IA369E

Tópicos em Engenharia de Computação VI

Segundo Semestre de 2013

Visualização Exploratória

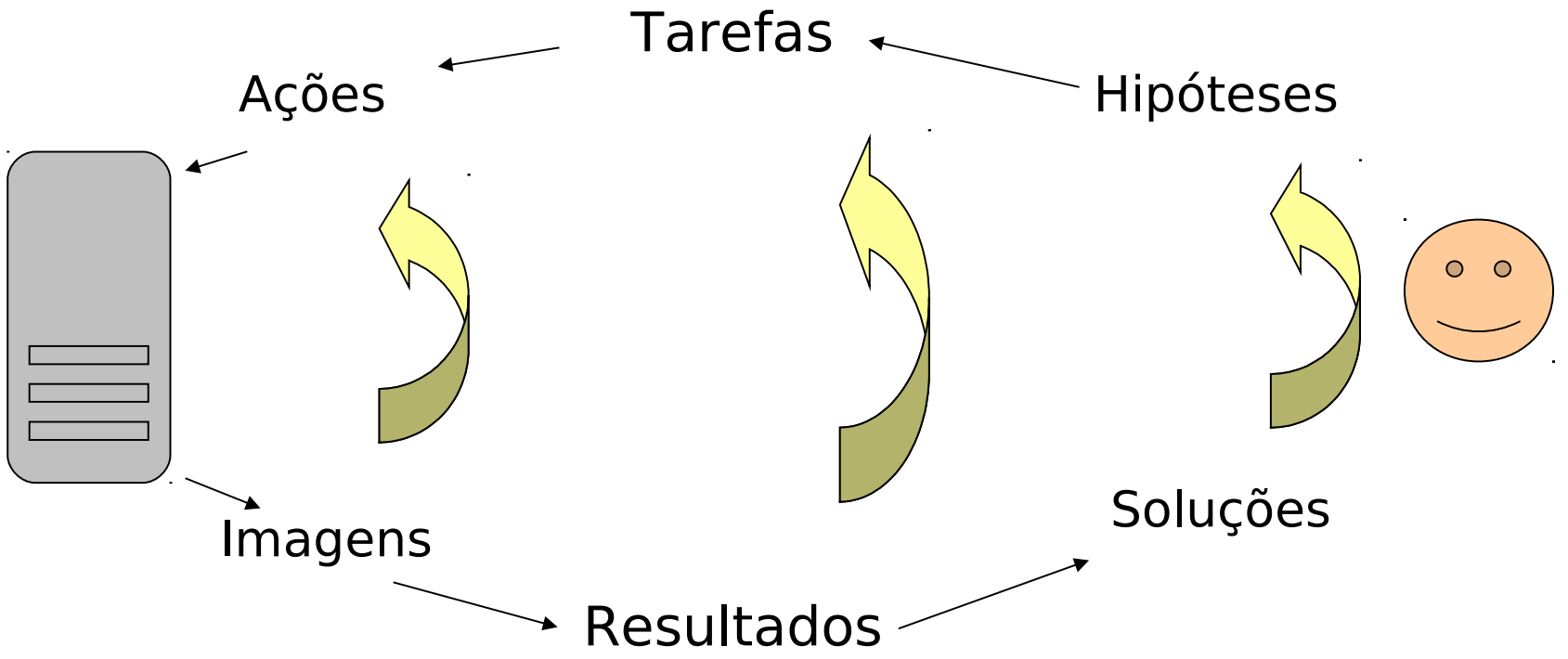
Profa. Ting

Uma demo

http://www.youtube.com/watch?v=KJErdrrtXsQ

Visualização Exploratória

- Laço de solução de problema
- Laço de exploração e navegação
- Laço de manipulação direta
  - responsividade dos fragmentos



Manipulações Diretas

- Ações
  - Similares às realizadas no mundo físico
  - Reversíveis
  - Incrementais
- Realimentação visual
  - Contínua
  - Em tempo interativo
  - Dentro da expectativa

Tarefas de Interação Básicas

- Posicionamento: (x,y) ou (x,y,z)
- Seleção
  - um item de uma lista
  - um item de uma imagem (*Picking*)
- “Texto”: uma cadeia de caracteres
- Quantificação: um valor
- Interações 3D
  - Posicionamento
  - Seleção

Fonte: Computer Graphics: Principles and Practice, Foley e outros

Interface Gráfica do Usuário (GUI)

- Dispõe de um conjunto de *widgets* para realizar tarefas de interação compostas
- Interação com os dispositivos de entrada e de saída através de elementos gráficos
    - Dispositivos predominantemente 2D

Glut

- Eventos de teclado
  http://www.swiftless.com/tutorials/opengl/keyboard.html

- Eventos de mouse

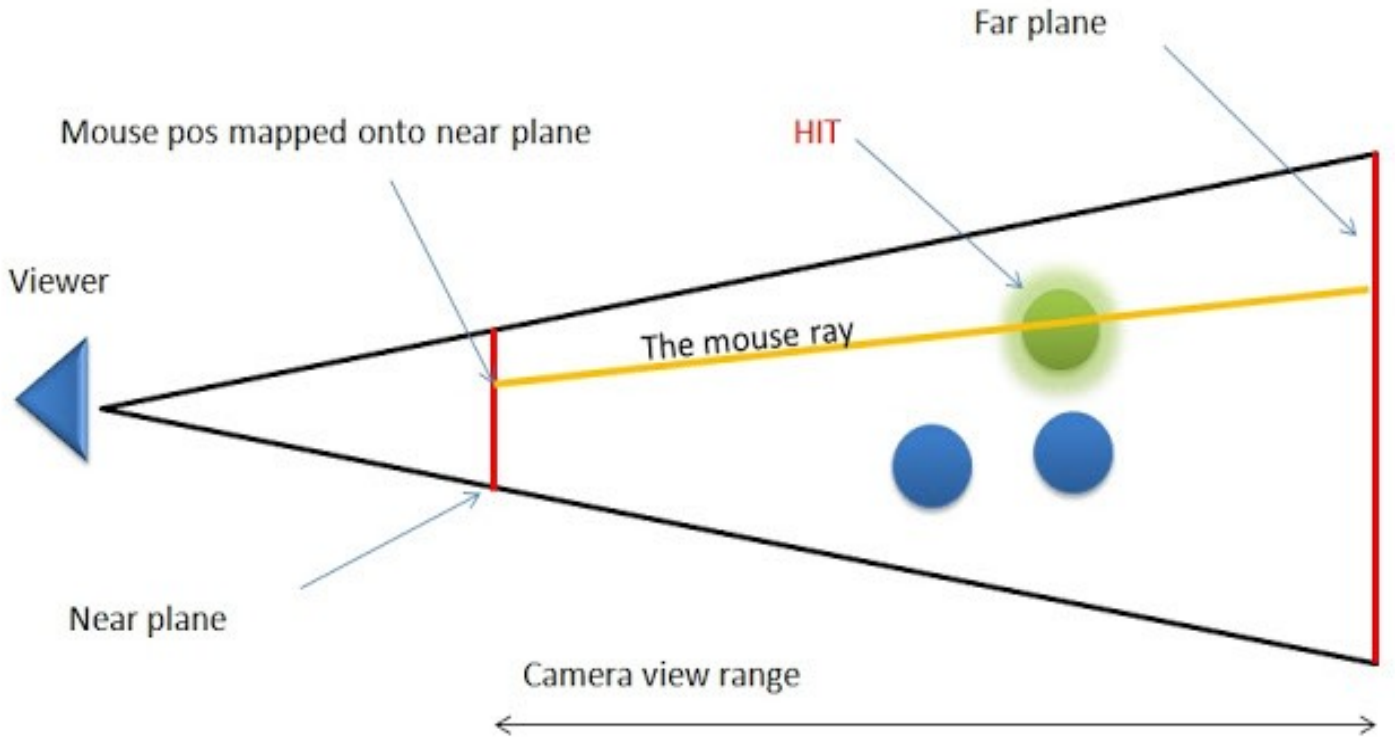
  http://www.lighthouse3d.com/opengl/glut/index.php3?9

Glut: Eventos de *Mouse*

```
main () {
    glutMouseFunc (MouseButton);
    glutMotionFunc (MouseMotion);
}

void MouseButton(int button, int state, int x, int y) {
    if (button == GLUT_LEFT_BUTTON) {
    } else if (button == GLUT_RIGHT_BUTTON && state ==   GLUT_DOWN)
    {
    } else if (button == 3) {  // Mouse wheel
    } else if (button == 4) {  // Mouse wheel
}

void MouseMotion(int x, int y) {
}
```

OpenGL

- Mecanismo de *picking* e *selection buffer*



http://www.lighthouse3d.com/opengl/picking/

Projeções em OpenGL

$$(anatomica_x, anatomica_y, anatomica_z)$$

GL_MODELVIEW

$$(cena_x, cena_y, cena_z)$$

GL_PROJECTION

$$(n_x, n_y, n_z)$$

$$win_x = \frac{(n_x + 1)\, viewport[2]}{2} + viewport[0]$$

$$win_y = \frac{(n_y + 1)\, viewport[3]}{2} + viewport[1]$$

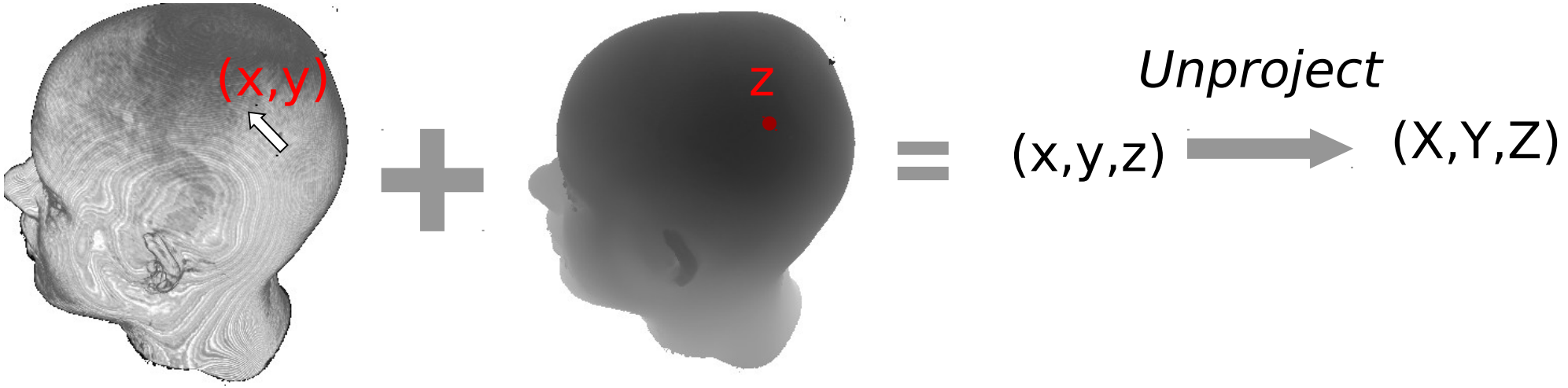
$$win_z = \frac{n_z + 1}{2}$$

$$(win_x, win_y, win_z)$$

glutUnproject

Rotações 3D com *Mouse* 2D

- Teclado → *Mouse*
  - Ação 2D (x,y)
  - Mapeamento (x,y) em (X,Y,Z) sobre a “superfície” visível
  - Dois pontos subsequentes em ângulo de rotação
  - Realimentação visual do volume de dados



Mapa de profundidade

Arcball

- Cômputo do quatérnio

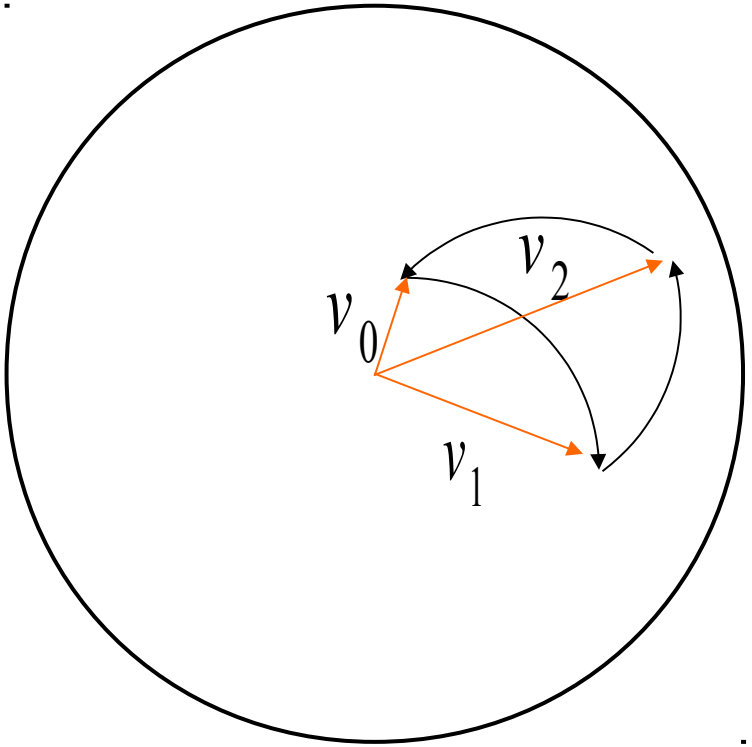
$$q_{arrasta} = [x, y, z, w] = [\vec{v}_0 \times \vec{v}_1, \vec{v}_0 \cdot \vec{v}_1] = [\vec{v}_r \, sen\theta, \cos\theta]$$

- Composição Recorrente

$$q_{nova} = q_{arrasta} \, q_{aperta}; \quad q_{aperta} = q_{nova}$$

- Matriz de Rotação

$$q = \begin{bmatrix} 1-2y^2-2z^2 & 2xy-2zw & 2xz+2yw & 0 \\ 2xy+2zw & 1-2x^2-2z^2 & 2yz-2zw & 0 \\ 2xz-2yw & 2yz+2xw & 1-2x^2-2y^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Controle de um *Cursor* 3D com *Mouse*

- Botão direito solto:

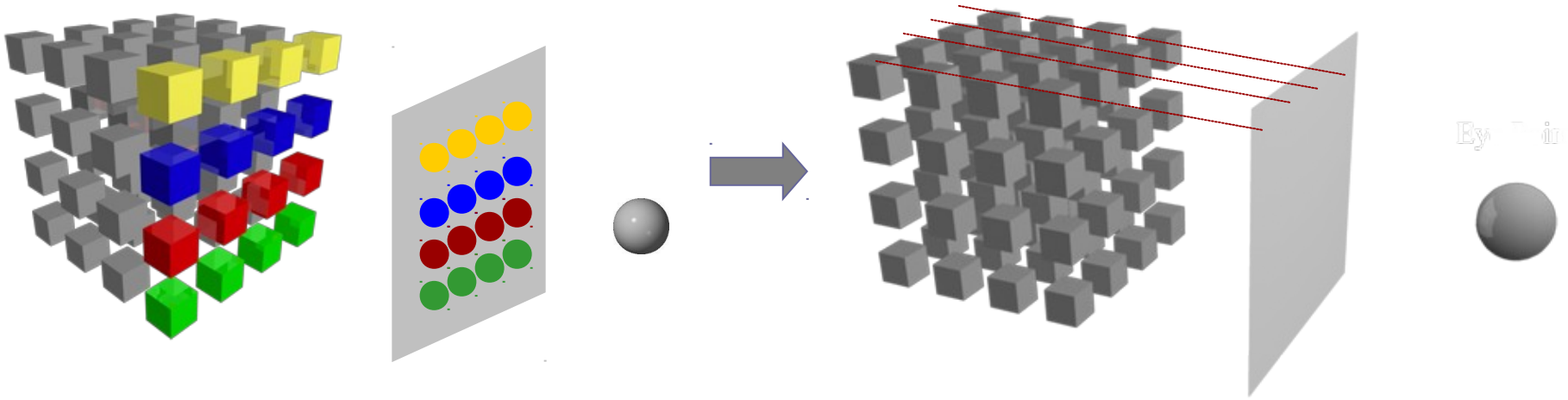
$$\begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} = View^{-1}\, Pj^{-1} \begin{bmatrix} x_i \\ y_i \\ z_i = z_{i-1} \end{bmatrix}$$

- Botão direito pressionado:

$$\begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} = View^{-1}\, Pj^{-1} \begin{bmatrix} x_i \\ y_i \\ z_i = z_{i-1} + \dfrac{\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}}{4\mathrm{D}} \end{bmatrix}$$

Mapa de Profundidade

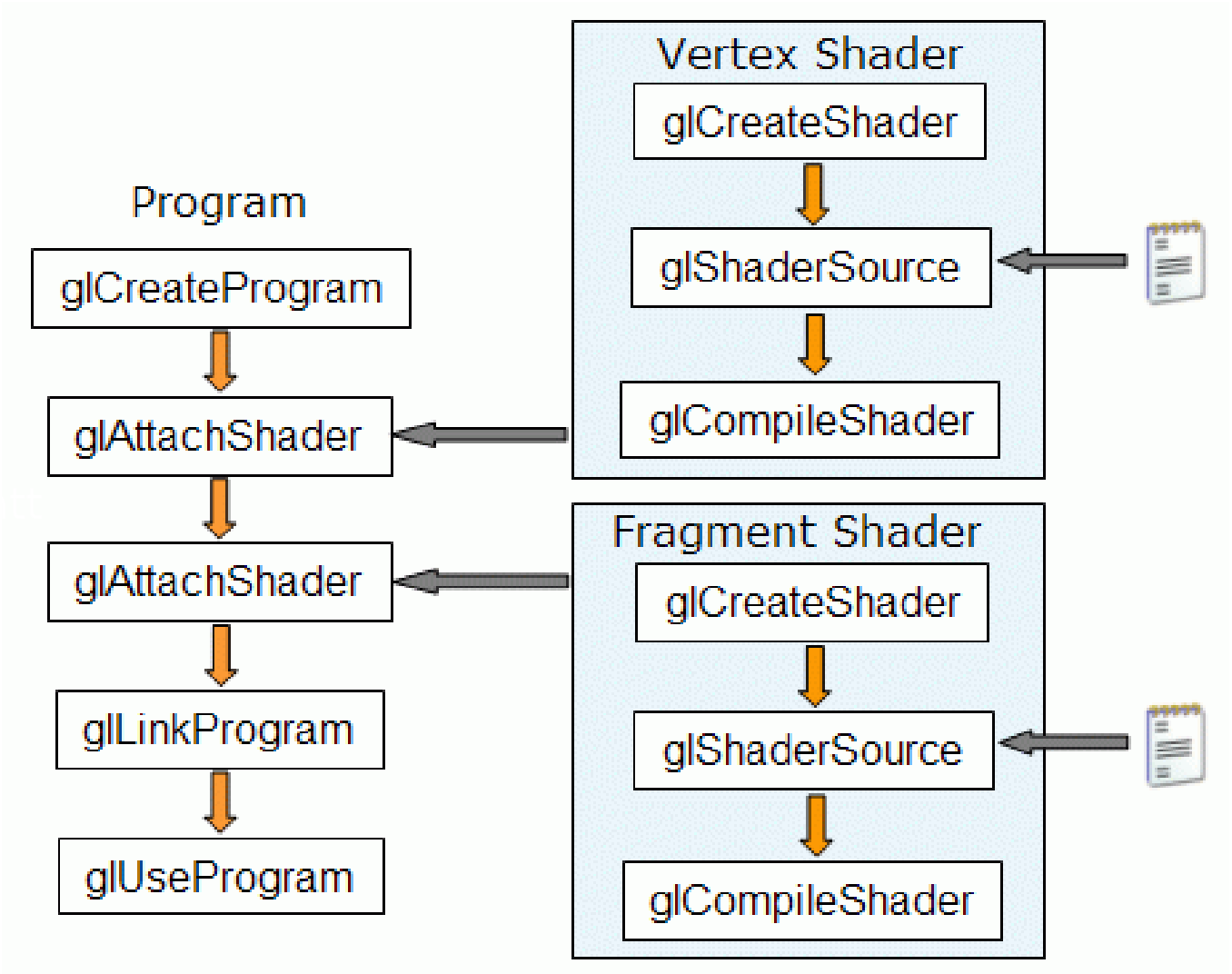
- Reprogramar o fluxo de renderização
  - Ao invés dos *texels* correspondentes, "texturizamos" cada fragmento com a profundidade do ponto de interesse visível.



fragmento ↔ *texel*

fragmento ↔ *ponto de interesse visível*

OpenGL: Fluxo Programável



Fonte: http://www.lighthouse3d.com/tutorials/glsl-tutorial/opengl-setup-for-g

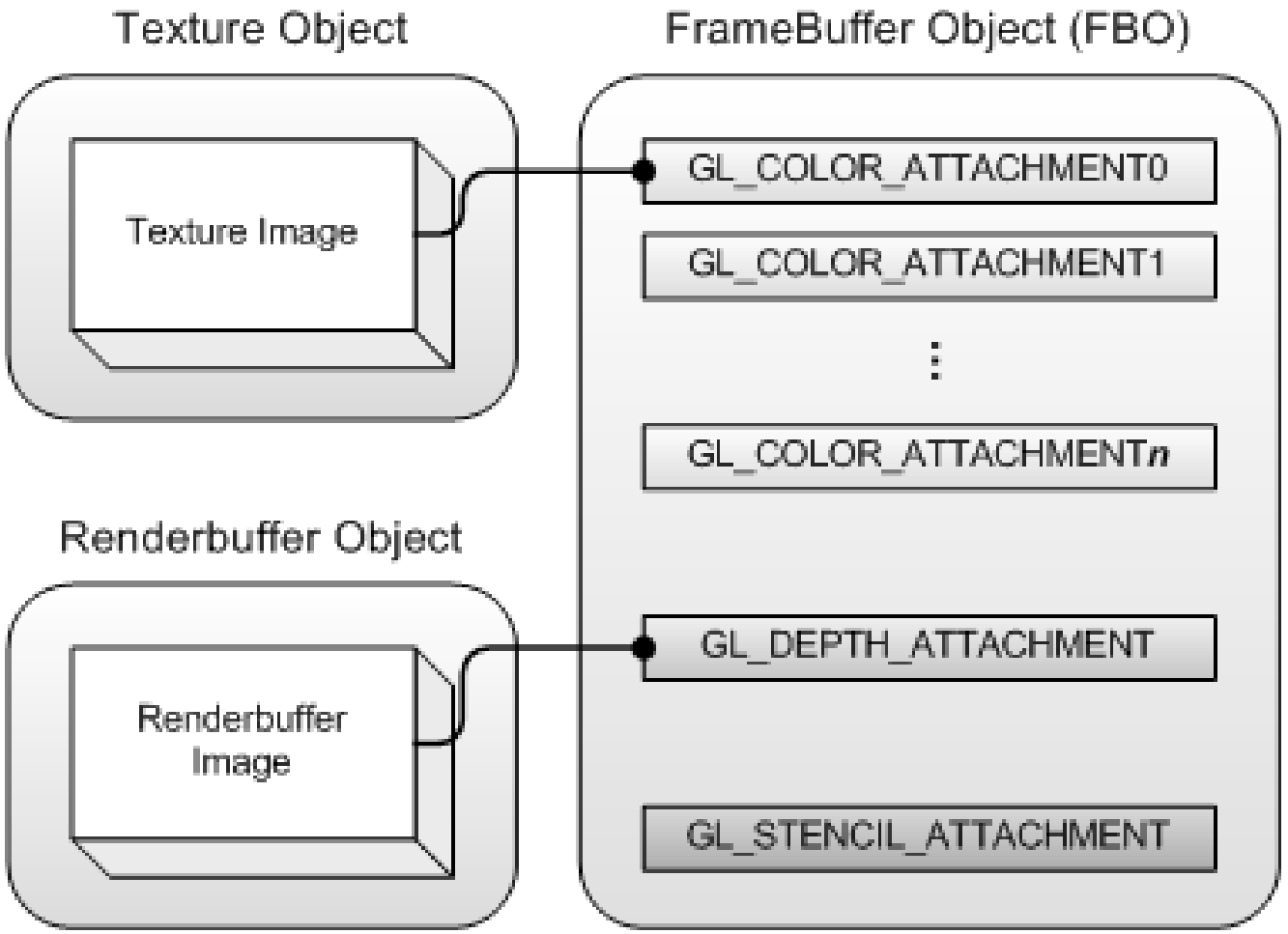
Shader de Vértices

```
uniform mat4 viewMatrix, projMatrix;
in vec3 position;
in vec3 texCoord0;
out vec3 texCoord;
void main (void)
{
   vec4 tmp;
   tmp = vec4(position, 1.0);
   gl_Position = projMatrix * viewMatrix * tmp;
   texCoord = texCoord0 ;
}
```

Shader de Fragmentos

```
t = 5.0;
do {
  geomPos = initPos + geomDir*t;
  texpos = geomPos * invScaleFactors;
  scalar = texture(VOLUME,
  texpos.xyz).r;
  if (scalar > limiar) {
    FragColor =
  vec4(geomPos.z,geomPos.z,geomPos.z,1.0) ;
   return;
  }
   t -= passo ;
 } while (t >= 0.0);
```

OpenGL: *Framebuffer Object*



Fonte: http://www.songho.ca/opengl/gl_fbo.htm

Alocação de FBO

```
glBindTexture(GL_TEXTURE_RECTANGLE, texID[2]);
glTexParameterf(GL_TEXTURE_RECTANGLE,
    GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameterf(GL_TEXTURE_RECTANGLE,
    GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexImage2D(GL_TEXTURE_RECTANGLE, 0, GL_RGB,
            VIEWPORT_WIDTH, VIEWPORT_HEIGHT, 0, GL_RGB,
            GL_FLOAT, 0);
glGenFramebuffers(1, &fbo);
glBindFramebuffer(GL_FRAMEBUFFER, fbo);
glFramebufferTexture2D(GL_FRAMEBUFFER,
    GL_COLOR_ATTACHMENT0,
                    GL_TEXTURE_RECTANGLE, texID[2], 0);
glBindFramebuffer(GL_FRAMEBUFFER, 0);
```

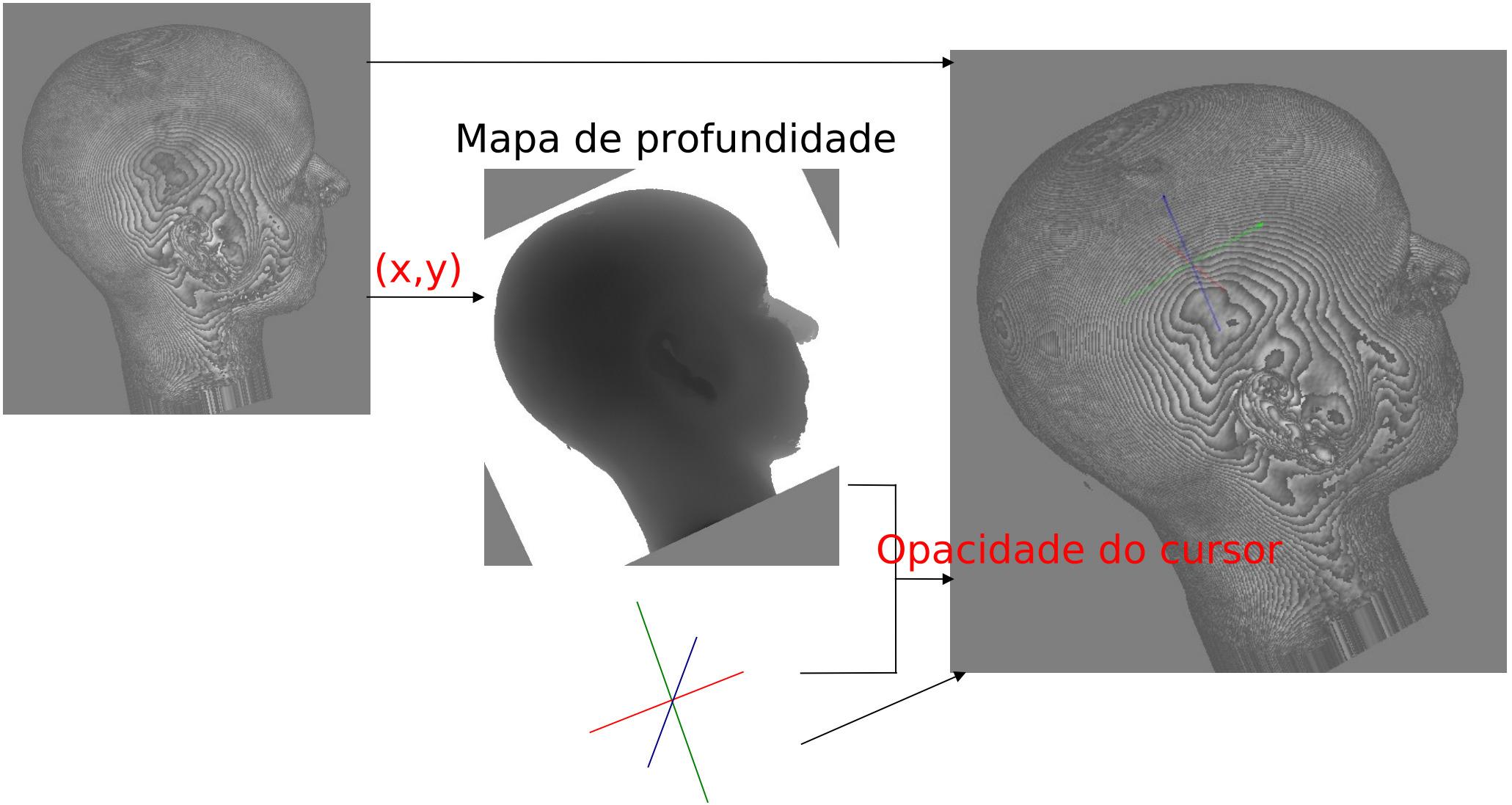
Redirecionamento do Fluxo

```
glBindFramebuffer(GL_FRAMEBUFFER, fbo);
GLenum buffers[1] = {GL_COLOR_ATTACHMENT0};

glDrawBuffers(1, buffers);

  :

glDrawElements(GL_QUADS, 24, GL_UNSIGNED_BYTE, (const
   GLvoid *)0);
```

Leitura de FBO

```
glReadBuffer(GL_COLOR_ATTACHMENT0);
glReadPixels(0, 0, VIEWPORT_WIDTH, VIEWPORT_HEIGHT,
    GL_RGB, GL_FLOAT,
            bufferTex2D);

glBindFramebuffer(GL_FRAMEBUFFER, 0);
```

Realimentação Visual



Percepção 3D

```
uniform sampler2DRect DEPTHMAP;
uniform vec4 iparam;
out vec4 FragColor;
void main(void) {
    float z;
    float alpha;
    z       = texture(DEPTHMAP, gl_FragCoord.xy).x;
    alpha = 1.0;
    if (gl_FragCoord.z > z)
       alpha = 1.0-(gl_FragCoord.z-z);
}
```