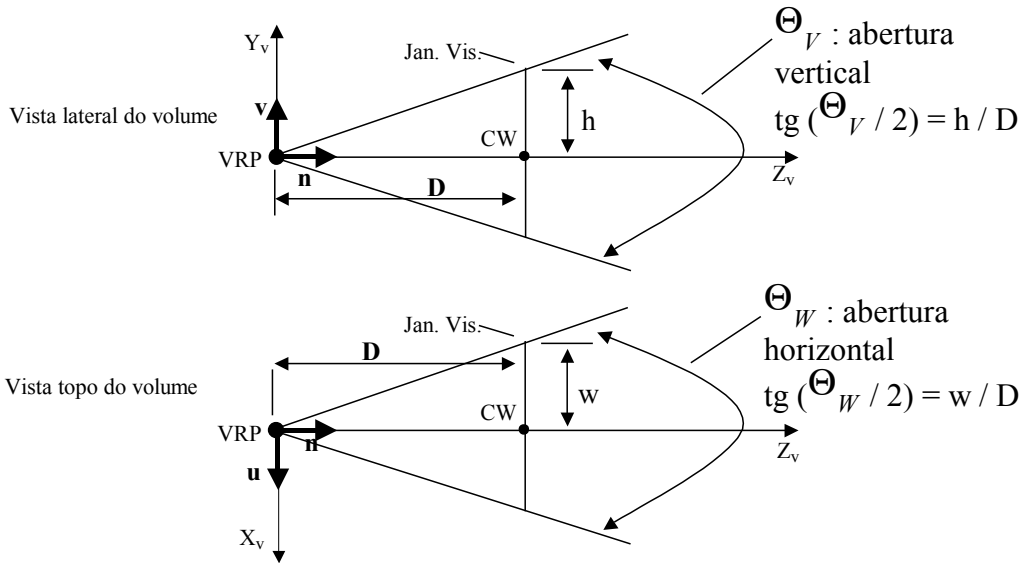


Figura 1.9 – Aberturas (FOVs) no volume de visualização perspectivo.

Se, por exemplo, o utilizador pretender implementar um efeito telescópico terá de reduzir as aberturas, de modo a concentrar a sua atenção no objecto em questão (ver figura 1.10).

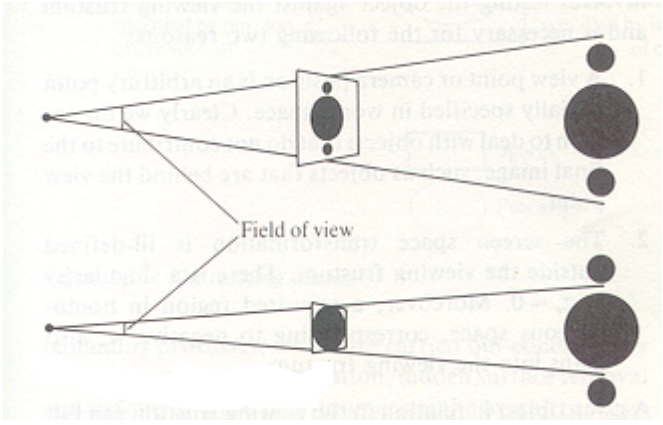
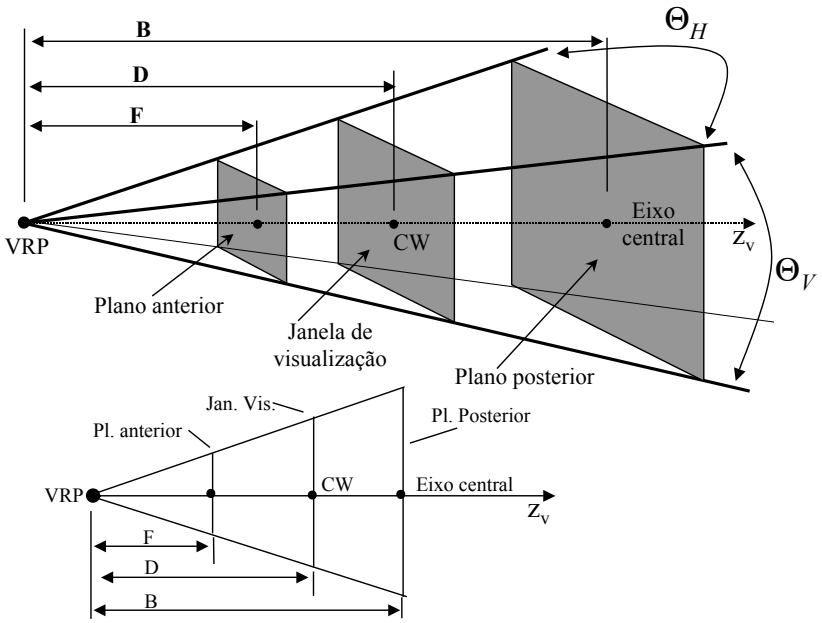


Figura 1.10 – Efeito telescópico através da manipulação do FOV

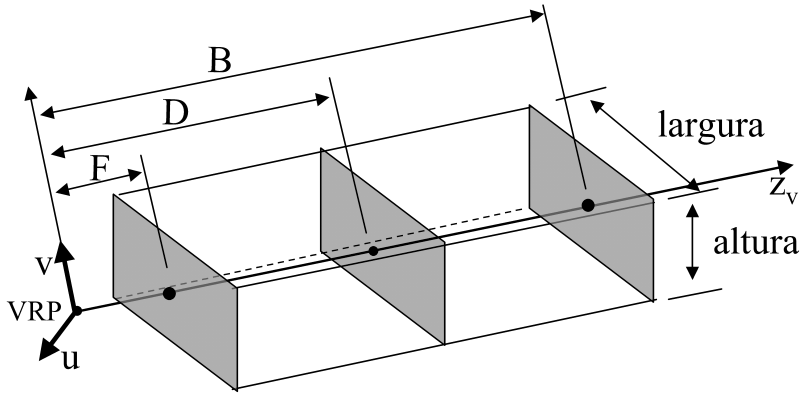
Existem consequências negativas na utilização de volumes de visualização infinitos, que são particularmente evidentes no caso das projecções em perspectiva. Os objectos muito afastados, depois de transformados ponto a ponto, podem reduzir-se a uma pequena mancha no ecrã, o que corresponde a um desperdício de tempo de computação. Por outro lado, projecções de objectos demasiado próximos, podem gerar resultados caóticos, impossibilitando deste modo uma correcta visualização (imagine-se captar uma fotografia a uma paisagem no momento em que uma pessoa passa à frente da câmara fotográfica).

A solução passa por definir planos de recorte paralelos ao plano de projecção, através da sua distância ao ponto VRP e medida ao longo da direcção VPN. Especifica-se, assim, um plano de recorte anterior através de uma distância F e um plano de recorte

posterior através de uma distância B. No referencial uvn, o qual se encontra orientado segundo a regra da mão esquerda, a distância F tem de ser maior que zero e menor que a distância B. Da incorporação destes planos de recorte, resultam os volumes de visualização finitos ilustrados na figura 1.11. É usual designar o tronco de pirâmide finito, que caracteriza o volume de visualização perspectivo, por ***frustum***.



Volume Perspectivo



Volume Ortogonal

Figura 1.11 – Volumes de visualização finitos após a incorporação dos planos de recorte anterior e posterior.

O volume ortogonal é definido pelos seguintes planos:

$$\begin{aligned} -w \le x_v \le w \\ -h \le y_v \le h \\ F \le z_v \le B \end{aligned} \qquad (1\text{-}1)$$

No caso do *frustum*, o volume é definido pelas seguintes desigualdades

$$\begin{aligned} -(w/D)\times z_v \le x_v \le (w/D)\times z_v \\ -(h/D)\times z_v \le y_v \le (h/D)\times z_v \\ F \le z_v \le B \end{aligned} \qquad (1\text{-}2)$$

Na figura 1.12 ilustra-se um resumo dos elementos necessários para a *interface* de um sistema gráfico baseado no modelo da câmara virtual.

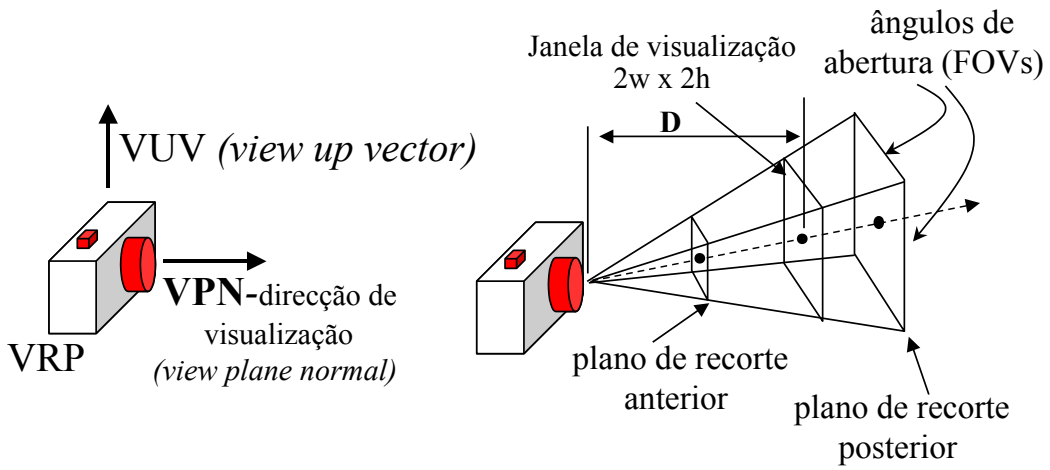


Figura 1.12 – Resumo dos elementos necessários para a interface de um sistema gráfico.

1.6 Construção de uma vista arbitrária 3D

A construção de uma vista arbitrária 3D, por parte de um sistema gráfico, passa, em primeiro lugar, por colocar os objectos, descritos no Sistema de Coordenadas do Mundo, no referencial da Câmara (ou uvn) estabelecido de acordo com a posição e a orientação da câmara. Após essa operação, é executado o algoritmo de recorte. Ir-se-á nesta secção determinar todas as transformações que deverão ser realizadas antes do algoritmo de recorte. Estas são efectuadas em coordenadas homogéneas, originando por conseguinte matrizes de dimensão 4 x 4.

Após a especificação do VRP (simultaneamente *eyepoint* e centro de projecção) e dos vectores que definem a orientação do plano de projecção, VPN eVUV, há que referenciar a cena no referencial uvn. Esta operação, designada por **Transformação de Visualização**, resume-se a colocar os objectos da cena de acordo com a posição e a direcção de um observador e é implementada por uma translação e uma rotação cujas matrizes são:

$$M_{rot} = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad M_{trans} = \begin{bmatrix} 1 & 0 & 0 & -VRP_x \\ 0 & 1 & 0 & -VRP_y \\ 0 & 0 & 1 & -VRP_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1\text{-}3)$$

Assim a transformação de visualização é implementada por

$$M_{vis} = M_{rot} \times M_{trans} \qquad (1\text{-}4)$$

O próximo passo tem como objectivo simplificar a operação de recorte e passa pela conversão de volumes de visualização genéricos para volumes de visualização normalizados, também apelidados de volumes canónicos. Essa conversão é realizada por duas Transformações de Normalização consoante o tipo de projecção pretendido seja ortogonal ou de perspectiva. Assim, ter-se-á N_{ort} para projeccção ortogonal e N_{persp} para a projecção perspectiva.

A topologia dos volumes canónicos está representada na figura 1.13.

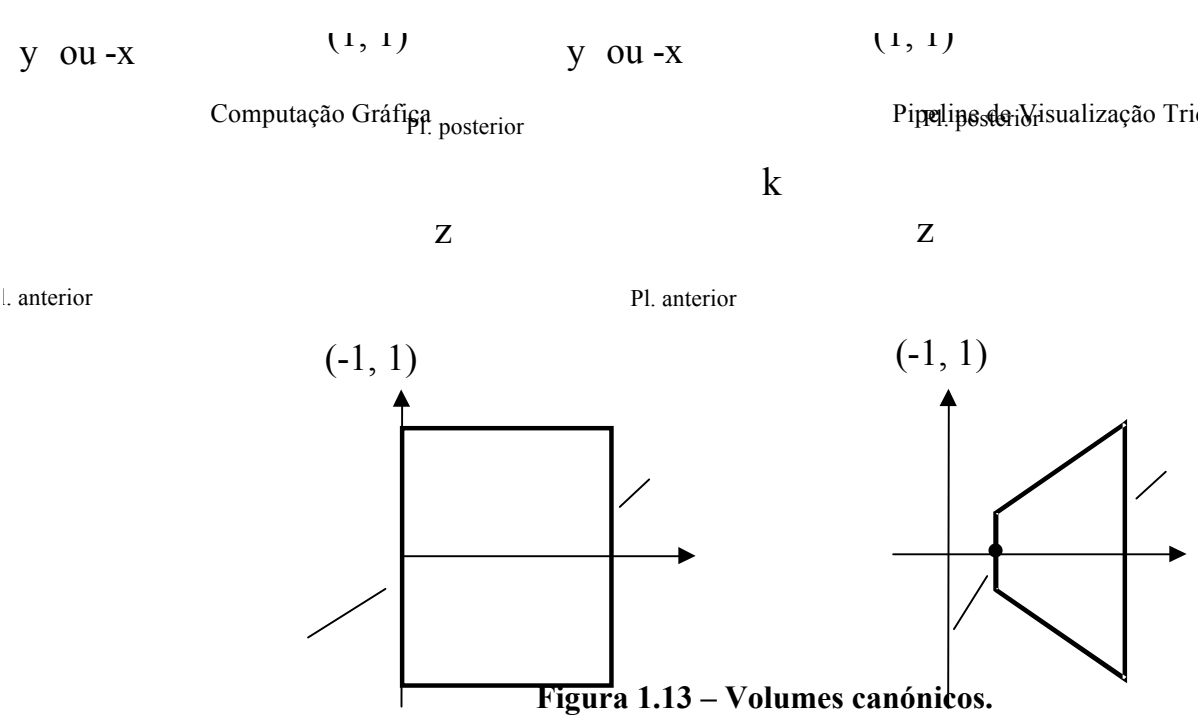


Figura 1.13 – Volumes canónicos.

De acordo com a figura 1.13, o volume canónico ortogonal é definido por seis planos para os quais

$$x=-1 \quad x=1 \quad y=-1 \quad y=1 \quad z=0 \quad z=1 \tag{1-5}$$

e o volume canónico perspectivo é caracterizado pelos planos

$$x=-z \quad x=z \quad y=-z \quad y=z \quad z=\mathrm{k}\,(0\le \mathrm{k}\le 1) \quad z=1 \tag{1-6}$$

A conversão para o volume canónico ortogonal implica a realização de dois passos:

- Translação do paralelepípedo de F unidades na direcção segundo Z, de modo a que o plano anterior se localize na origem;
- Escalar para que $-1\le x,y\le 1$ e $0\le z\le 1$.A estas transformações correspondem as matrizes de transformação

$$T_{ort}=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&-F\\0&0&0&1\end{bmatrix} \qquad S_{ort}=\begin{bmatrix}1/w&0&0&0\\0&1/h&0&0\\0&0&1/B-F&0\\0&0&0&1\end{bmatrix} \tag{1-7}$$

A transformação completa para o cálculo do volume canónico ortogonal será então

$$N_{ort}=S_{ort}\times T_{ort} \tag{1-8}$$

Para obter o volume canónico perspectivo há que escalar o volume de visualização genérico nas duas direcções x e y de modo a que se obtenha um declive unitário para os planos laterais. Ora as equações destes planos do *frustum* genérico são (veja-se a figura 1.9)

$$-\frac{w}{D}z_v \le x_v \le \frac{w}{D}z_v \quad \text{e} \quad -\frac{h}{D}z_v \le y_v \le \frac{h}{D}z_v \tag{1-9}$$

o que conduz a utilizar a seguinte matriz de escala

$$S_{XY} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1-10}$$

em que

$$\begin{array}{l} S_x = \cot g \dfrac{\theta_w}{2} \\ Sy = \cot g \dfrac{\theta_h}{2} \end{array} \quad \text{ou} \quad \begin{array}{l} S_x = \dfrac{D}{w} \\ Sy = \dfrac{D}{h} \end{array} \tag{1-11}$$

Por outro lado, z não pode ser maior que 1. Ora isto significa escalar o plano z = B. Para isso dever-se-á realizar um escalamento uniforme nas três direcções espaciais de modo a manter o declive unitário dos planos laterais.

$$S_Z = \begin{bmatrix} 1/B & 0 & 0 & 0 \\ 0 & 1/B & 0 & 0 \\ 0 & 0 & 1/B & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1-12}$$

Em conclusão

$$N_{persp} = S_z \times S_{xy} \tag{1-13}$$

Note-se que *k* no volume canónico perspectivo vale

$$k = F/B \tag{1-14}$$

Estamos agora em condições, de indicar, para cada tipo de projecção, qual a transformação total a ser aplicado aos objectos da cena antes da execução do algoritmo de recorte.

Para o caso da projecção ortogonal, o sistema aplica uma transformação, cuja descrição matricial é

$$M_{ort} = S_{ort} \bullet T_{ort} \bullet M_{rot} \bullet M_{trans} \tag{1-15}$$

No caso da projecção perspectiva, a transformação é

$$M_{persp} = S_Z \bullet S_{XY} \bullet M_{rot} \bullet M_{trans} \tag{1-16}$$

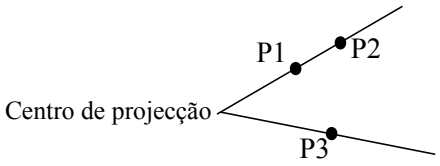
Após aplicar destas transformações, o sistema está em condições de realizar a operação de recorte contra os respectivos volumes canónicos.

1.7 7. Transformação Perspectiva

É óbvio que se afirme que a projecção 2D no ecrã só poderá ocorrer após a realização de todas as operações que envolvam tridimensionalidade. Em particular, a questão de visibilidade revela-se pertinente quanto a este aspecto, pois a sua implementação baseia-se em testes de profundidade (utilização da terceira coordenada). Na realidade, a solução para o problema da visibilidade passa, em termos básicos, por verificar se um determinado ponto obstrui a visibilidade de outro. Para que tal ocorra, é necessário que ambos os pontos se localizem sobre o mesmo raio projector. Em caso afirmativo, basta depois comparar as profundidades de cada um deles para se decidir qual deles está mais próximo do observador[3].

Na projecção ortogonal, os cálculos destinados a determinar se dois pontos P1 e P2 se encontram sobre o mesmo raio projector são bastante simples pois consistem em verificar se x1 = x2 **e** y1 = y2. Esta simplicidade decorre do facto de os raios projectores na projecção ortogonal do modelo de câmara virtual simples serem paralelos ao eixo dos ZZ.

No caso da projecção perspectiva, os cálculos complicam-se pois os raios projectores são convergentes num centro de projecção e, portanto, oblíquos (ver figura 1.14). Isto implica imediatamente que o cálculo de visibilidade se revele dispendioso sobre o frustum canónico.



- P3 não obstrui nenhum outro ponto
- P1 oculta P2:
  - x1/z1 = x2/z2 e y1/z1 = y2/z2 (4 divisões)
- no vol.ortogonal bastava verificar se
  - x1 = x2 e y1 = y2

Figura 1.14 – Cálculos na projecção perspectiva, para determinar se dois pontos se localizam no mesmo raio projector.

A solução para este problema passa por converter o *frustum* normalizado no paralelepípedo canónico através de uma operação conhecida por Transformação

[3] Uma vez que o referencial da câmara se encontra orientado segundo a regra da mão esquerda, valores menores de profundidade correspondem a objectos mais próximos do observador.

Perspectiva que opera num espaço 3D e de que resultam pontos ainda num espaço 3D. As vantagens inerentes à utilização desta transformação são:

- A determinação da possibilidade de oclusão entre dois pontos é realizada por simples comparações das duas primeiras coordenadas (x1 = x2 e y1 = y2)
- O recorte é especializado (eventualmente por hardware) apenas para o volume canónico ortogonal
- No desenho 2D utiliza-se a projecção ortogonal

Para uma análise dos fundamentos matemáticos que permitem calcular a Transformação Perspectiva aconselha-se a consulta de [Rogers97, Blinn83]. A sua descrição matricial é:

$$M_P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{1-k} & \frac{-k}{1-k} \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad (1\text{-}17)$$

O uso desta transformação tem as seguintes consequências (ver figura 1.15):

- Transforma vértices do frustrum em vértices do paralelepípedo canónico
- O vértice da pirâmide (centro de projecção) desloca-se para -∞
- Plano anterior, localizado em k, é deslocado para a origem
- Os valores de z variam entre 0 e 1

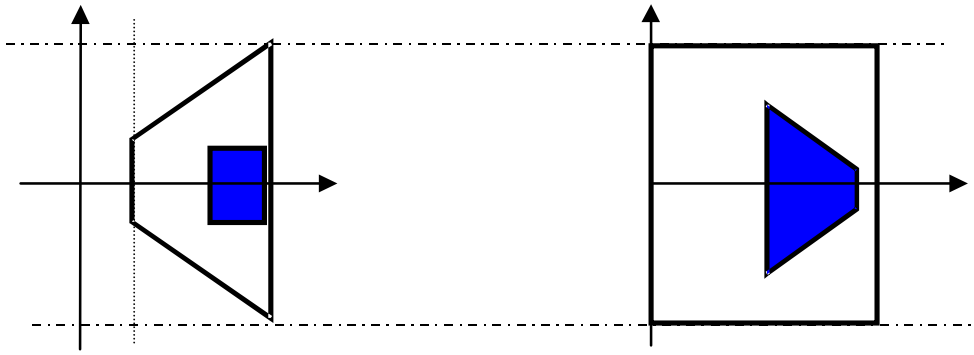


Figura 1.15 – Transformação perspectiva

Analisando a figura 1.15, pode-se considerar que a Transformação Perspectiva representa uma extensão 3D da projecção perspectiva.

No caso da projecção perspectiva, pode-se agora definir uma única matriz 4x4 a aplicar a todos os pontos da cena tal que

$$M'_{persp} = M_P \bullet S_Z \bullet S_{XY} \bullet M_{rot} \bullet M_{trans} \quad (1\text{-}18)$$

De seguida, é executado o algoritmo de recorte que tem a particularidade de, no caso da projecção perspectiva, ser efectuado em coordenadas homogéneas. Obtém-se, depois, então as coordenadas cartesianas dos pontos resultantes, através da divisão pela quarta coordenada homogénea, divisão essa designada por **divisão perspectiva**, neste caso particular. Esses pontos são depois enviados para o andar de rasterização.