Revisão Ray-casting

Elias Saraiva Barroso

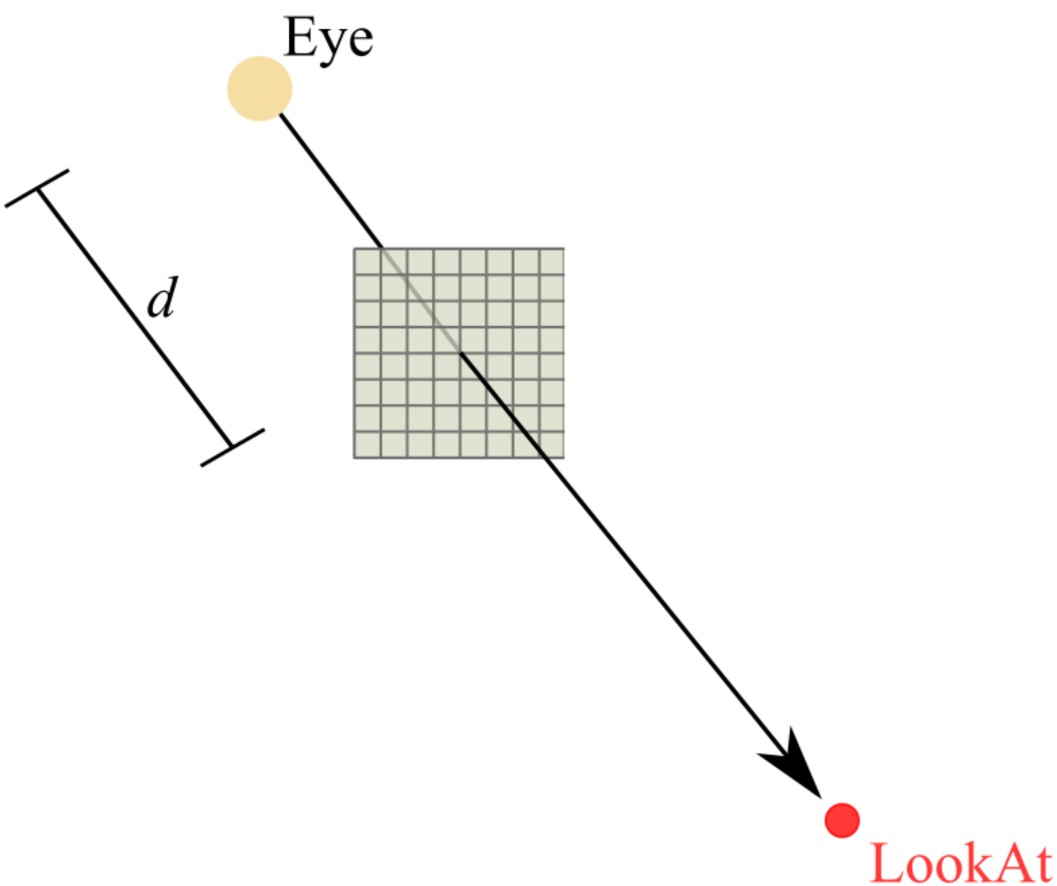
Universidade Federal do Ceará
Programa de Pós-graduaçao em Ciência da Computação
Grupo de Pesquisa CRAb

Computação Gráfica I 2017.2

Master and Doctorate Program in Computer Science

Computer graphics virtual Reality Animation

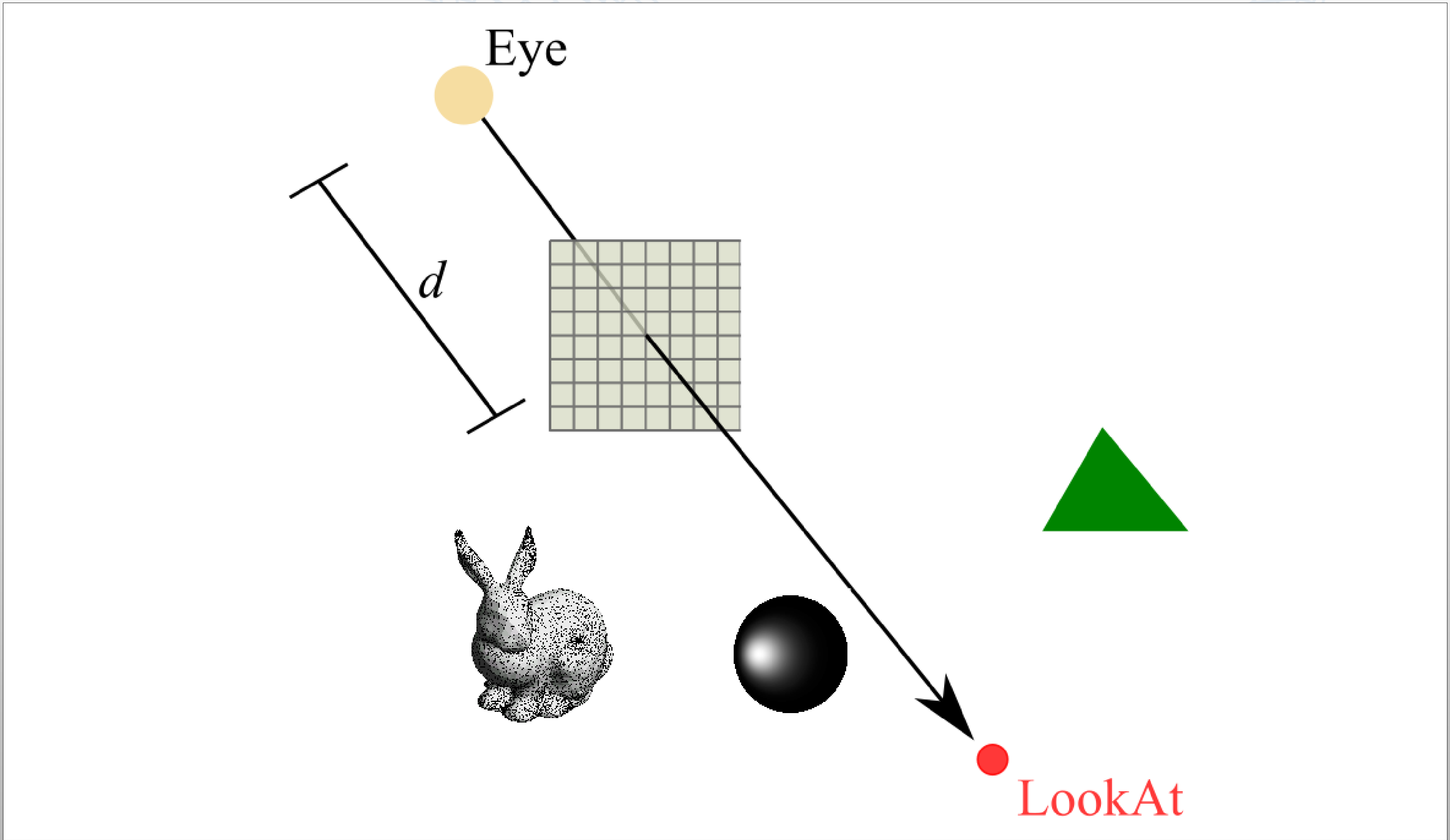
Visão Geral



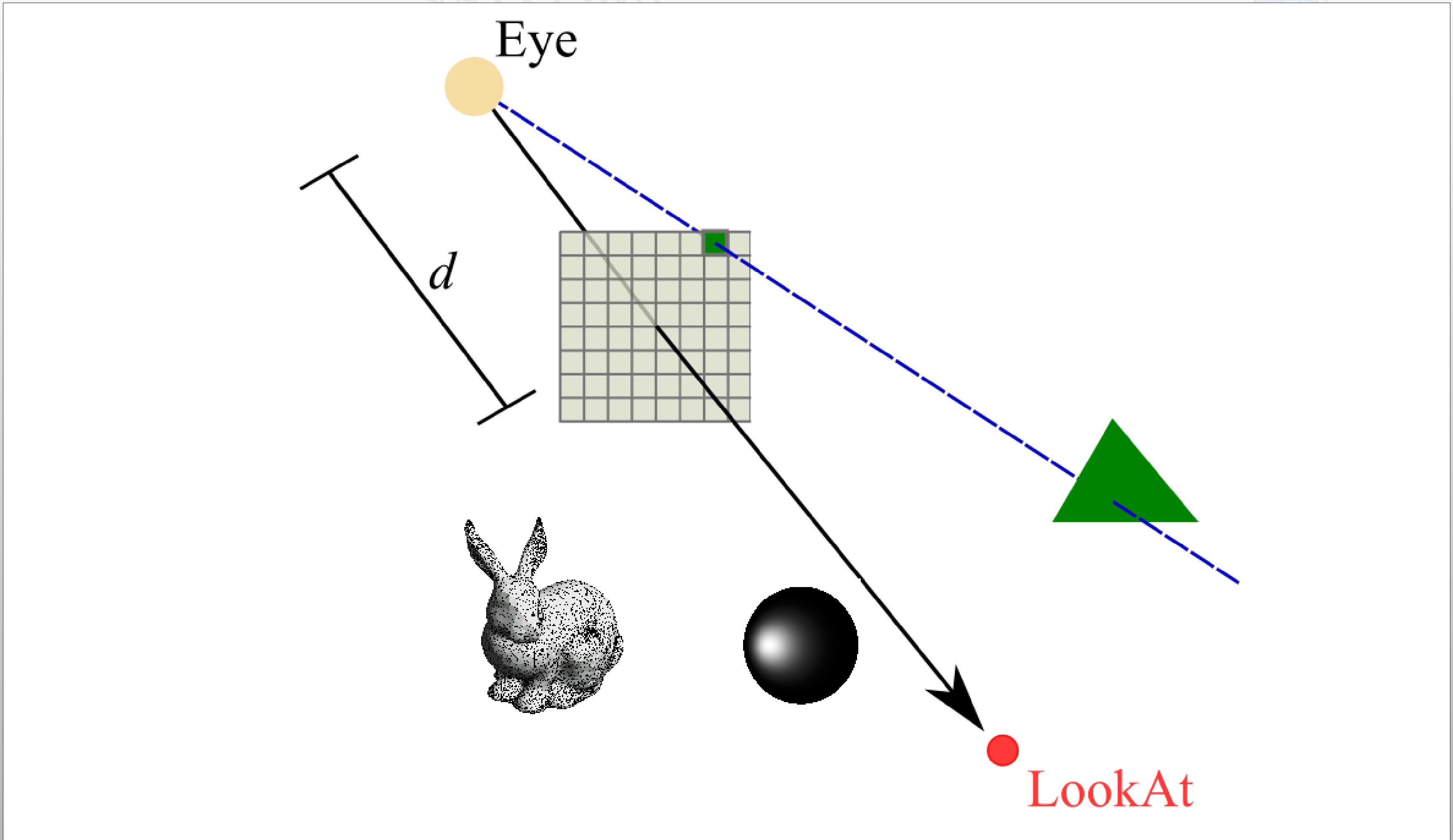
<u>DADOS</u>

Eye^w : $\{x_e, y_e, z_e, 1\}$;
ViewUp: $\{x_v, y_v, z_v, 0\}$;
d;
w;
h;
nx;
ny;

Visão Geral

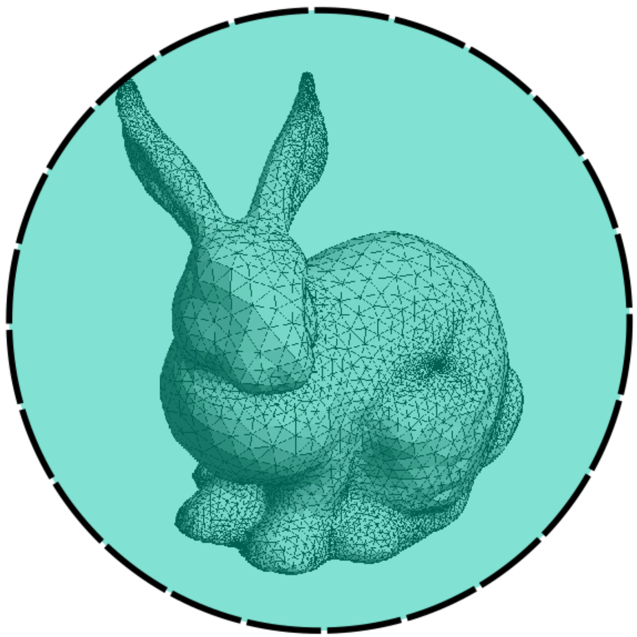


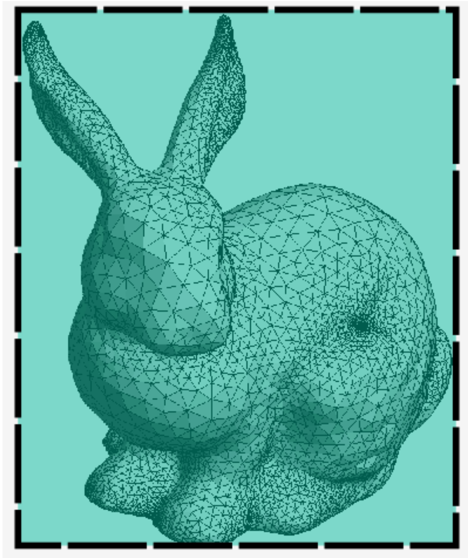
Visão Geral



Otimização

- Testa primeiro interseção com “capsula”.





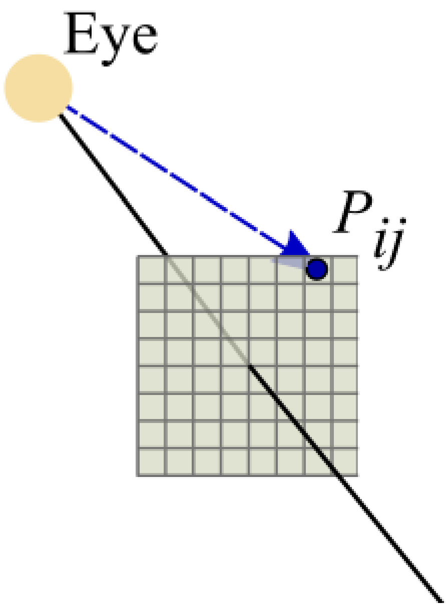
Envoltória Esférica (Bounding Sphere)

Centro = $(P_{max} + p_{min})$ *0.5;
Raio = Maior distância entre vértices ao centro;

Envoltória plana (Bounding Box)

P_{min} e P_{max} do modelo;

Equação paramétrica do Raio



$\boxed{\text{Em coordenadas de câmera}}$

$$P_{ij}.x = -w * 0.5 + w/nx * (i + 0.5);$$

$$P_{ij}.y = h * 0.5 - h/ny * (j + 0.5);$$

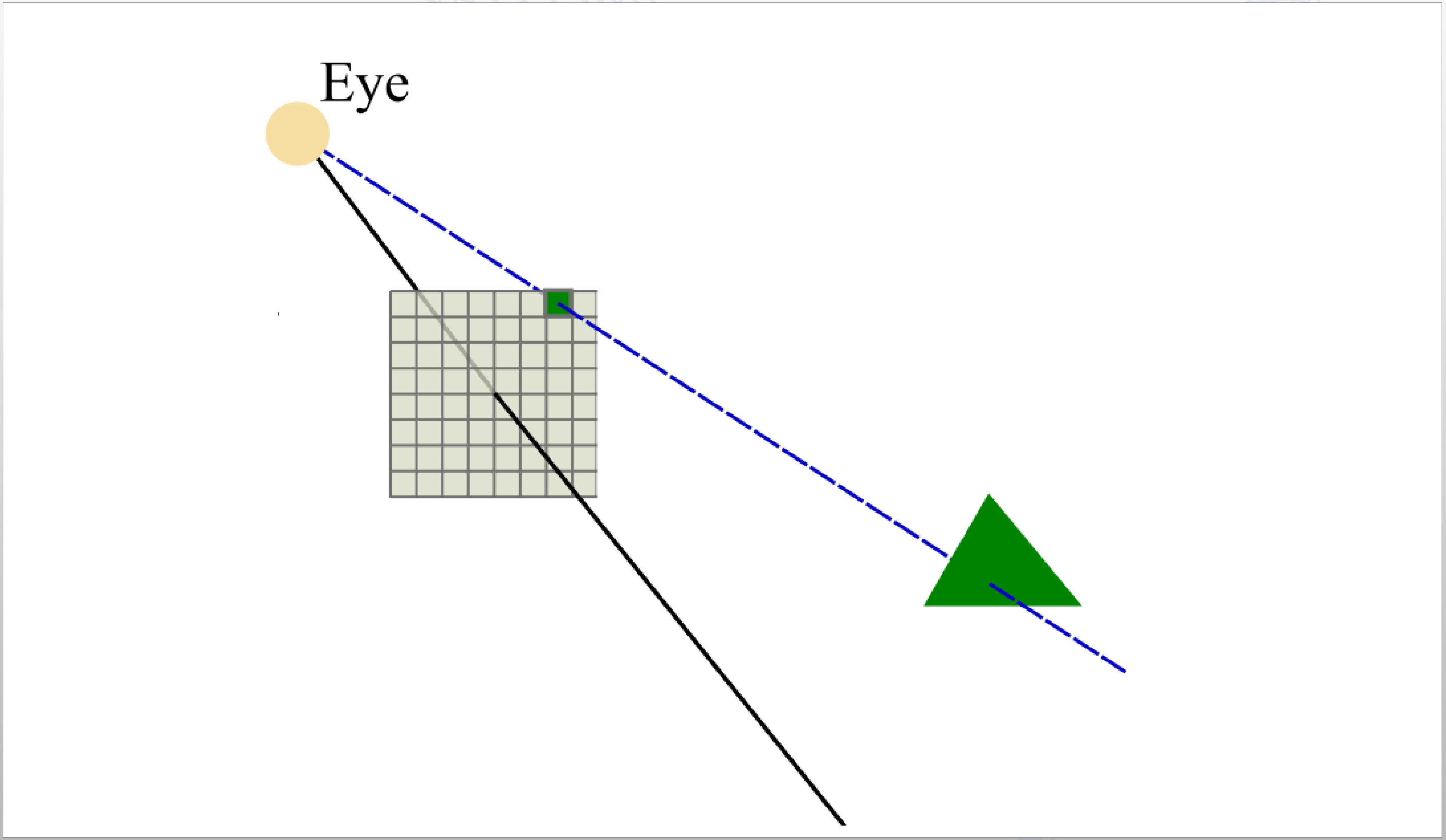
$$P_{ij}.z = -d;$$

Raio:

$$P(t) = \overset{0}{\cancel{Eye}} + t\,(P_{ij} - \overset{0}{\cancel{Eye}})$$

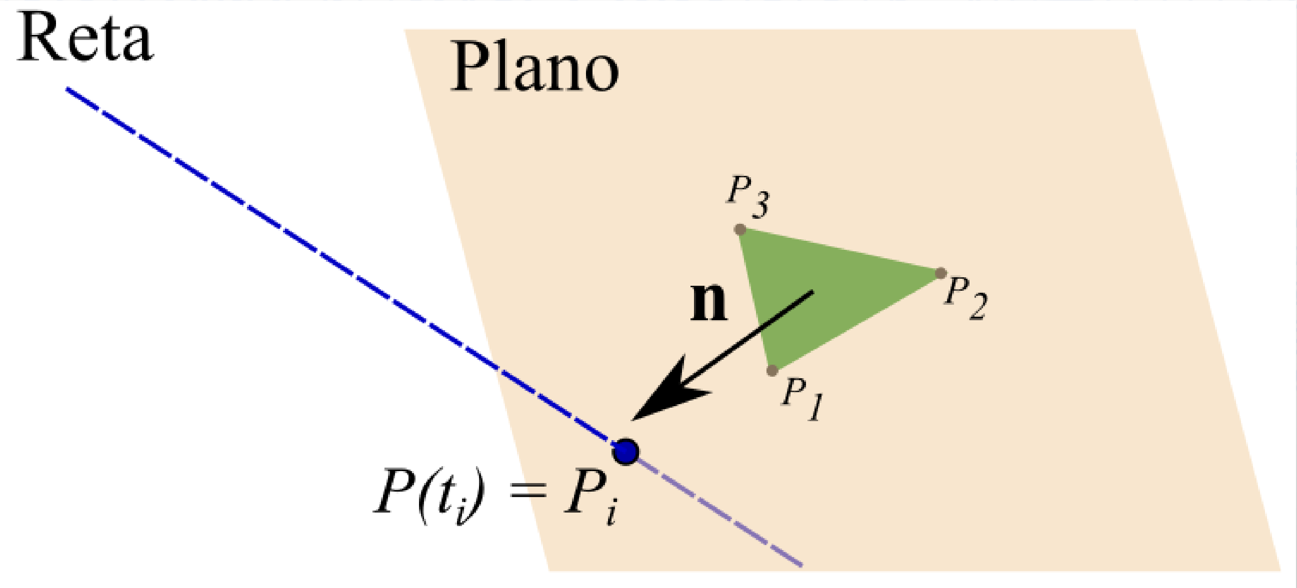
$$P(t) = t\,(P_{ij})$$

Interseção reta-triângulo



Interseção reta-triângulo

- Estratégia I:
  - **Calcular interseção reta-plano:**



  - **Ponto de interseção:**

$$t_i = \frac{n \cdot P_1}{n \cdot P_{ij}} \longrightarrow P_i = t_i P_{ij}$$

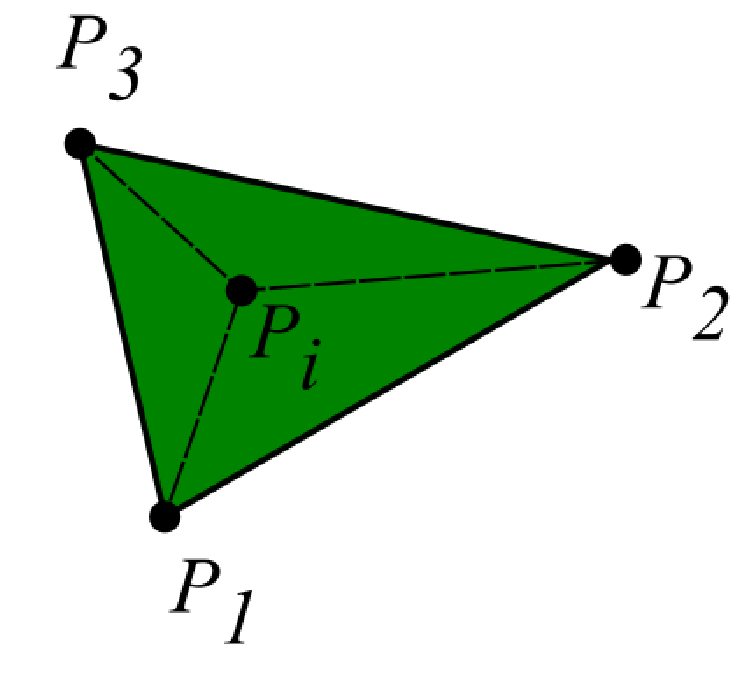
Interseção reta-triângulo

- Estratégia I:
  - **Observação:**
    - Se ($\mathbf{n} \cdot P_{ij} = 0$) → raio tangência triangulo → return false!
    - Se ($t_i < 1$) → P_i está fora do frustum → return false!
  - **Condição para P_i estar no triângulo:**
    - As Normais dos triângulos $\{P_1, P_2, P_i\}$, $\{P_2, P_3, P_i\}$ e $\{P_3, P_1, P_i\}$ devem estar no sentido de **n**.
    - Caso a condição falhe em algum dos triângulos, então o ponto não está no triângulo $\{P_1, P_2, P_3\}$.

Interseção reta-triângulo

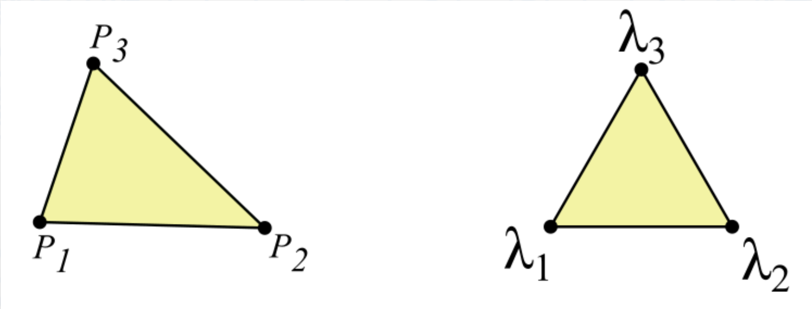
- Estratégia I:
  - **Condição para P_i estar no triângulo:**

$$n_i \text{ está no mesmo sentido que } n \text{ se}:$$
$$n_i . n > 0$$



Interseção reta-triângulo

- Estratégia II:
  - **Coordenadas baricêntricas:**
    - Tomando um triângulo {P_1, P_2 e P_3} como referência, são definidas coordenadas {λ_1,λ_2,λ_3} associada a cada ponto.



    - Pontos do sistema baricêntrico podem ser avalidos por:

$$P(\lambda_{1,}\lambda_{2,}\lambda_3)=P_1*\lambda_1+P_2*\lambda_2+P_3*\lambda_3$$

    - Com:

$$\lambda_1+\lambda_2+\lambda_3=1$$

Interseção reta-triângulo

- Estratégia II:
  - **Coordenadas baricêntricas:**
    - Considerando a restrição, a equação pode ser rescrita como:

$$\boxed{P(\lambda_{1,}\lambda_2)=P_1*\lambda_1+P_2*\lambda_2+P_3*(1-\lambda_1-\lambda_2)}$$

    - Exemplos:

$$P(1,0,0) = P_1*1 + P_2*0 + P_3*0 = P_1$$

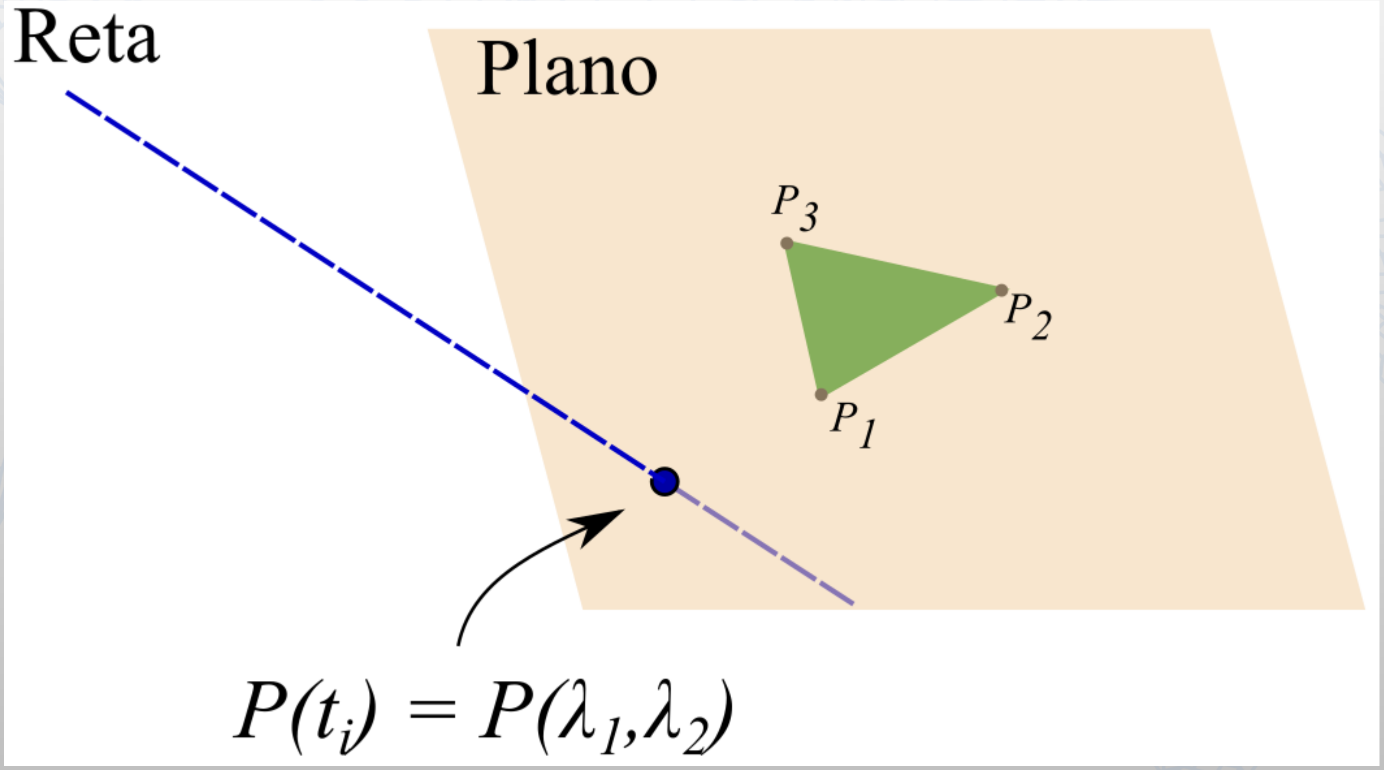
$$P(1/3,1/3,1/3) = P_1*1/3 + P_2*1/3 + P_3*1/3 = \frac{P_1+P_2+P_3}{3}$$

    - A condição para que um ponto esteja dentro do triângulo é:

$$\boxed{0\leq\lambda_i\leq1 \qquad i=1:3}$$

Interseção reta-triângulo

- Estratégia II:
  - **Calcular interseção reta-plano:**



Interseção reta-triângulo

- Estratégia II:
  - **Calcular interseção reta-plano:**
    - Igualando as equações:

$$P(t) = P(\lambda_1, \lambda_2)$$

$$t_i P_{ij} = \lambda_1 P_1 + \lambda_2 P_2 + (1 - \lambda_1 - \lambda_2) P_3$$

$$P_3 = \lambda_1 (P_3 - P_1) + \lambda_2 (P_3 - P_1) + t_i P_{ij}$$

    - A equação acima vale para x, y e z, formando um sistema linear que pode ser escrito em formato matricial:

$$P_3 = \mathbf{A}\boldsymbol{\lambda}$$

$$\begin{Bmatrix} \\ P_3 \\ \\ \end{Bmatrix} = \begin{bmatrix} \\ P_3 - P_1 & P_3 - P_2 & P_{ij} \\ \\ \end{bmatrix} \begin{Bmatrix} \lambda_1 \\ \lambda_2 \\ t_i \end{Bmatrix}$$

Interseção reta-triângulo

- Estratégia II:
  - **Coordenadas baricêntricas:**
    - Em seguida, é verificado se λ_1, λ_2, λ_2 e t_i atende os requisitos para que a interseção com o triângulo ocorra.
    - Os valores de $\{\lambda_1, \lambda_2, t_i\}$ podem ser obtidos diretamente fazendo:

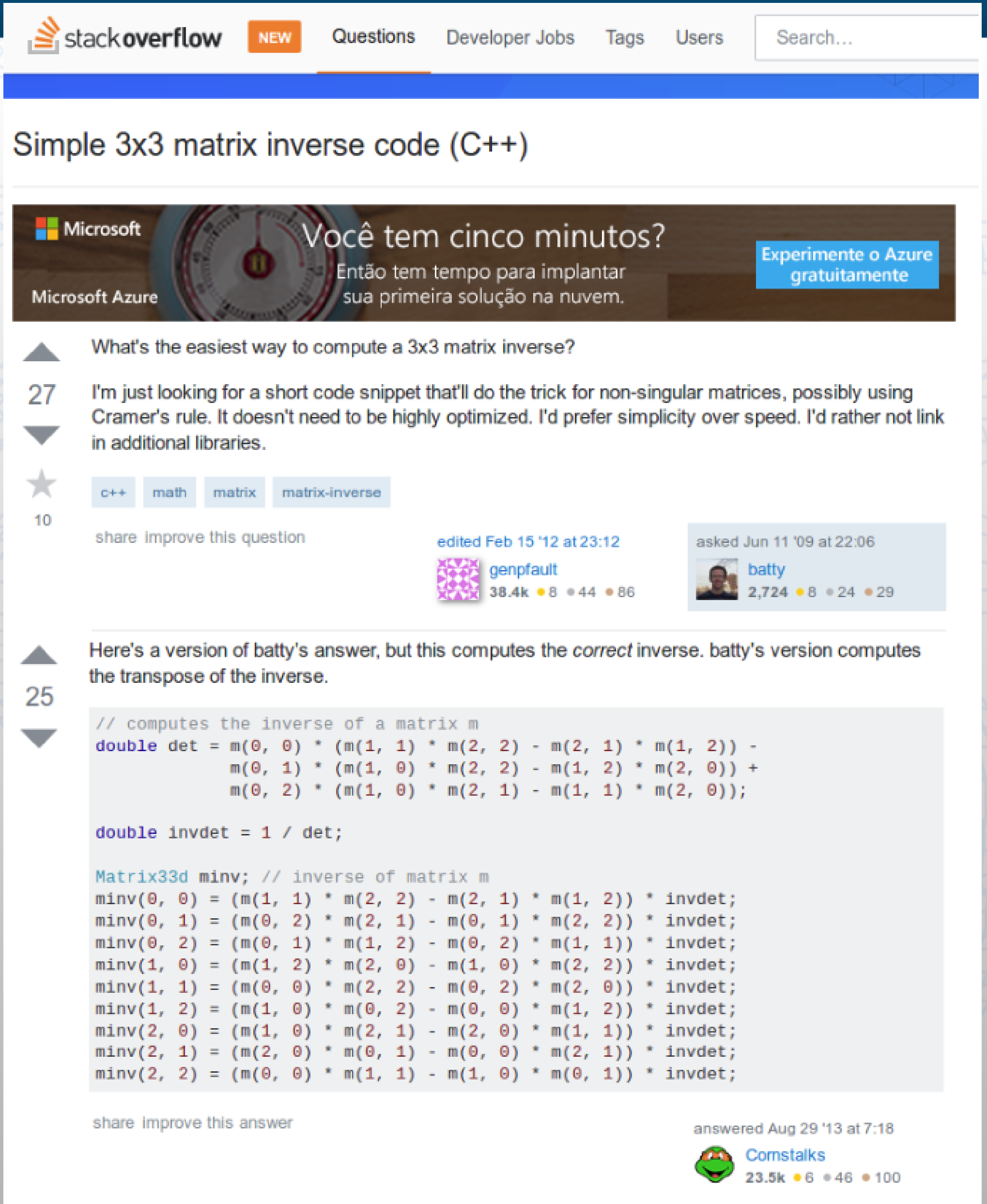
$$\mathbf{A}^{-1} P_3 = \mathbf{A}^{-1} \mathbf{A} \boldsymbol{\lambda}$$

$$\mathbf{A}^{-1} P_3 = \mathbf{I} \boldsymbol{\lambda}$$

$$\boldsymbol{\lambda} = \mathbf{A}^{-1} P_3$$

    - Como encontrar $\mathbf{A}^{-1}$ algoritmicamente? Resposta: Stackoverflow.

Interseção reta-triângulo


stackoverflow NEW Questions Developer Jobs Tags Users Search...

Simple 3x3 matrix inverse code (C++)

27

10

What's the easiest way to compute a 3x3 matrix inverse?

I'm just looking for a short code snippet that'll do the trick for non-singular matrices, possibly using Cramer's rule. It doesn't need to be highly optimized. I'd prefer simplicity over speed. I'd rather not link in additional libraries.

c++ math matrix matrix-inverse

share improve this question

edited Feb 15 '12 at 23:12
genpfault
38.4k ●8 ●44 ●86

asked Jun 11 '09 at 22:06
batty
2,724 ●8 ●24 ●29

25

Here's a version of batty's answer, but this computes the *correct* inverse. batty's version computes the transpose of the inverse.

```
// computes the inverse of a matrix m
double det = m(0, 0) * (m(1, 1) * m(2, 2) - m(2, 1) * m(1, 2)) -
             m(0, 1) * (m(1, 0) * m(2, 2) - m(1, 2) * m(2, 0)) +
             m(0, 2) * (m(1, 0) * m(2, 1) - m(1, 1) * m(2, 0));

double invdet = 1 / det;

Matrix33d minv; // inverse of matrix m
minv(0, 0) = (m(1, 1) * m(2, 2) - m(2, 1) * m(1, 2)) * invdet;
minv(0, 1) = (m(0, 2) * m(2, 1) - m(0, 1) * m(2, 2)) * invdet;
minv(0, 2) = (m(0, 1) * m(1, 2) - m(0, 2) * m(1, 1)) * invdet;
minv(1, 0) = (m(1, 2) * m(2, 0) - m(1, 0) * m(2, 2)) * invdet;
minv(1, 1) = (m(0, 0) * m(2, 2) - m(0, 2) * m(2, 0)) * invdet;
minv(1, 2) = (m(1, 0) * m(0, 2) - m(0, 0) * m(1, 2)) * invdet;
minv(2, 0) = (m(1, 0) * m(2, 1) - m(2, 0) * m(1, 1)) * invdet;
minv(2, 1) = (m(2, 0) * m(0, 1) - m(0, 0) * m(2, 1)) * invdet;
minv(2, 2) = (m(0, 0) * m(1, 1) - m(1, 0) * m(0, 1)) * invdet;
```

share improve this answer

answered Aug 29 '13 at 7:18
Cornstalks
23.5k ●6 ●46 ●100

Interseção reta-triângulo

- Estratégia II:
  - **Código para inverte matrix:**

```
// computes the inverse of a matrix m
double det = m(0, 0) * (m(1, 1) * m(2, 2) - m(2, 1) * m(1, 2)) -
             m(0, 1) * (m(1, 0) * m(2, 2) - m(1, 2) * m(2, 0)) +
             m(0, 2) * (m(1, 0) * m(2, 1) - m(1, 1) * m(2, 0));

double invdet = 1 / det;

Matrix33d minv; // inverse of matrix m
minv(0, 0) = (m(1, 1) * m(2, 2) - m(2, 1) * m(1, 2)) * invdet;
minv(0, 1) = (m(0, 2) * m(2, 1) - m(0, 1) * m(2, 2)) * invdet;
minv(0, 2) = (m(0, 1) * m(1, 2) - m(0, 2) * m(1, 1)) * invdet;
minv(1, 0) = (m(1, 2) * m(2, 0) - m(1, 0) * m(2, 2)) * invdet;
minv(1, 1) = (m(0, 0) * m(2, 2) - m(0, 2) * m(2, 0)) * invdet;
minv(1, 2) = (m(1, 0) * m(0, 2) - m(0, 0) * m(1, 2)) * invdet;
minv(2, 0) = (m(1, 0) * m(2, 1) - m(2, 0) * m(1, 1)) * invdet;
minv(2, 1) = (m(2, 0) * m(0, 1) - m(0, 0) * m(2, 1)) * invdet;
minv(2, 2) = (m(0, 0) * m(1, 1) - m(1, 0) * m(0, 1)) * invdet;
```

Fim

Obrigado pela sua atenção!

Bom Almoço!