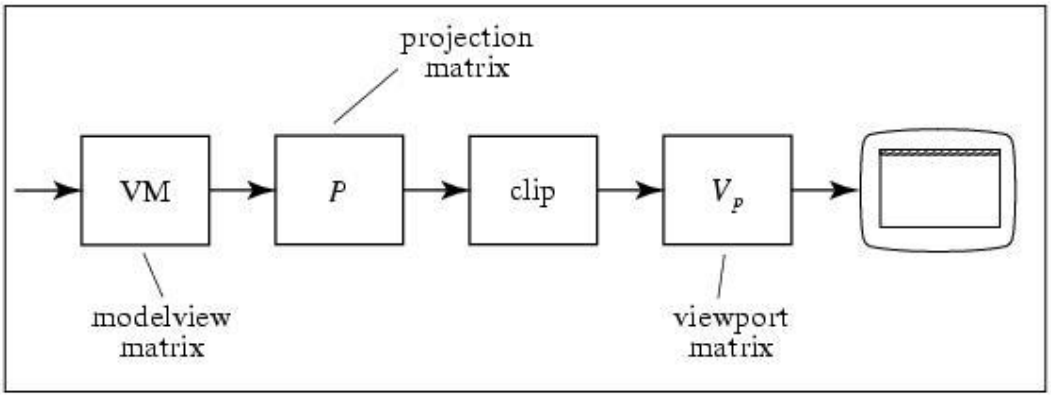
INF 390 – Computação Gráfica
Prof. Marcus V. A. Andrade

Visualização 3D em OpenGL

Gerando cenas 3D usando OpenGL

- o processo de geração de imagens tridimensionais em OpenGL pode ser descrito através do seguinte *pipeline*:

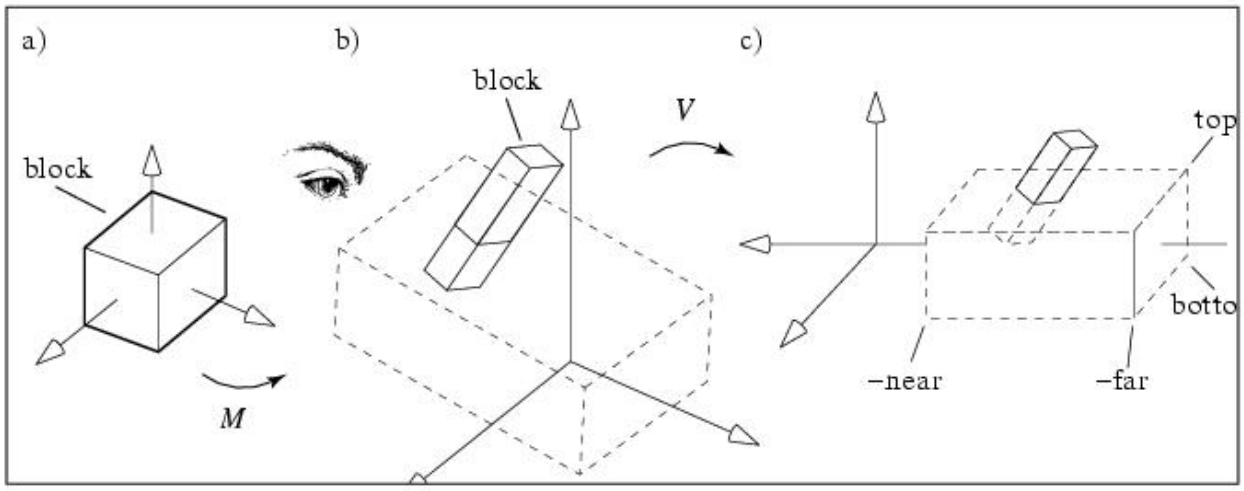


- Basicamente, o processo consiste na aplicação de 3 matrizes:
  - ***Modelview***
  - ***Projection***
  - ***Viewport***

- A matriz *Modelview* combina duas transformações: a seqüência de operações de modelagem dos objetos e as operações para orientar e posicionar a câmera no espaço.

- Esta matriz corresponde ao produto *VM* de duas matrizes, onde *M* é a matriz de modelagem e *V* é a matriz de visualização.

- A figura a seguir ilustra o resultado da aplicação de cada das matrizes *M* e *V* aos objetos da imagem



- A matriz de projeção é a responsável pela translação e mudança de escala nos objetos correspondentes à aplicação da projeção (ortográfica ou perspectiva) aos objetos

- A matriz *viewport* mapeia os objetos (que restaram após a realização da operação de recorte - *clip*) do sistema de coordenadas da janela da imagem para janela de visualização

- O posicionamento da câmera (definição da matriz *V*) em OpenGL é realizado através da função *gluLookAt* que recebe como parâmetros as coordenadas do observador, do "ponto de interesse" (foco) e de um ponto para definir o vetor *view up*.

- Mais precisamente,

```
glMatrizMode(GL_MODELVIEW);  // estabelece a matriz corrente

glLoadIdenttity();
gluLookAt(eye.x,   eye.y,  eye.z,       // coordenadas do observador
          look.x, look.y, look.z,       // "ponto de interesse"
          upp.x,  upp.y,  upp.z);       //  direção do vetor view up
```

- Isto corresponde a posicionar a câmera no ponto (eye.x, eye.y, eye.z) no seguinte sistema de coordenadas:

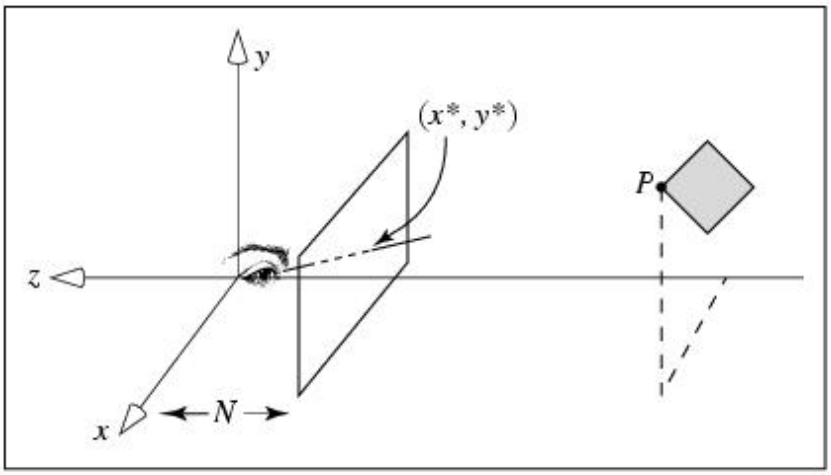
  n = *eye – look*
  u = ***up*** **x** ***n*** // onde ***up*** é o vetor na direção do ponto upp
  v = ***n*** **x** ***u***

- Supondo que os vetores acima estão normalizados, então o comando acima define a seguinte matriz:

$$V = \begin{pmatrix} u_x & u_y & u_z & d_x \\ v_x & v_y & v_z & d_y \\ n_x & n_y & n_z & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

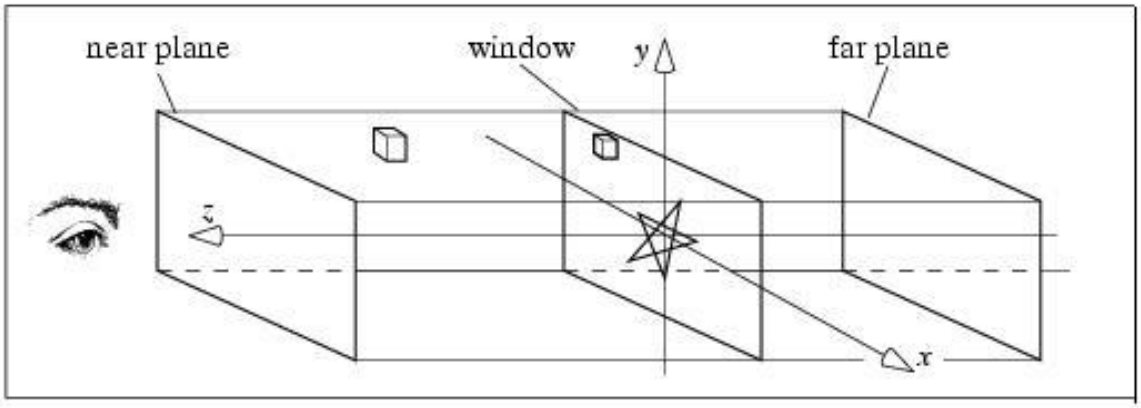
onde (d_x, d_y, d_z) = (- eye . ***u,*** - eye . ***v,*** - eye . ***n)***

- Por definição, se a posição da câmera não é definida pelo usuário (se função *gluLookAt* não é chamada), então a câmera é posicionada na origem do sistema orientada ("olhando") na direção negativa de *z*.



- A definição da matriz de projeção *P* depende do tipo de projeção desejada.

- No caso da projeção paralela, esta matriz é definida pelo comando *glOrtho* que recebe como parâmetros os 6 valores para definir o volume de visualização:

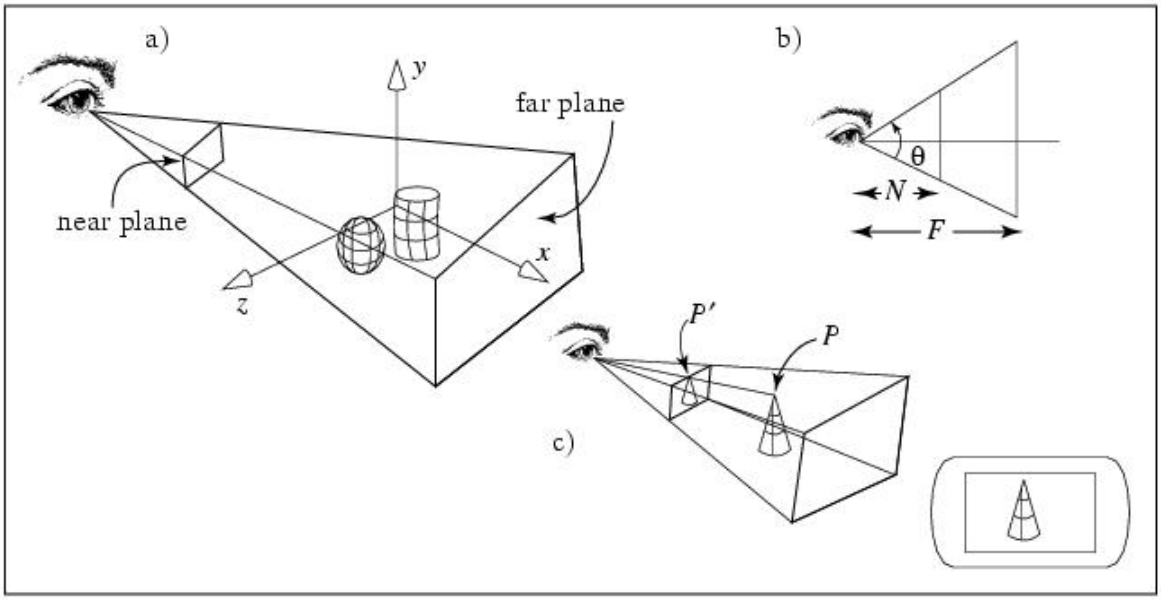


- Mais precisamente, a janela de visualização para uma projeção paralela é definida da seguinte forma:

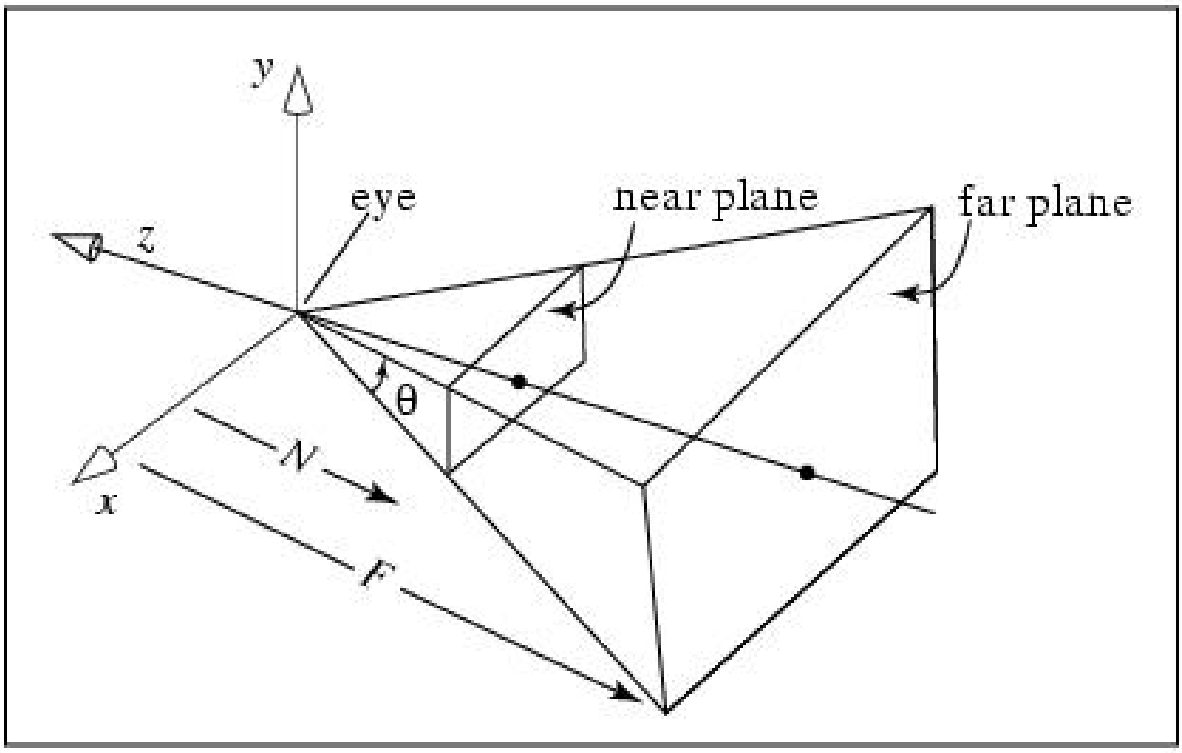
```
glMatrizMode(GL_PROJECTION);  // estabelece a matriz corrente

glLoadIdenttity();
glOrtho(left, right, bottom, top, near, far);
```

- Na projeção em perspectiva, o volume de visualização corresponde a:



- Este volume de visualização pode ser definido pelos seguintes parâmetros:



- Em OpenGL, este volume de visualização é definido pelos seguintes comandos:

```
glMatrizMode(GL_PROJECTION);  // estabelece a matriz corrente

glLoadIdentity();
gluPerspective(viewAngle, aspectRatio, N, F);
```

onde *viewAngle* é o angulo θ (na figura anterior) fornecido em graus, o *aspectRatio* corresponde ao valor da divisão da largura da janela pela sua altura e *N* e *F* são as distâncias da origem dos planos *Near* e *Far*

- Além disso, também existe o comando *glFrustum* que permite a definição do volume de visualização através dos planos:

```
glFrustum(left, right, bottom, top, N, F);
```

onde *left, right, bottom* e *top* definem os limites esquerdo, direito, inferior e superior da janela de visualização enquano *N* e *F* definem os planos *near* e *far* que completam o volume de visualização