

Câmera e Viewing 3D

SCC0250/0650 - Computação Gráfica

Prof^a. Rosane Minghim

<https://edisciplinas.usp.br/course/view.php?id=61213>

<https://edisciplinas.usp.br/course/view.php?id=61210>

rminghim@icmc.usp.br

P.A.E. Diego Cintra e Fábio Felix

diegocintra@usp.br, f_diasfabio@usp.br

Instituto de Ciências Matemáticas e de Computação (ICMC)

Universidade de São Paulo (USP)

baseado no material de anos anteriores, vários autores

23 de abril de 2018



Sumário

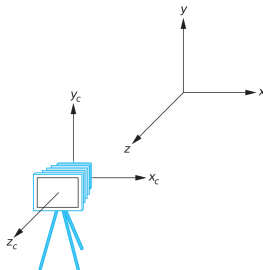
1 Câmera

2 Projeções

3 ViewPort

Câmera virtual

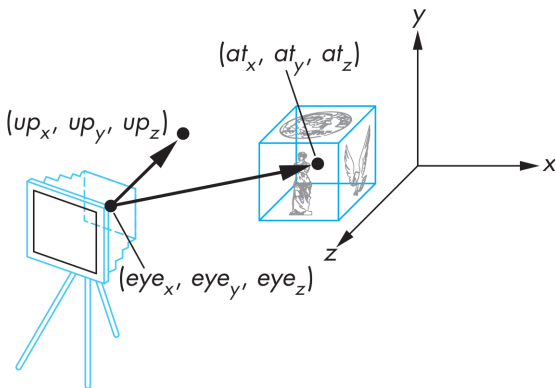
- A visualização de uma cena 3D é feita com o uso de uma “câmera”, ou uma coordenada referência para essa abstração.
- Um ponto definido como *View Reference Point* define a posição do plano de vista (ou plano de projeção) da cena, que de fato é a abstração do filme da câmera.
- Esse plano define a porção visível da cena, contida no **volume de vista** ou **volume de visão**.



Propriedades de uma câmera virtual

- O ponto que define a posição inicial da câmera é o *View Reference Point*.
- Deste ponto, define-se um plano de projeção, ou *View Plane*.
- Estabelecem-se versores para o *View Reference Point*, denominados $\vec{u}$, $\vec{v}$ e $\vec{n}$, que definem o *View Reference Coordinates* (VRC).
 - O vetor $\vec{n}$ caracteriza a normal do plano de projeção, definindo a direção positiva de z .
 - O vetor $\vec{v}$ define o *View Up Vector* (VUP), que especifica a direção positiva de y e a direção da vista da cena.
 - O vetor $\vec{u}$ é o resultado do produto vetorial entre o VUP e o vetor normal do plano, que define a direção positiva de x .
- A largura e altura da janela, contida no plano de projeção, são dadas por parâmetros W e H .

Câmera - OpenGL



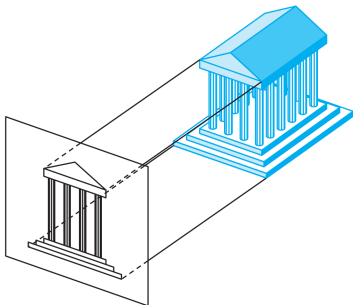
```
1 void gluLookAt(GLdouble eyeX, GLdouble eyeY, GLdouble eyeZ, GLdouble centerX, ↵  
    GLdouble centerY, GLdouble centerZ, GLdouble upX, GLdouble upY, GLdouble ↵  
    upZ);
```

Projeções

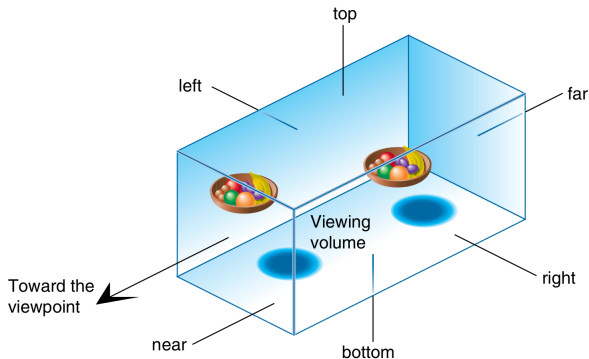
- Obtém representações bidimensionais de objetos tridimensionais.
- A projeção é definida por raios de projeção (projetores) que passam através de cada vértice dos objetos e interceptam o plano de projeção.
- Uma projeção pode ser ou paralela ortográfica ou perspectiva.
- A matriz de projeção é resultante de duas etapas: em um primeiro momento, o volume de vista é normalizado, obtendo-se um volume de vista **canônico**. A segunda etapa consiste na projeção das coordenadas de vista no plano de projeção. Portanto, essa matriz é obtida de maneira distinta, dependendo do tipo de projeção adotada.

Projeção paralela ou ortogonal

Projeta os pontos de um objeto ao longo de linhas paralelas. Essa projeção geralmente é utilizada em desenhos arquitetônicos e de engenharia para representação de um objeto mantendo suas propriedades relativas.



Projeção paralela ou ortogonal - OpenGL



1

```
void glOrtho(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble nearVal, GLdouble farVal);
```

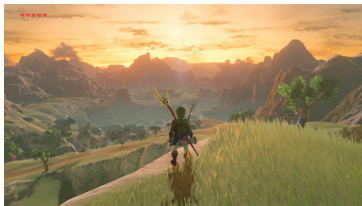
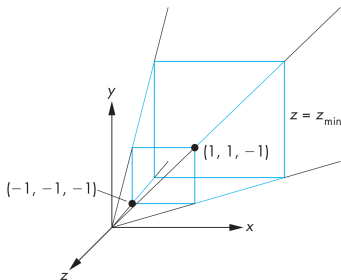

Projeção paralela ou ortogonal - matriz

$$M_{orthon} = \begin{bmatrix} \frac{2}{xw_{max}-xw_{min}} & 0 & 0 & -\frac{xw_{max}+xw_{min}}{xw_{max}-xw_{min}} \\ 0 & \frac{2}{yw_{max}-yw_{min}} & 0 & -\frac{yw_{max}+yw_{min}}{yw_{max}-yw_{min}} \\ 0 & 0 & -\frac{2}{d_{near}-d_{far}} & \frac{d_{near}+d_{far}}{d_{near}-d_{far}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

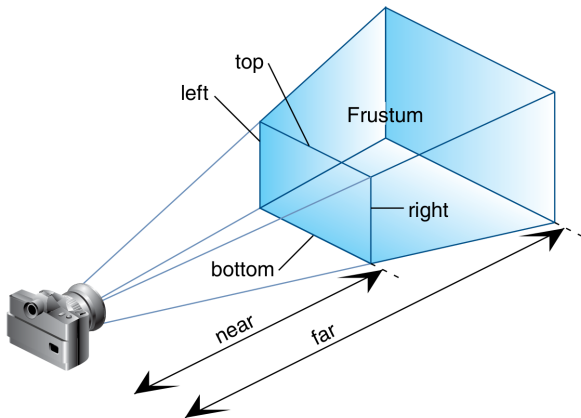
Essa é uma matriz genérica e não é necessariamente a matriz utilizada pela OpenGL. Verificar a documentação da função *glOrtho*.

Projeção perspectiva

Projeta os pontos de um objeto ao longo de caminhos convergentes. O volume de vista formado por essa projeção é um tronco de uma pirâmide, sem o topo, denominado **frustum**. Isso faz com que objetos distantes do plano de projeção fiquem menores do que objetos mais próximos do plano, causando maior realismo na cena.



Projeção perspectiva - OpenGL



1

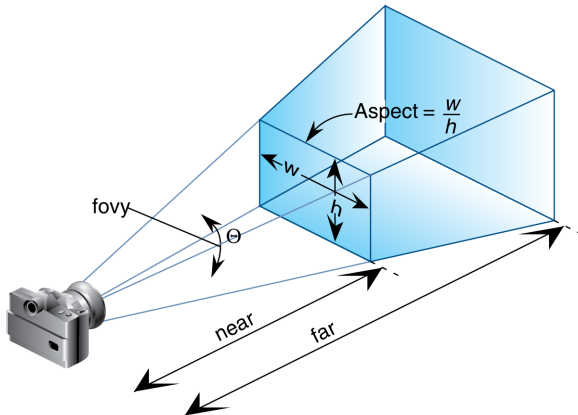
```
void glFrustum(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble nearVal, GLdouble farVal);
```

Matriz de projeção

$$M_{\text{pern}} = \begin{bmatrix} \frac{-2d_{\text{near}}}{xw_{\text{max}} - xw_{\text{min}}} & 0 & \frac{xw_{\text{max}} + xw_{\text{min}}}{xw_{\text{max}} - xw_{\text{min}}} & 0 \\ 0 & \frac{-2d_{\text{near}}}{yw_{\text{max}} - yw_{\text{min}}} & \frac{yw_{\text{max}} + yw_{\text{min}}}{yw_{\text{max}} - yw_{\text{min}}} & 0 \\ 0 & 0 & \frac{d_{\text{near}} + d_{\text{far}}}{d_{\text{near}} - d_{\text{far}}} & -\frac{2d_{\text{near}}d_{\text{far}}}{d_{\text{near}} - d_{\text{far}}} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Essa é uma matriz genérica e não é necessariamente a matriz utilizada pela OpenGL. Verificar a documentação da função *glFrustum*.

Projeção perspectiva simétrica - OpenGL



1

```
void gluPerspective(GLdouble fovy, GLdouble aspect, GLdouble zNear, GLdouble zFar);
```

Matriz de projeção

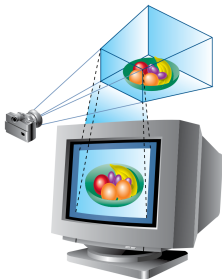
$$M_{persn} = \begin{bmatrix} \frac{\cot(\frac{\theta}{2})}{aspect} & 0 & 0 & 0 \\ 0 & \cot(\frac{\theta}{2}) & 0 & 0 \\ 0 & 0 & \frac{d_{near}+d_{far}}{d_{near}-d_{far}} & -\frac{2d_{near}d_{far}}{d_{near}-d_{far}} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Teste de profundidade

- Informações de profundidade são importantes para bibliotecas gráficas, pois dependendo da projeção, pode haver ambiguidade na visualização de certos objetos. Algumas soluções adotadas por sistemas gráficos incluem:
 - Destacar as arestas ou linhas visíveis de um objeto com outra cor;
 - Definir as linhas ou arestas não visíveis com linhas tracejadas;
 - Remover as arestas ou linhas não visíveis.



Viewport e coordenadas de tela



Após o conteúdo do volume de visão ter sido definido, esse pode ser transferido para coordenadas da tela.

Esse é um processo semelhante ao 2D, porém informação de profundidade é preservada para teste de visibilidade e rendering.

- A variável z é normalizada entre 0 e 1, sendo que $z = 0$ temos a altura da tela

Viewport e coordenadas de tela

Nas coordenadas normalizadas, $z_{norm} = -1$ é mapeada para a coordenada na tela $z_{screen} = 0$

- *Viewport* é definida como $(xv_{min}, yv_{min}, 0)$ e $(xv_{max}, yv_{max}, 1)$

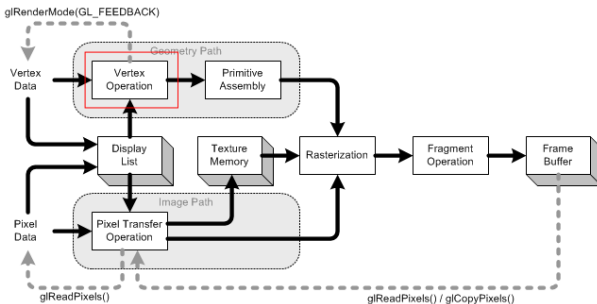
As posições x e y são enviadas para o **frame buffer** (informação de cor para os pontos na tela)

Os valores de z são enviados para o **depth buffer** para serem usados em rotinas para determinação de cor e visibilidade.

$$M_{viewport} = \begin{bmatrix} \frac{xv_{max} - xv_{min}}{2} & 0 & 0 & \frac{xv_{max} + xv_{min}}{2} \\ 0 & \frac{yv_{max} - yv_{min}}{2} & 0 & \frac{yv_{max} + yv_{min}}{2} \\ 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Viewing 3D

- Viewing 3D: Etapa de geração de coordenadas de recorte para cada vértice da cena.
- Realizada na etapa de processamento de vértices do pipeline da OpenGL.



Observações

Para trabalhar com 3D, a depender das necessidades, habilitar a detecção de faces visíveis dos objetos.

```

1 //Dizer para a OpenGL utilizar o buffer de profundidade (GLUT_DEPTH)
2 glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
3
4 ...
5
6 //Habilitar o teste de profundidade
7 glEnable(GL_DEPTH_TEST);
8
9 //Informar qual tipo de teste deve ser realizado:
10 //GL_ALWAYS (todos os pontos são processados), GL_NEVER (nenhum ponto é processado),
11 //GL_LESS (default), GL_EQUAL, GL_LEQUAL, GL_GREATER, GL_NOTEQUAL, ↔
   GL_GEQUAL
12 glDepthFunc(GL_LESS);
13
14 //Se necessário habilitar a remoção de faces
15 glEnable(GL_CULL_FACE);
16
17 //E informar qual face deve ser removida: GL_BACK (default), GL_FRONT, ↔
   GL_FRONT_AND_BACK
18 glCullFace(GL_BACK);
19
20 ...
21
22 //Inicializar os valores do buffer de profundidade
23 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

```

Bibliografia

- **Básica:**

- Hearn, D. Baker, M. P. Computer Graphics with OpenGL, Prentice Hall, 2004. **(livro texto)**
- Neider, J. Davis, T. Woo, M. OpenGL programming guide, 2007.
- Angel, E. Interactive computer graphics: a top-down approach with OpenGL, Addison Wesley, 2000.
- Foley, J. et. al. Introduction to Computer Graphics, Addison-Wesley, 1993.
- Angle, E. and Shreiner, D., 2011. Interactive computer graphics: A topdown approach with shader-based opengl.

Bibliografia

- **Complementar:**

- Computer Graphics Comes of Age: An Interview with Andries van Dam. CACM, vol. 27, no. 7. 1982
- The RenderMan – And the Oscar Goes to... IEEE Spectrum, vol. 38, no. 4, abril de 2001.
- Material do ano passado:
<https://sites.google.com/site/computacaograficaicmc2017t2/>
- Apostilas antigas da disciplina Computação Gráfica
 - <http://www.gbdi.icmc.usp.br/material?q=system/files/apostilas.pdf>
- Curso da ACM SIGGRAPH (on line)