



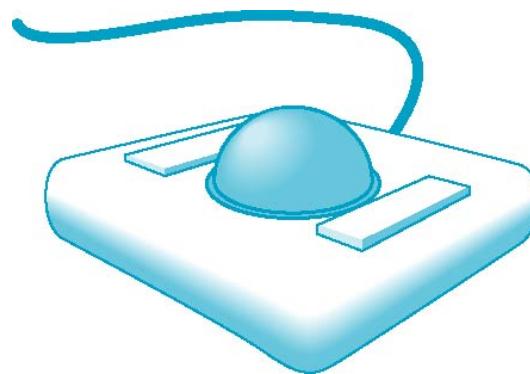
MAC420/5744: Introdução à Computação Gráfica

Marcel P. Jackowski
mjack@ime.usp.br

Aula #10: Introdução à projeções

Trackball

- Dispositivo similar ao mouse, porém com a esfera na parte superior



- Se existir pouca fricção entre a esfera e os roladores, podemos imprimir uma força para ele rotacionar continuamente em um eixo
- Dois modos de operação
 - Rastreamento dos movimentos da mão
 - Rolagem contínua

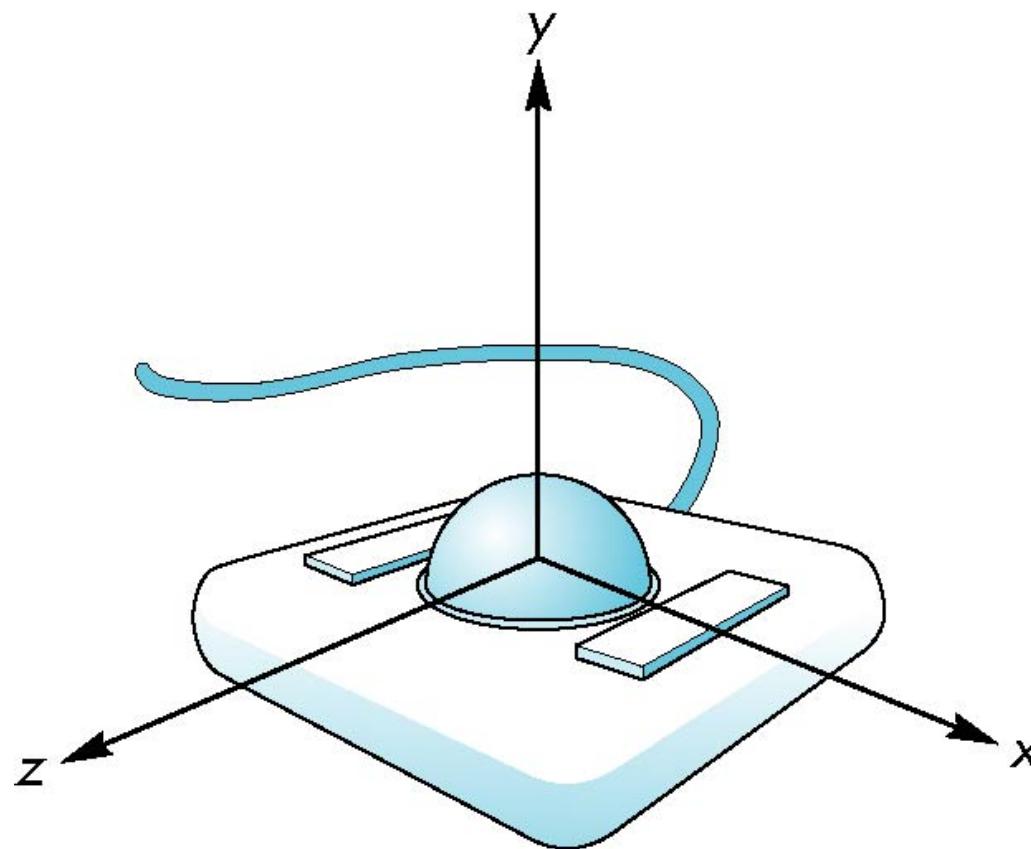
Um trackball a partir do mouse

- Problema: queremos obter o mesmo comportamento a partir de um mouse comum
- Emular um trackball ideal (sem fricção) usando o mouse
- Resolvemos em dois passos:
 - Mapeamos a posição do trackball para a posição do mouse
 - Usamos observadores de eventos para gerenciar cada modo de operação

Usando quatérnios

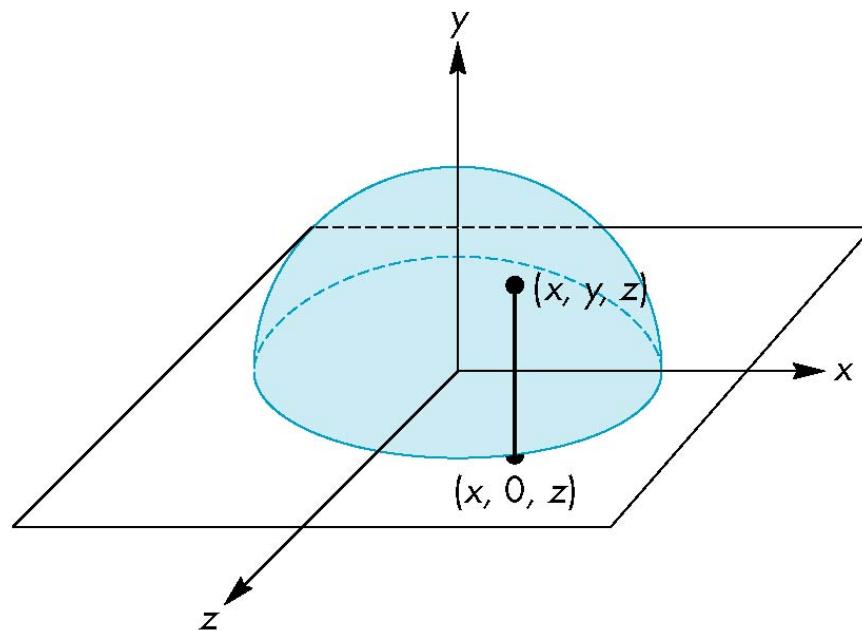
- A matemática com quatérnios funciona bem para representar rotações ao redor da origem
- Podemos utilizá-los para evitar matrizes de rotação no trackball
 - O código para essa tarefa foi feito e disponibilizado há bastante tempo (antes da era dos shaders) pela Silicon Graphics, Inc. (SGI)
- Shaders baseados em quatérnios são simples

Referencial do trackball



Projeção da posição do trackball

- Podemos relacionar a posição do trackball para coordenadas normalizadas no mouse pad utilizando uma projeção ortogonal



Revertendo a projeção

- A superfície esférica superior (domo) da esfera do trackball e o mouse pad são superfícies bidimensionais
- Um ponto (x, z) no mouse pad corresponde ao ponto (x, y, z) na superfície esférica do trackball onde

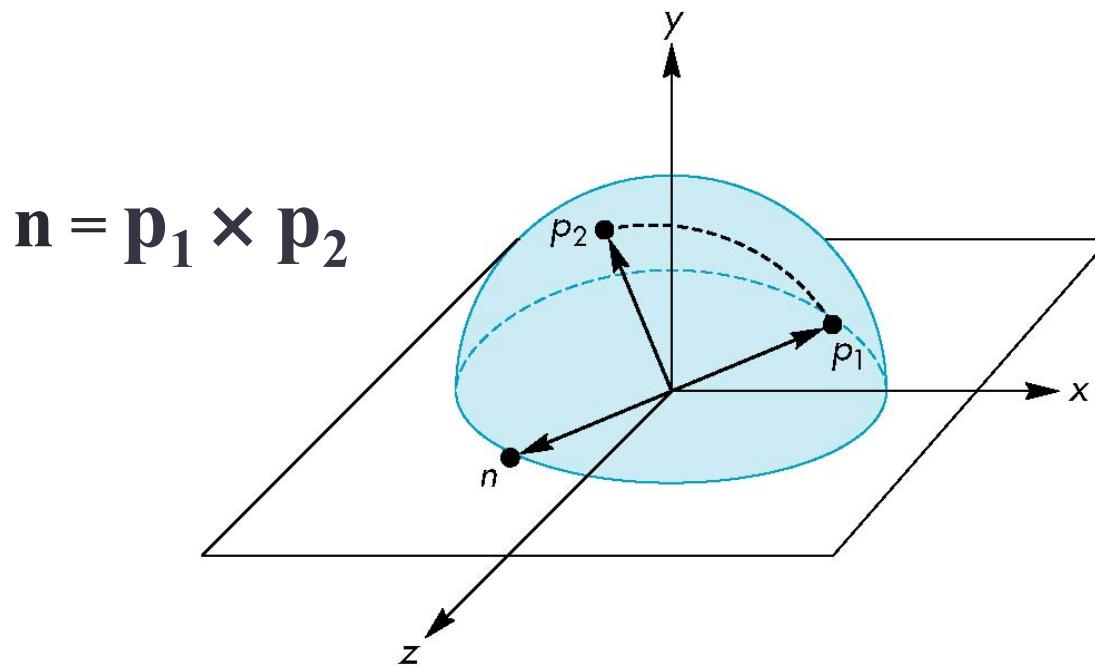
$$y = \sqrt{1^2 - x^2 - z^2}$$

Cálculo de rotações

- Vamos supor que termos dois pontos obtidos com o mouse
- Podemos projetá-los na esfera para transformá-los nos pontos \mathbf{p}_1 e \mathbf{p}_2
- Estes pontos determinam um grande círculo na esfera
- Podemos rotacionar de \mathbf{p}_1 para \mathbf{p}_2 , achando o eixo de rotação apropriado e o ângulo entre estes dois pontos.

Eixo de rotação

- O eixo de rotação é dado pela normal ao plano determinado pela origem, \mathbf{p}_1 e \mathbf{p}_2



Ângulo de rotação

- O ângulo do arco entre \mathbf{p}_1 e \mathbf{p}_2 é dado pela fórmula:

$$|\mathbf{a} \times \mathbf{b}| = |\mathbf{a}| |\mathbf{b}| \sin \theta \quad \sin \theta = |\mathbf{n}|$$

- Se movermos o mouse lentamente ou amostrá-lo frequentemente, q será pequeno e poderemos utilizar a aproximação

$$\sin \theta \approx \theta$$

Vertex shader

```
attribute vec4 vPosition;
attribute vec4 vColor;
varying vec4 fColor;
uniform vec4 rquat; // rotation quaternion

// quaternion multiplication
vec4 multq(vec4 a, vec4 b)
{
    return(
        vec4(a.x*b.x - dot(a.yzw, b.yzw),
              a.x*b.yzw + b.x*a.yzw +
              cross(b.yzw, a.yzw))
    );
}
```

Vertex shader

```
// inverse quaternion
vec4 invq(vec4 a)
{ return(vec4(a.x, -a.yzw) / dot(a,a)); }

void main() {
    vec3 axis = rquat.yzw;
    vec4 r, p;

    // input point quaternion
    p = vec4(0.0, vPosition.xyz);
    // rotate point quaternion
    p = multq(rquat, multq(p, invq(rquat)));

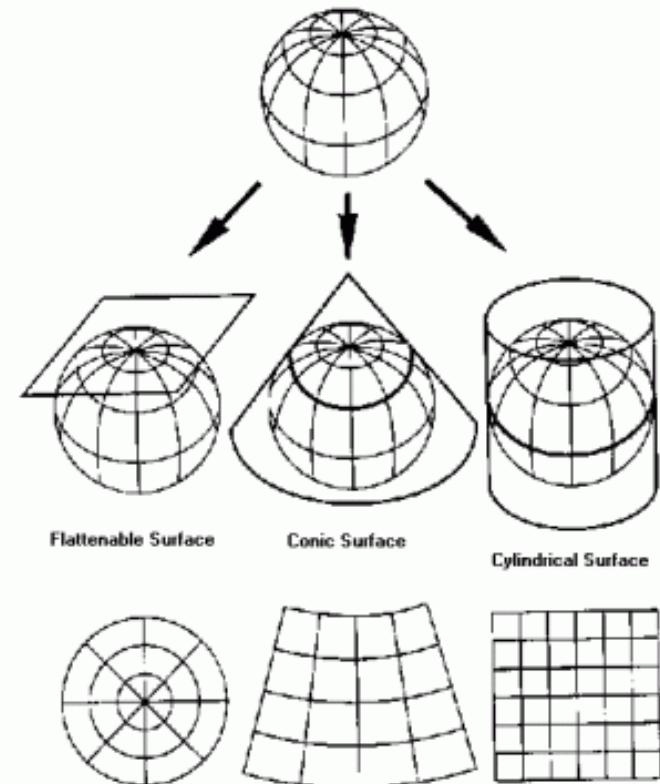
    gl_Position = vec4(p.yzw, 1.0);
    fColor = vColor;
}
```

Introdução

- A visualização de uma cena requer a presença de três elementos básicos
 - Objetos
 - Câmera (observador)
 - Plano de projeção
 - Raios projetores (linhas de visada)
- Visualizações clássicas são baseadas em relacionamentos entre estes elementos
- Assume-se que os objetos são construídos com faces planares.

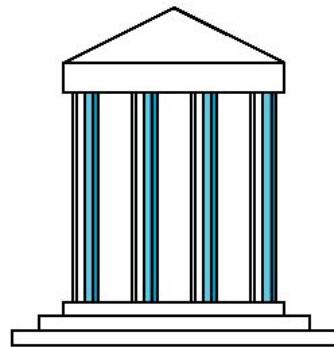
Projeções planares

- Projetam em um plano
- Raios projetores são retas que:
 - Convergem em um centro de projeção
 - Ou são paralelos
- Tais projeções preservam retas
 - Não necessariamente ângulos
- Projeções não planares são utilizadas em aplicações como a construção de mapas.

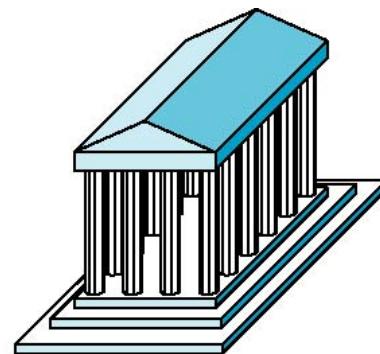


Tipos de projeções

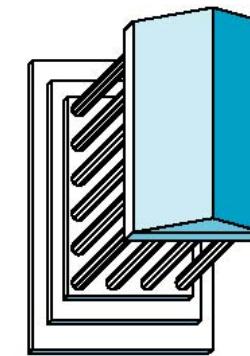
Projeções planares clássicas



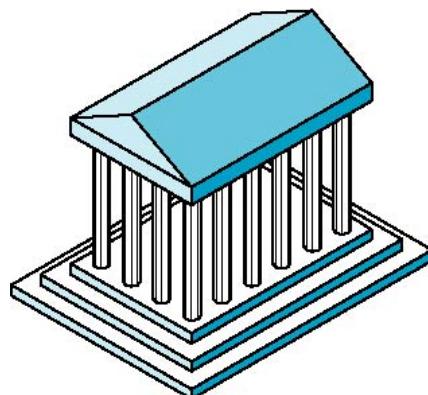
Front elevation



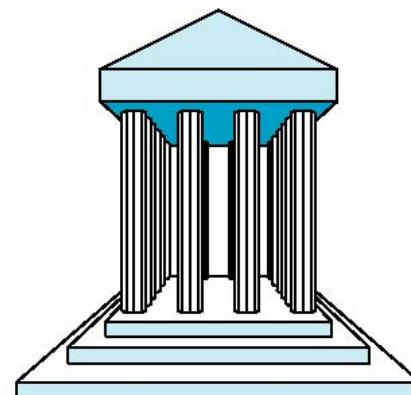
Elevation oblique



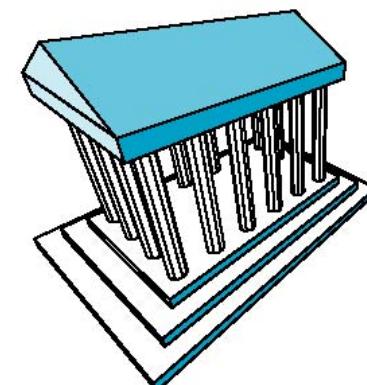
Plan oblique



Isometric



One-point perspective



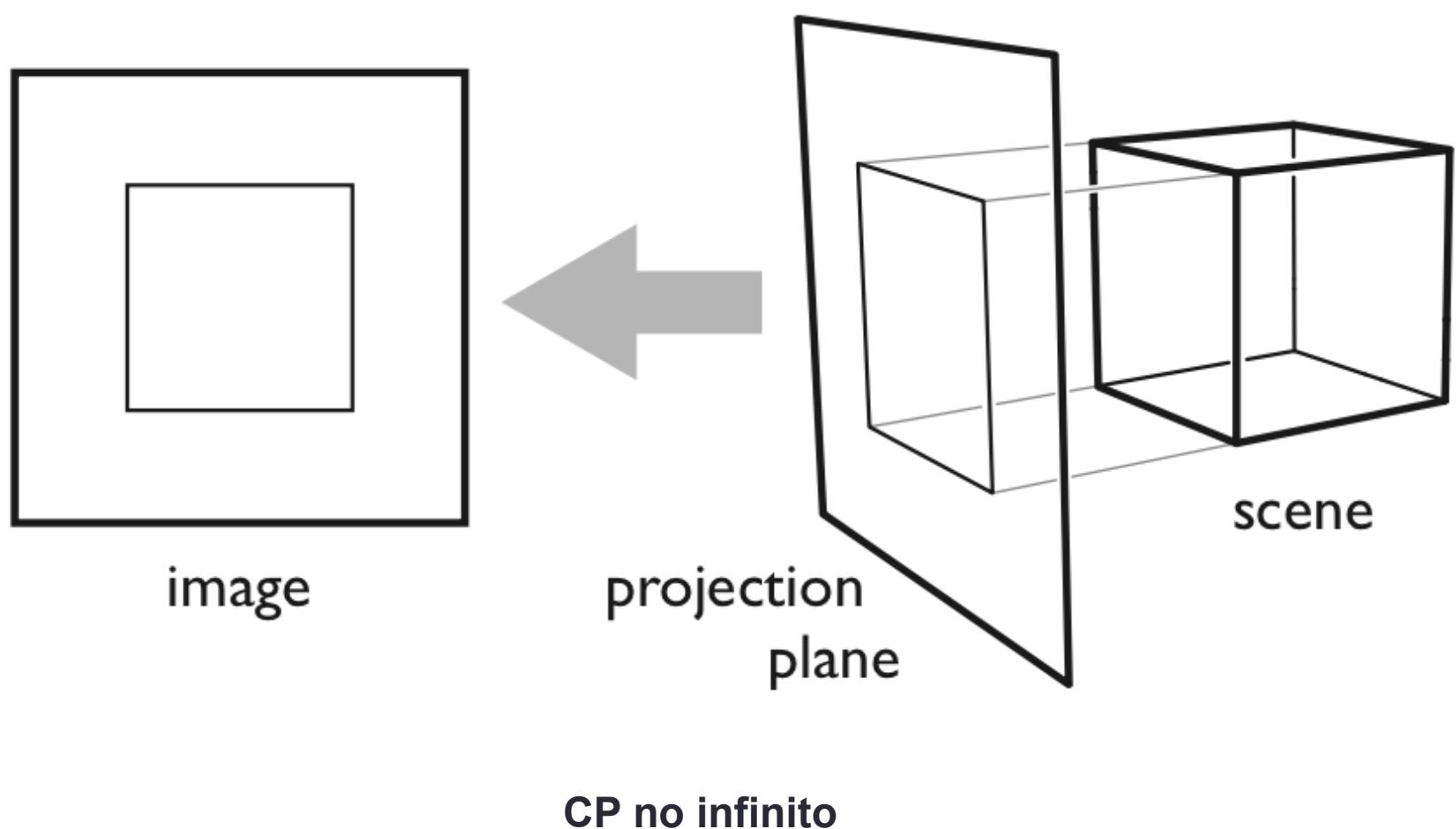
Three-point perspective

noção de face principal

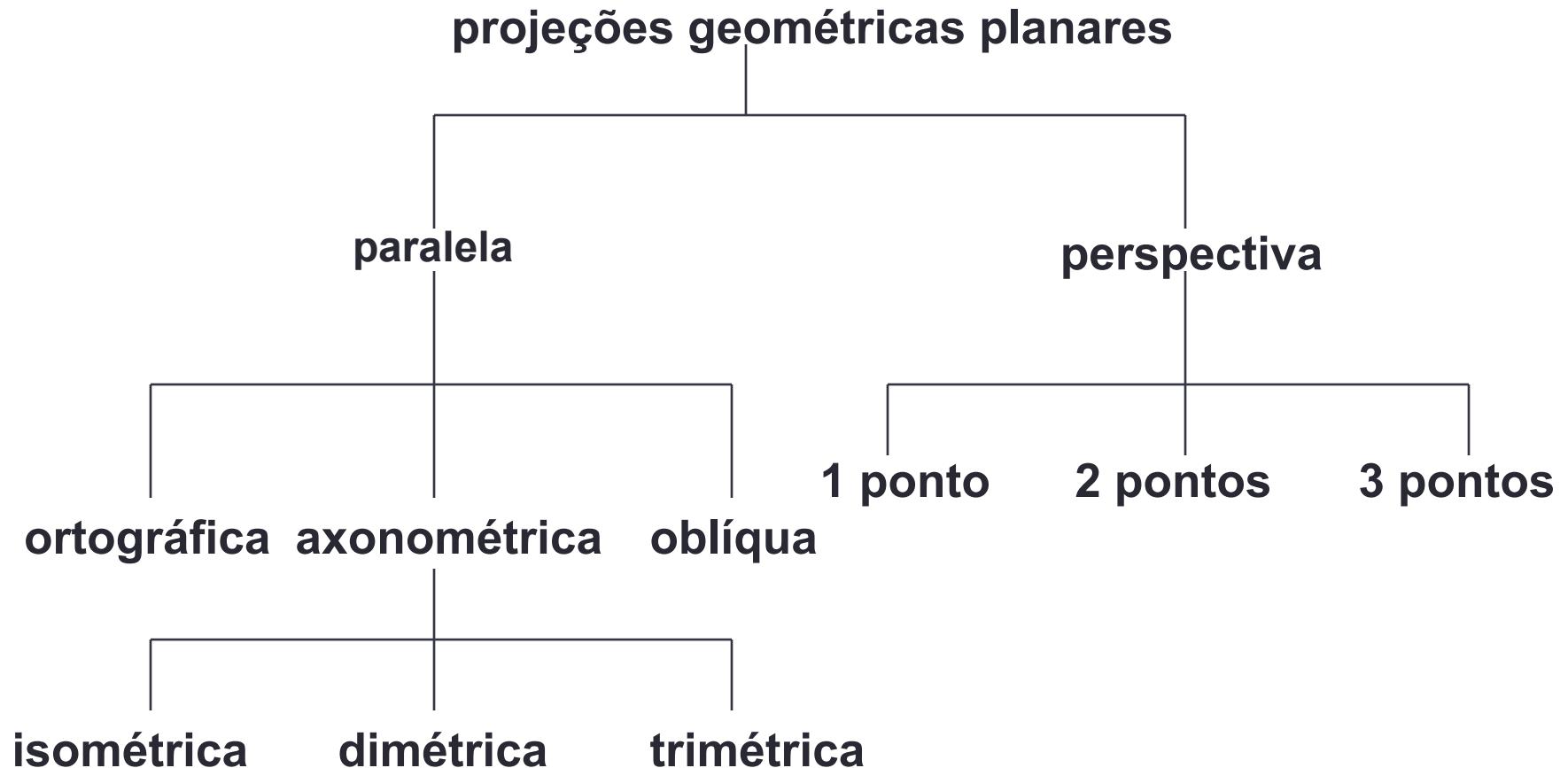
Clássica vs CG

- Visualização clássica
 - Técnicas específicas para cada tipo de projeção
 - Diversidade de projeções
 - Dependência no relacionamento entre objeto, observador e plano de projeção
- Computação gráfica
 - Projeção paralela e perspectiva
 - Um único pipeline de visualização
 - Independência de especificações

Projeção paralela

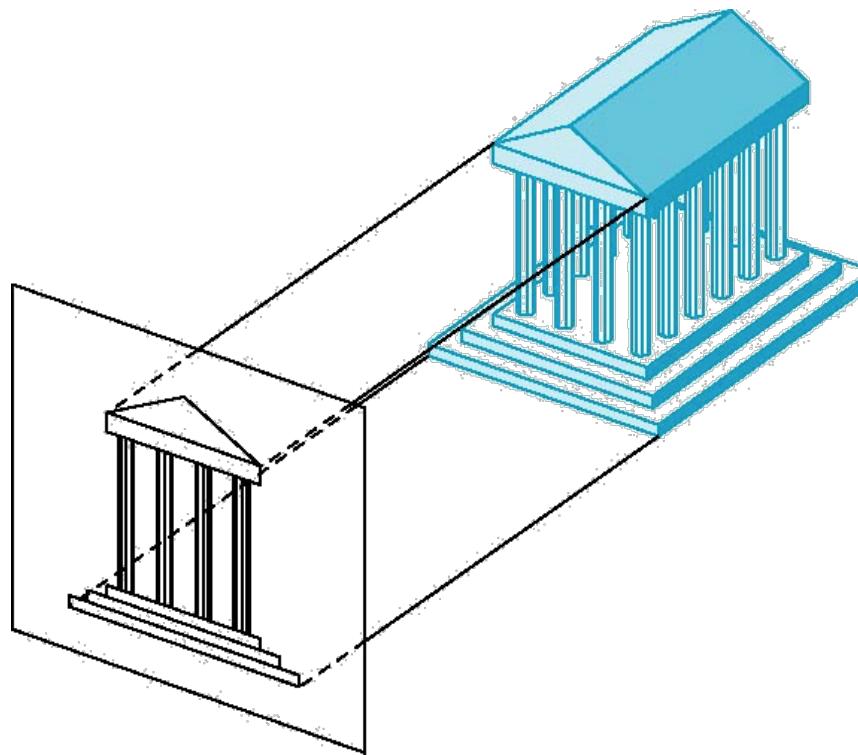


Taxonomia das projeções



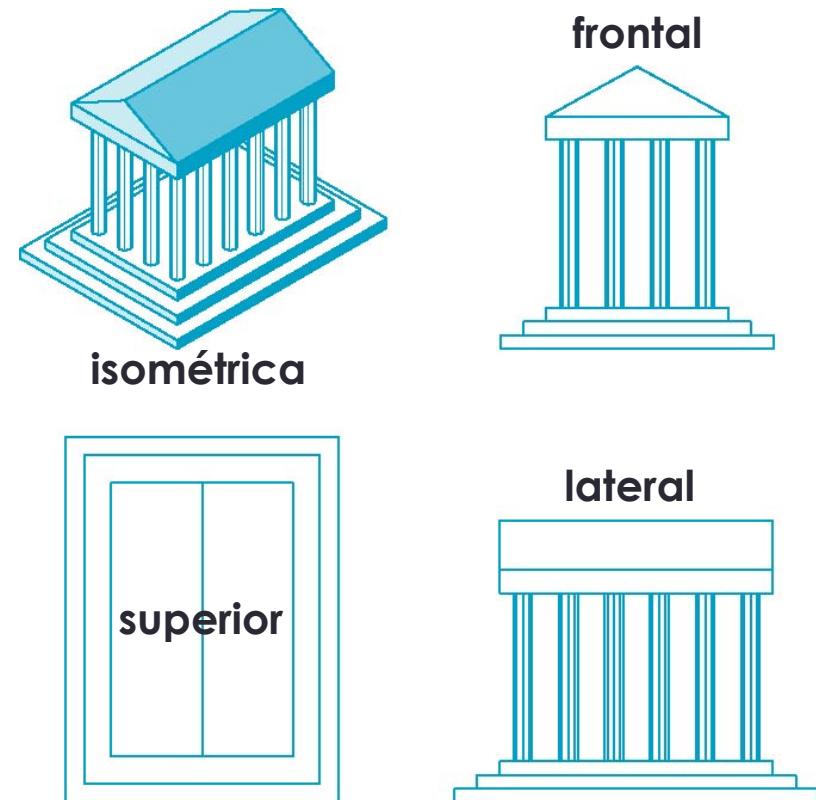
Projeção ortográfica

Raios projetores são ortogonais ao plano de projeção



Projeção ortográfica múltipla

- O plano de projeção é paralelo à face principal
- Normalmente formam-se as visões frontal, superior, e lateral
- Usado em CAD, engenharia e arquitetura
- Utilizada em conjunto com a projeção isométrica



Projeção ortográfica múltipla

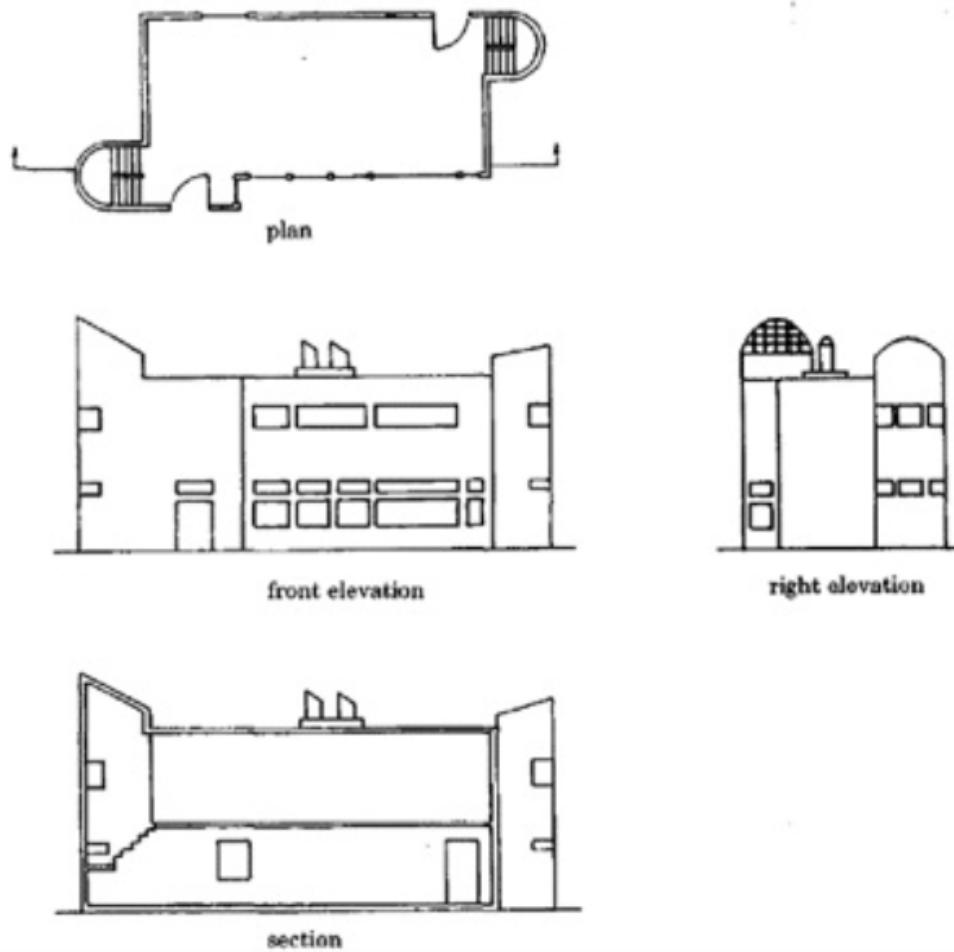


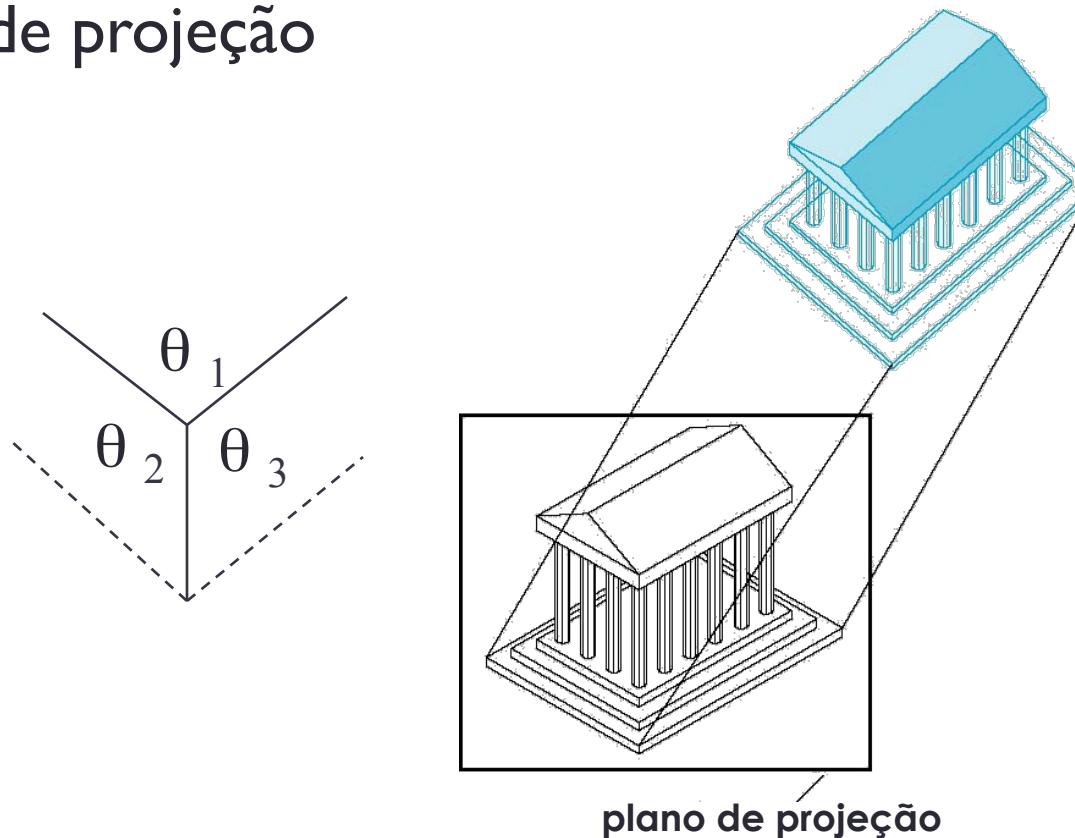
FIGURE 2-1. Multiview orthographic projection: plan, elevations, and section of a building.

Vantagens e desvantagens

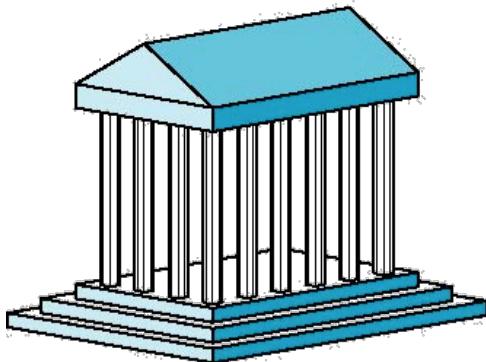
- Preservam distâncias e ângulos
 - Formas são preservadas
 - Podem ser usados para mensuração
 - Plantas (prédios, casas, etc)
 - Manuais
- Impossibilidade de visualizar como o objeto aparece em 3D
 - Quase sempre adicionamos a visão isométrica

Projeções axonométricas

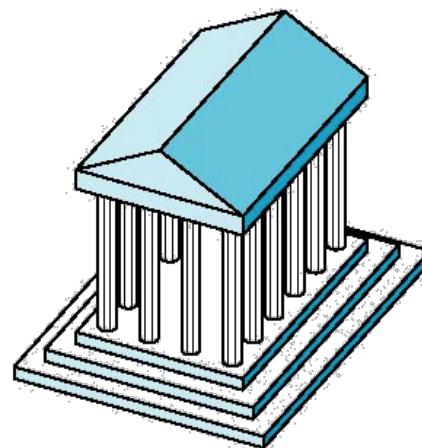
O plano de projeção pode ter qualquer orientação em relação ao(s) objeto(s), mas raios continuam ortogonais ao plano de projeção



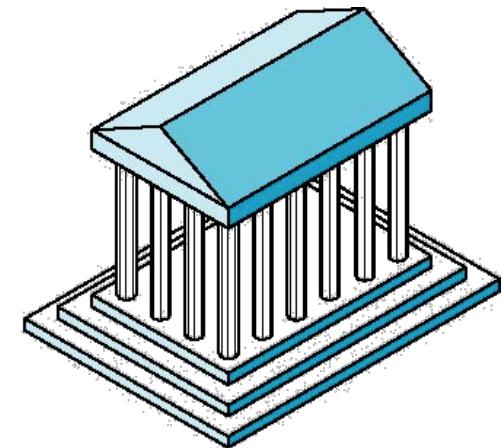
Projeções axonométricas



dimétrica



trimétrica



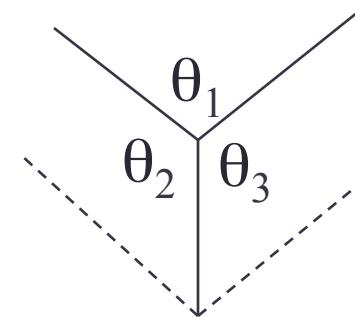
isométrica

Quantos ângulos de um canto de um cubóide projetado são iguais ?

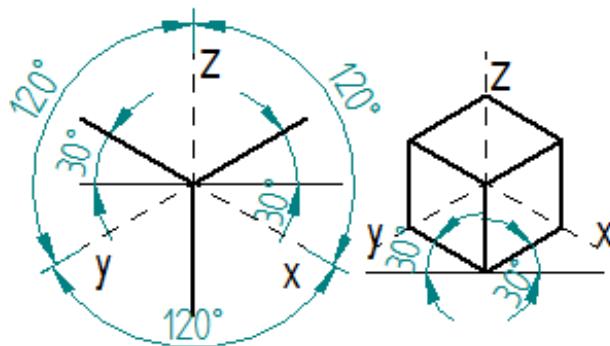
nenhum: trimétrica

dois: dimétrica

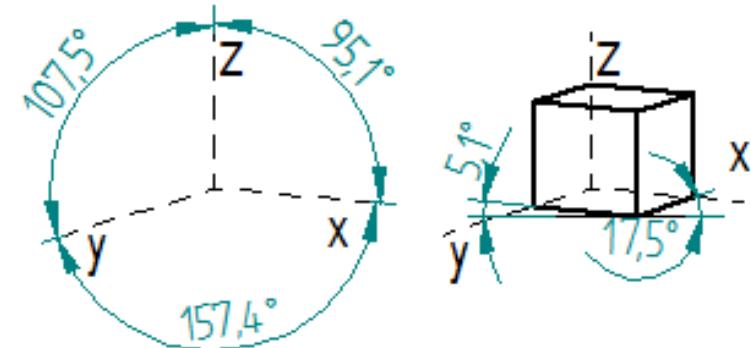
três: isométrica



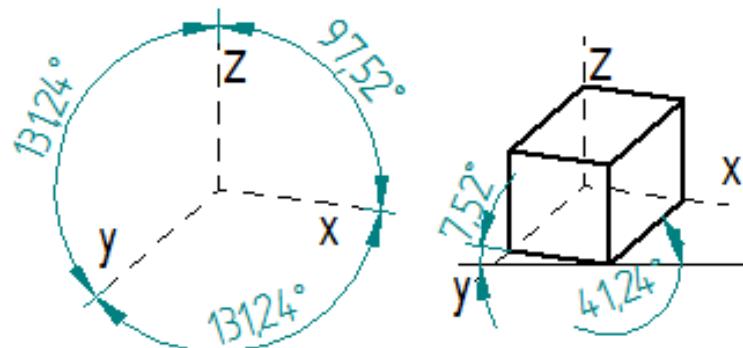
Projeções axonométricas



**perspectiva isométrica
eixos axonométricos em 120°**



**perspectiva dimétrica
dois ângulos iguais e um diferente**



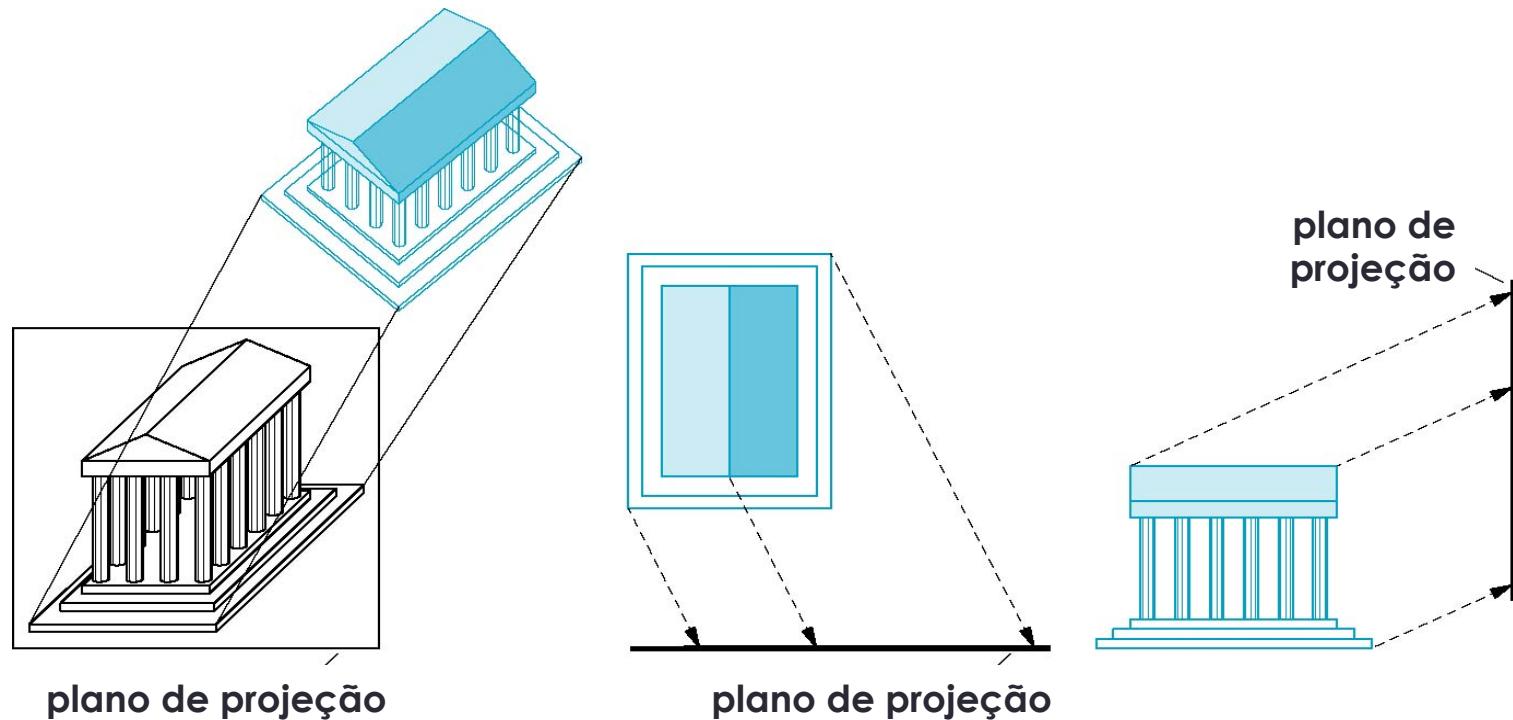
**perspectiva trimétrica
três ângulos diferentes**

Vantagens e desvantagens

- Linhas são preservadas mas são escaladas
- Ângulos não são preservados
- A projeção de um círculo em um plano não paralelo ao plano de projeção se torna uma elipse
- Permite visualizar os eixos principais de objetos
- Não parece real, pois a escala é a mesma para objetos próximos ou distantes do observador
- Normalmente utilizadas em desenhos arquitetônicos ou mecânicos.

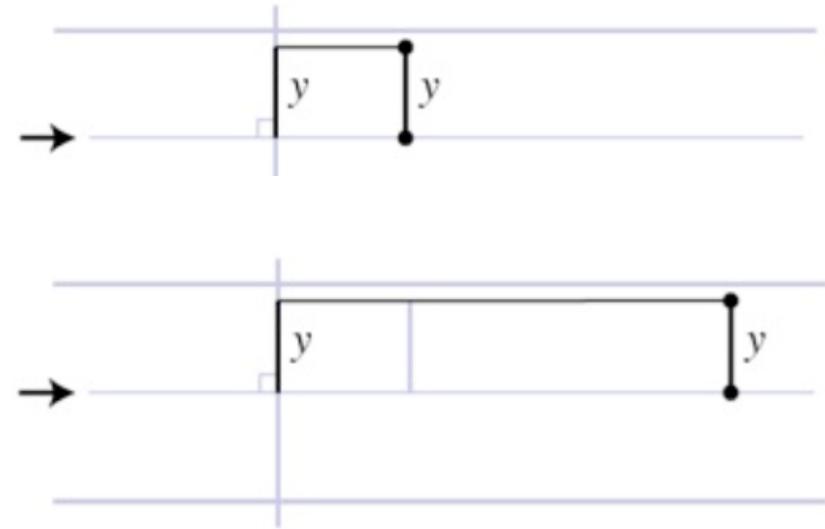
Projeção oblíqua

Relacionamento arbitrário entre os projetores e plano de projeção



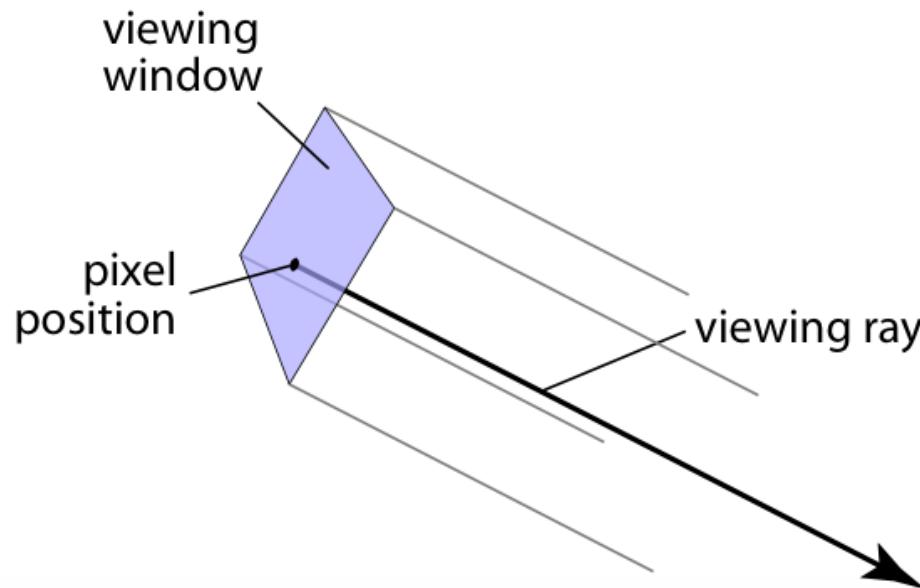
Projeção “ortográfica”

- In graphics usually we lump axonometric with orthographic
 - projection plane perpendicular to projection direction
 - image height determines size of objects in image



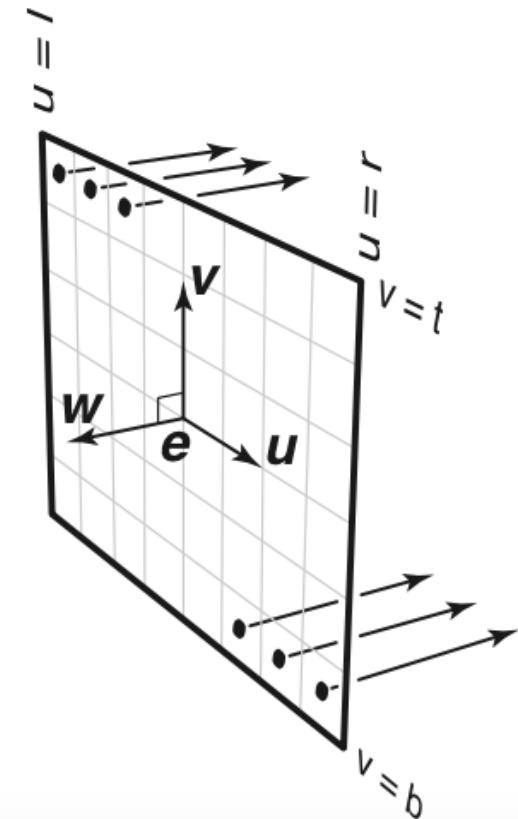
Geração de raios

- Ray origin (varying): pixel position on viewing window
- Ray direction (constant): view direction



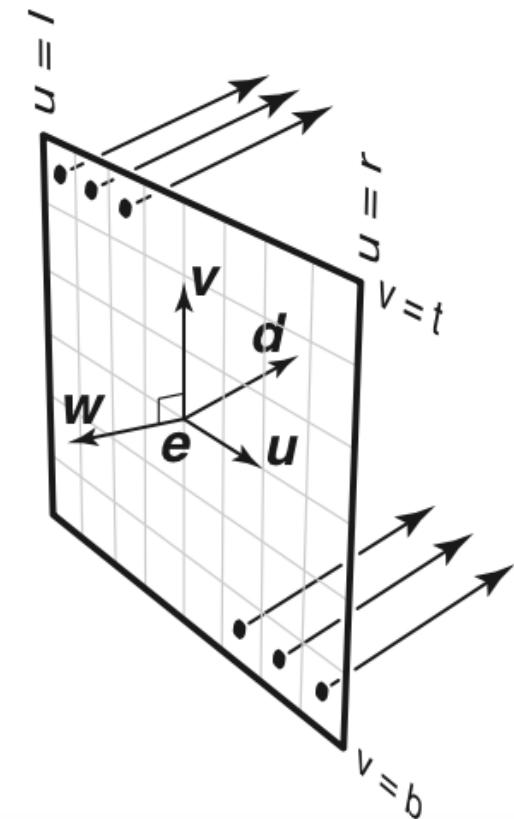
Geração de raios

- Positioning the view rectangle
 - establish three vectors to be *camera basis*: **u**, **v**, **w**
 - view rectangle is in **u**–**v** plane, specified by l, r, t, b
(often $l = -r$ and $b = -t$)
- Generating rays
 - for (u, v) in $[l, r] \times [b, t]$
 - ray.origin = **e** + $u \mathbf{u} + v \mathbf{v}$
 - ray.direction = $-\mathbf{w}$



Geração de raios- oblíqua

- View rectangle is the same
 - ray origins identical to orthographic
 - view direction \mathbf{d} differs from $-\mathbf{w}$
- Generating rays
 - for (u, v) in $[l, r] \times [b, t]$
 - ray.origin = $\mathbf{e} + u \mathbf{u} + v \mathbf{v}$
 - ray.direction = \mathbf{d}



História da projeção perspectiva

- Ancient times: Greeks wrote about laws of perspective
- Renaissance: perspective is adopted by artists



Duccio c. 1308

História das projeções



[William of Tyre's Histoire d'Outremer](#)

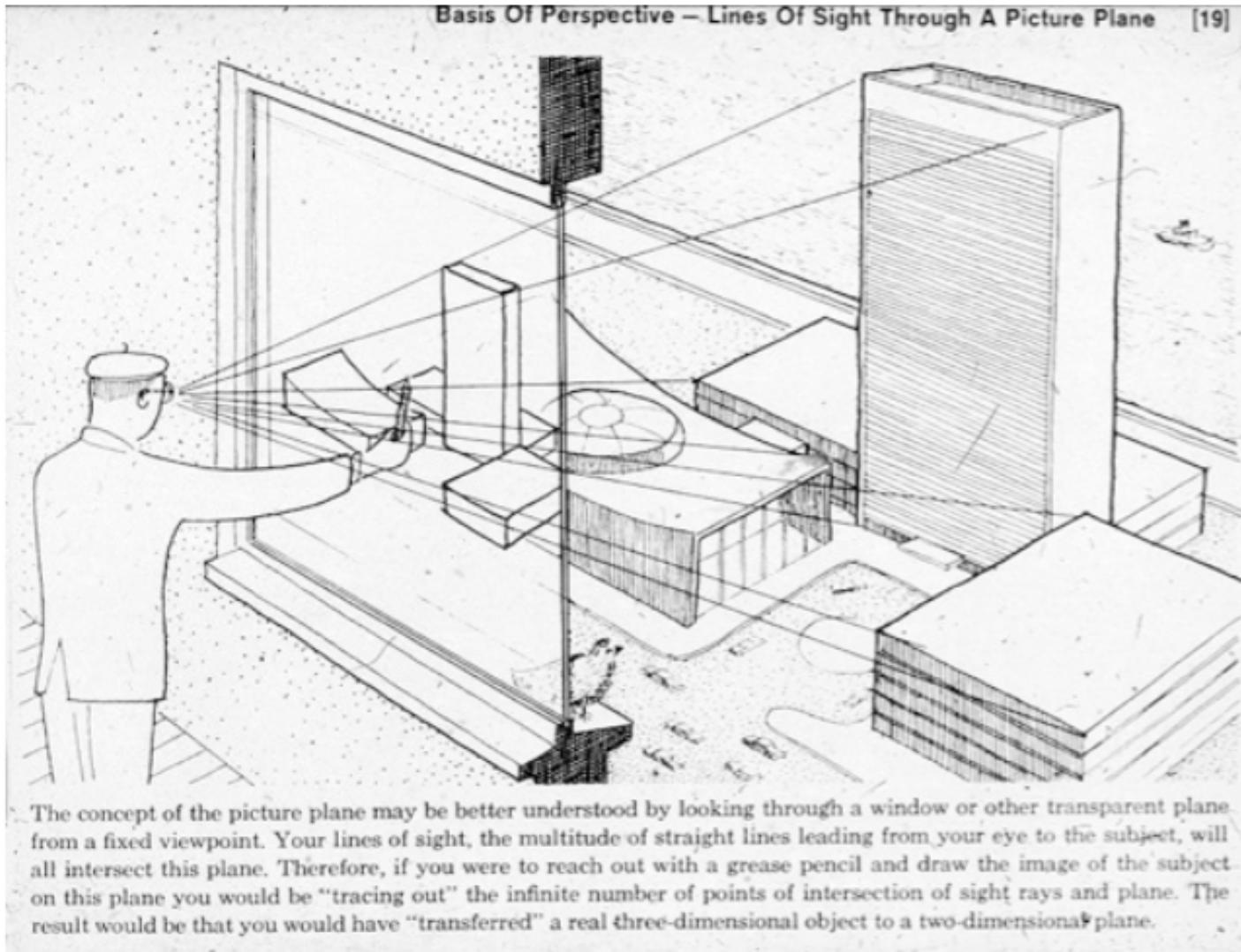
História da projeção perspectiva

- Later Renaissance: perspective formalized precisely



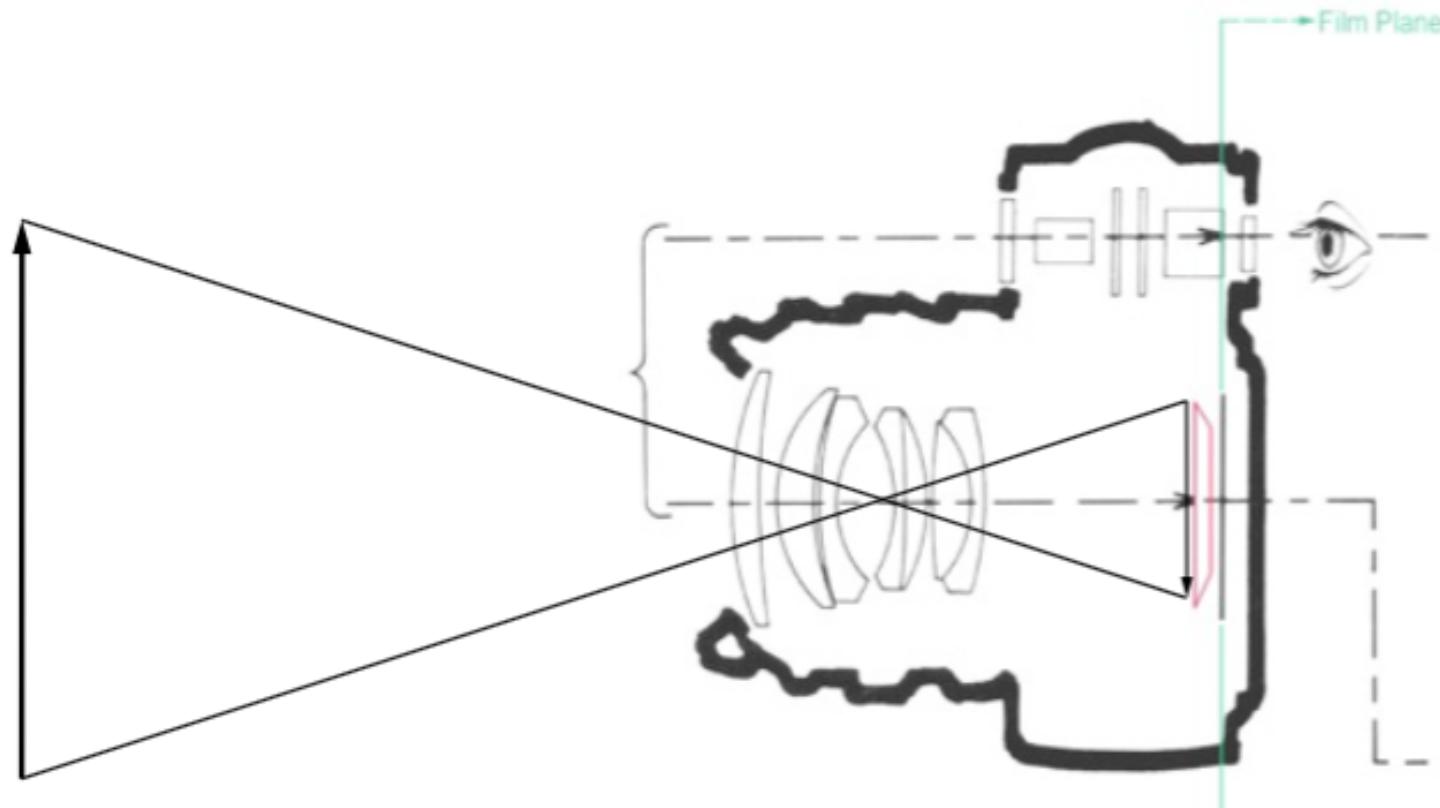
da Vinci c. 1498

Projeção planar no desenho



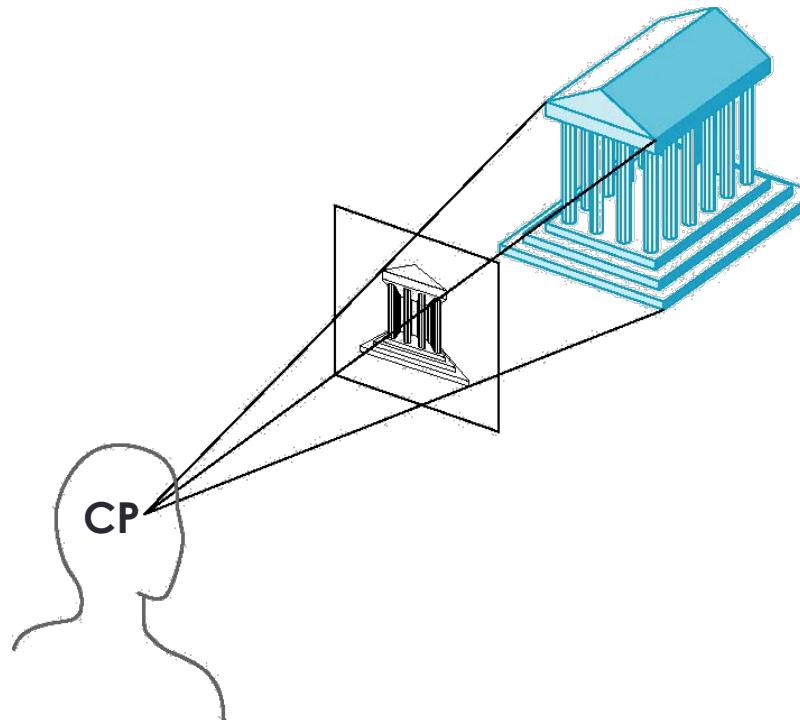
Projeção planar na fotografia

- This is another model for what we are doing
 - applies more directly in realistic rendering



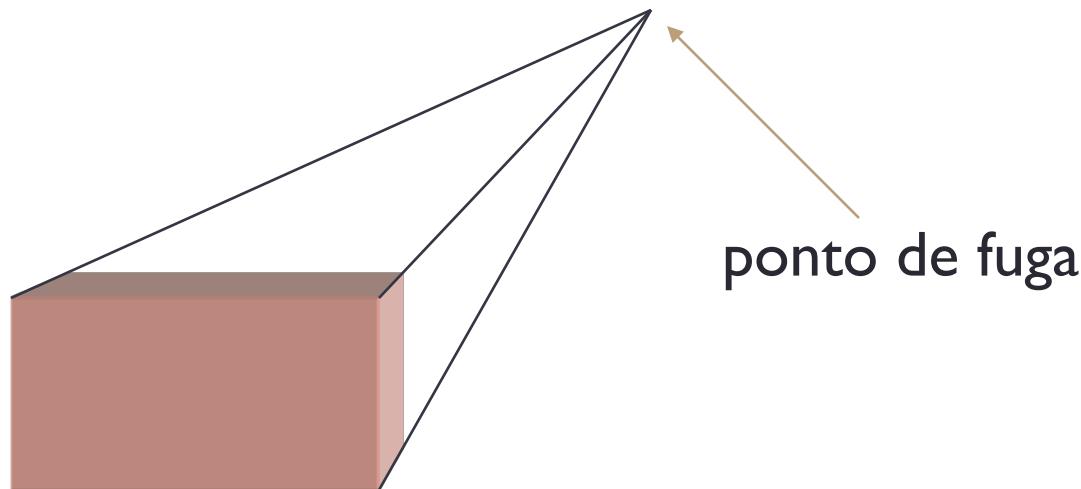
Projeção perspectiva

- Projetores convergem no centro de projeção
- Caracterizados pela diminuição de tamanho



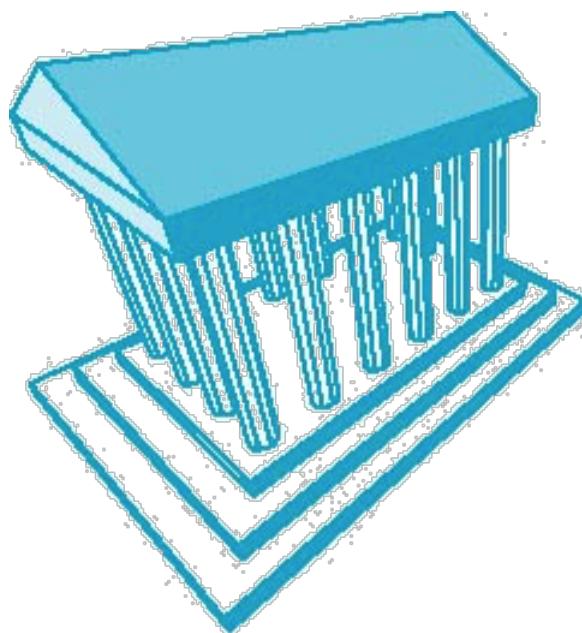
Pontos de fuga

- Linhas não paralelas ao plano de projeção partem do objeto e convergem em um único ponto (*ponto de fuga*)
- O desenho manual de perspectivas simples usam pontos de fuga



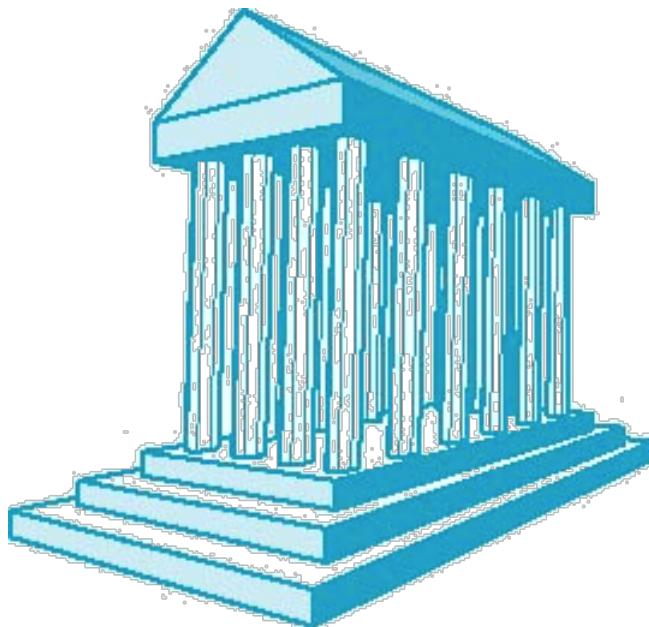
Perspectiva de 3 pontos

- Nenhuma direção principal é paralela ao plano de projeção
- Três pontos de fuga para um cubóide



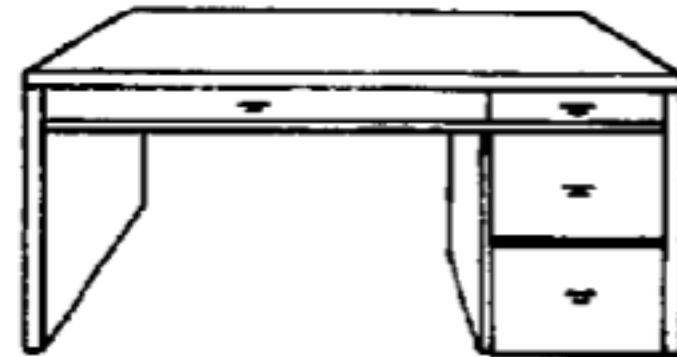
Perspectiva de 2 pontos

- Uma direção principal paralela ao plano de projeção
- Dois pontos de fuga para um cubóide



Perspectiva de 1 ponto

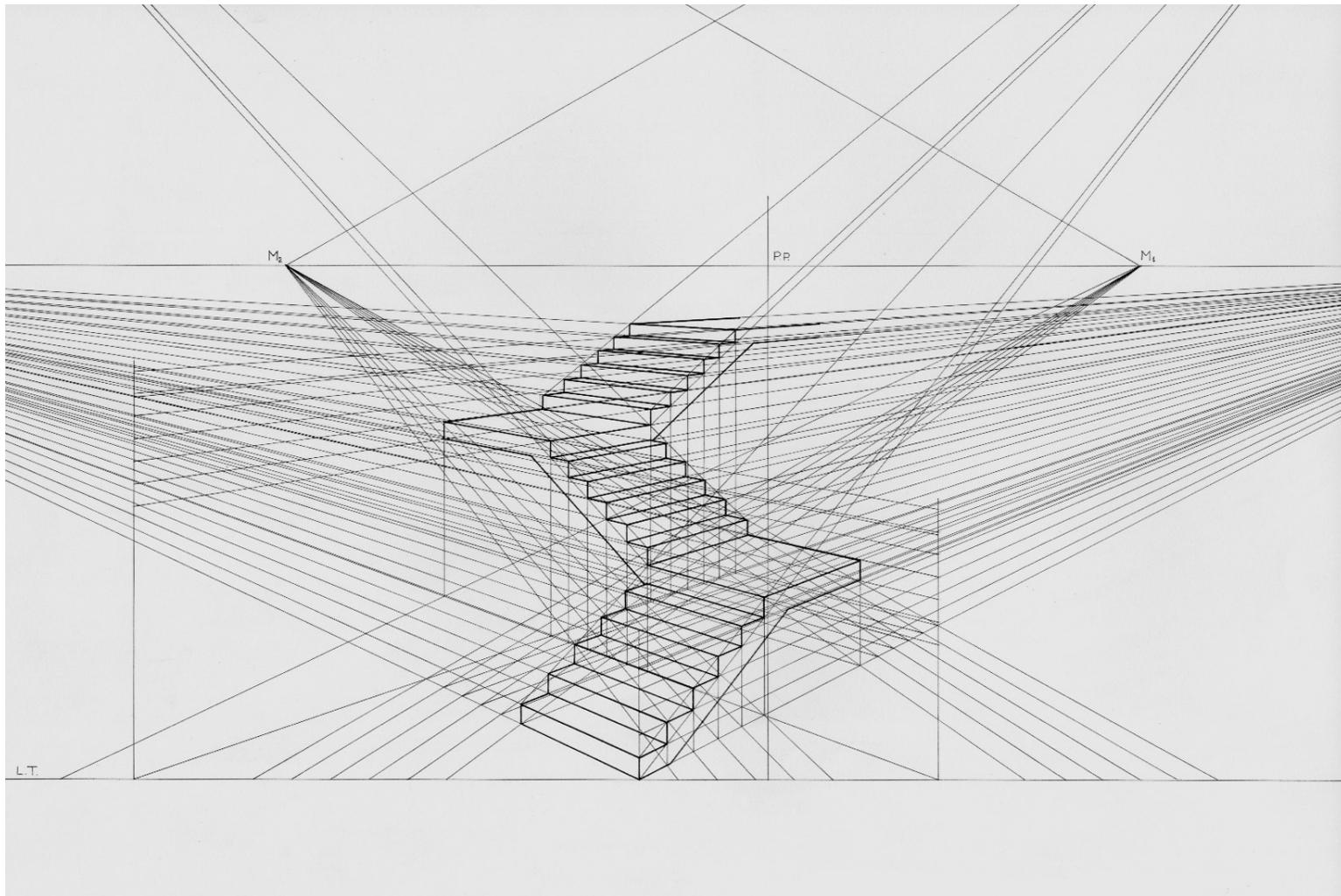
- Uma face principal paralela ao plano de projeção
- Um único ponto de fuga para um cubóide



Vantagens e desvantagens

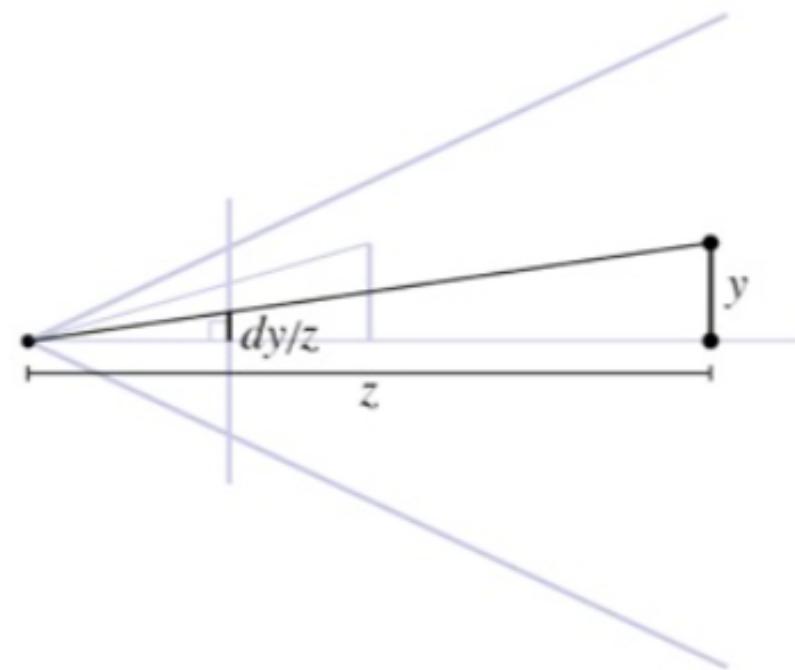
- Objetos distantes do observador são projetados em escala menor que objetos perto do observador
 - Realismo
- As projeções de duas distâncias iguais em uma linha quando projetadas não serão mais iguais
- Ângulos somente são preservados em planos paralelos ao plano de projeção
- Manualmente mais difícil de construir que as projeções ortogonais

Projeção perspectiva



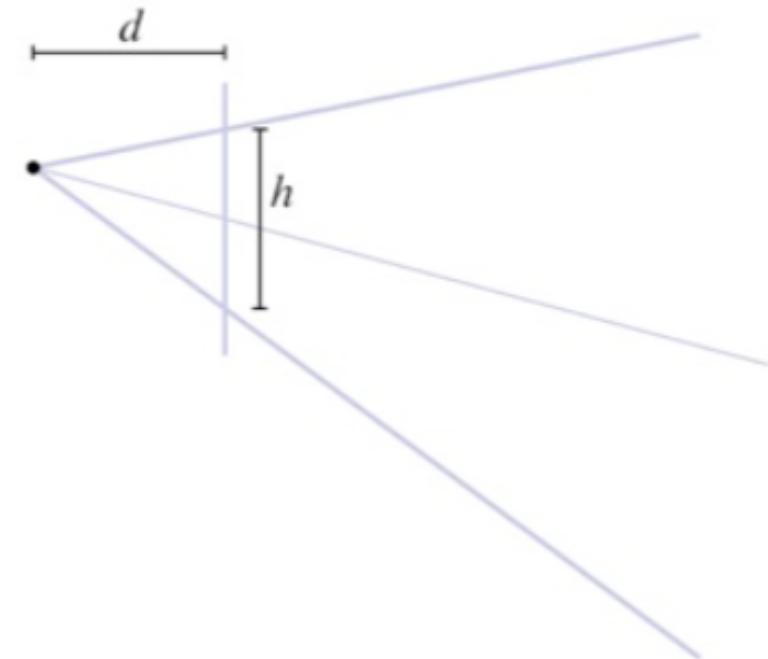
Projeção perspectiva

- Perspective is projection by lines through a point
- Magnification determined by:
 - image height
 - object depth
 - image plane distance
- FOV: $\alpha = 2 \tan(h/(2d))$
- $y' = dy/z$
- “normal” case corresponds to common types of cameras



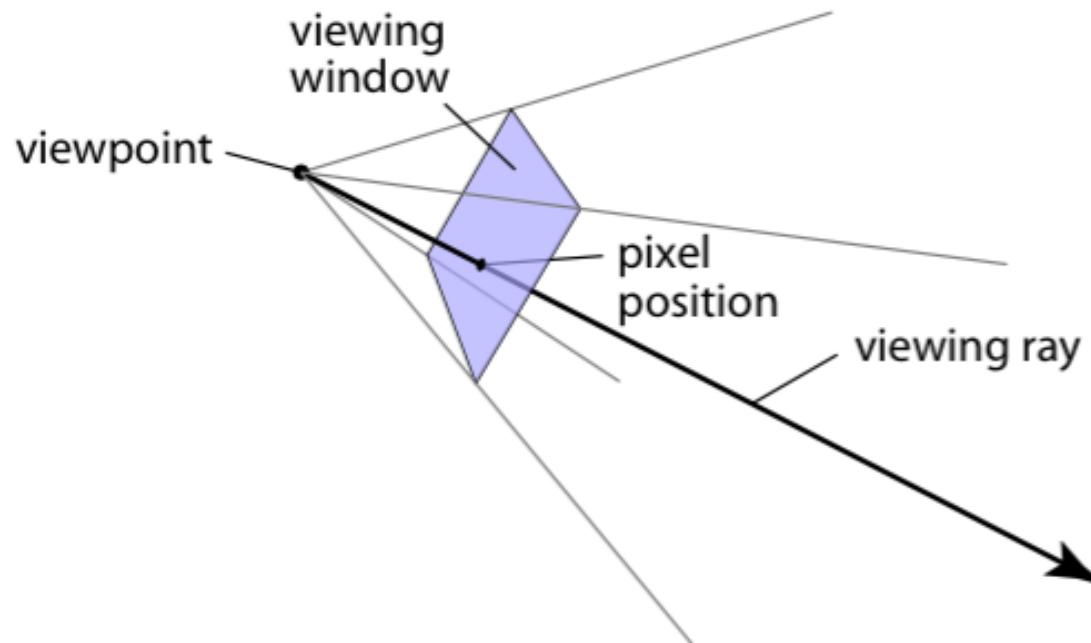
Projeção perspectiva oblíqua

- Perspective but with projection plane not perpendicular to view direction
 - additional parameter:
projection plane normal



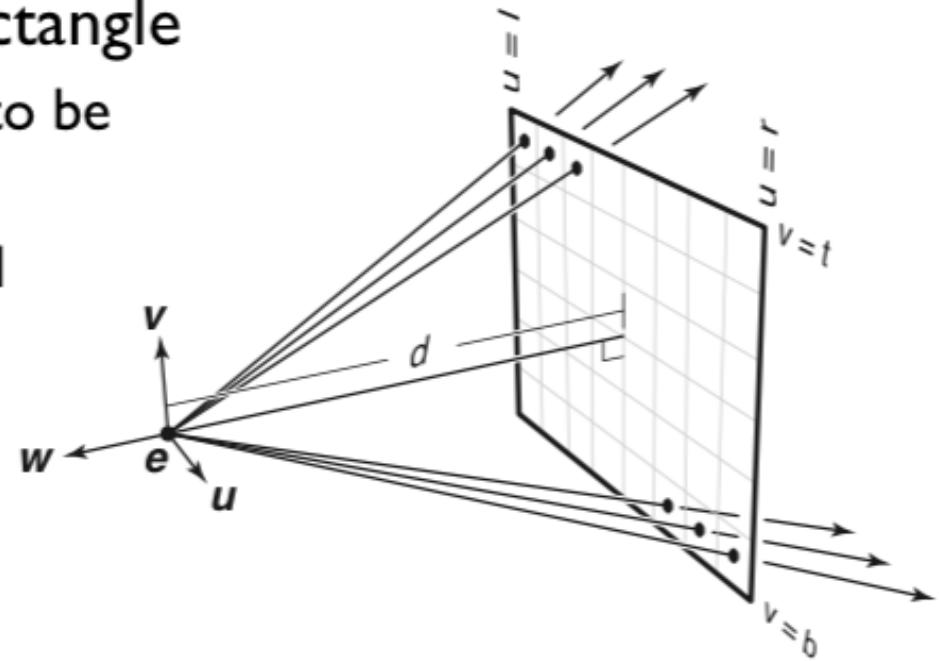
Geração de raios

- Use window analogy directly
- Ray origin (constant): viewpoint
- Ray direction (varying): toward pixel position on viewing window



Geração de raios

- Positioning the view rectangle
 - establish three vectors to be *camera basis*: \mathbf{u} , \mathbf{v} , \mathbf{w}
 - view rectangle is parallel to $\mathbf{u}-\mathbf{v}$ plane, at $w = -d$, specified by l, r, t, b
- Generating rays
 - for (u, v) in $[l, r] \times [b, t]$
 - ray.origin = \mathbf{e}
 - ray.direction = $-d \mathbf{w} + u \mathbf{u} + v \mathbf{v}$



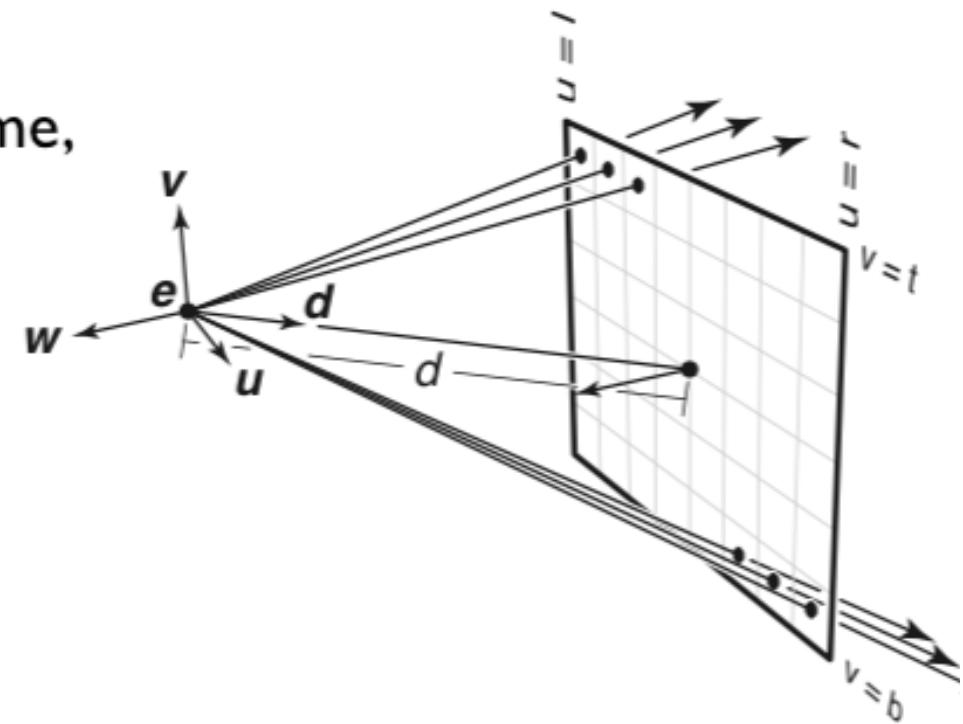
Geração de raios - oblíqua

Positioning the view rectangle

- establish three vectors to be *camera basis*: \mathbf{u} , \mathbf{v} , \mathbf{w}
- view rectangle is the same, but shifted so that the center is in the direction \mathbf{d} from \mathbf{e}

Generating rays

- for (u, v) in $[l, r] \times [b, t]$
- ray.origin = \mathbf{e}
- ray.direction = $d \mathbf{d} + u \mathbf{u} + v \mathbf{v}$



Field of view (FOV)

- The angle between the rays corresponding to opposite edges of a perspective image
 - simpler to compute for “normal” perspective
 - have to decide to measure vert., horiz., or diag.
- In cameras, determined by focal length
 - confusing because of many image sizes
 - for 35mm format (36mm by 24mm image)
 - 18mm = 67° v.f.o.v. — super-wide angle
 - 28mm = 46° v.f.o.v. — wide angle
 - 50mm = 27° v.f.o.v. — “normal”
 - 100mm = 14° v.f.o.v. — narrow angle (“telephoto”)

Field of view (FOV)

- Determines “strength” of perspective effects



close viewpoint
wide angle
prominent foreshortening



far viewpoint
narrow angle
little foreshortening

Por que perspectiva oblíqua ?

- Control convergence of parallel lines
- Standard example: architecture
 - buildings are taller than you, so you look up
 - top of building is farther away, so it looks smaller
- Solution: make projection plane parallel to facade
 - top of building is the same distance *from the projection plane*



camera tilted up: converging vertical lines



lens shifted up: parallel vertical lines

Tarefa de casa

- Leitura livro-texto
 - Shirley and Marschner. Fundamentals of Computer Graphics, CRC Press, 3rd Ed. 2010
 - Capítulo 7