Atividade de laboratório 2

P.A.E. Diego Cintra e Fábio Felix
diegocintra@usp.br, f_diasfabio@usp.br

03 de maio de 2018

As atividades descritas a seguir devem seguir as seguintes restrições:

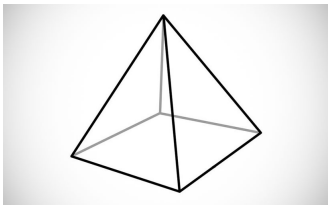
- Todas devem ser implementadas **individualmente**;
- As atividades devem ser implementadas utilizando a API da OpenGL, sendo as bibliotecas `gl`, `glu`, `glut` e `glew` as únicas que podem ser utilizadas.
- As linguagens permitidas são **C** e **C++**.
- Para submissão, aqueles que optarem por utilizar Windows devem compactar todo o código-fonte como um arquivo ".zip", incluindo executável. Os que optarem por sistemas operacionais baseados em UNIX também devem enviar todo o código-fonte compactado, acompanhado de um `Makefile`.

Atividade 1

Para desenvolver as questões dessa atividade utilize como base os códigos gerados na última aula de laboratório.

Questões propostas

1. Modifique o *menu* da **Atividade 1** da aula anterior, para que ele possa apresentar apenas opções para desenhar **cubo** e **pirâmide**. Ao escolher uma dessas opções o objeto deve ser desenhado no centro da tela com um tamanho fixo. Desenhe a pirâmide com uma base quadrada como na figura abaixo.



2. Estenda a funções de transformações geométricas para trabalhar com dados em 3D. Lembre-se de implementar a translação, escala e rotação com operações matriciais, mesmo que não tenha implementado dessa maneira na primeira atividade. Para rotação, **não utilize** uma matriz para cada direção, isso é mais complicado e demanda mais operações matriciais. Utilize o conceito de *Quaternion*, aplicando a matriz apresentada a seguir para efetuar a rotação dos pontos.

$$M_R = \begin{bmatrix} x^2(1-c)+c & xy(1-c)-zs & xz(1-c)+ys & 0 \\ yx(1-c)+zs & y^2(1-c)+c & yz(1-c)-xs & 0 \\ zx(1-c)-ys & zy(1-c)+xs & z^2(1-c)+c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

sendo $c = cos(\theta)$, $s = sen(\theta)$ e (x, y, z) as coordenadas do vetor unitário $\vec{u}$ em torno do qual a rotação deve acontecer.

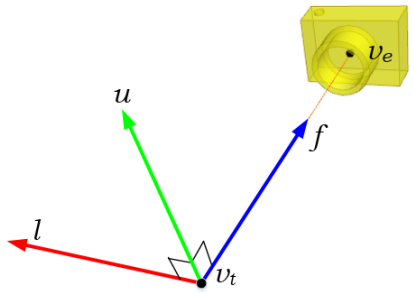
obs.: No lugar de rotacionar com as teclas direcionais, utilize as letras x, y e z para rotacionar em torno dos respectivos eixos. Os sentidos da rotação podem ser modificados utilizando, por exemplo, x para o sentido horário e X para o anti-horário. As transformações de escala e translação devem continuar utilizando os mesmos atalhos ($+$ e $-$ para escala e o clique do *mouse* para translação).

Atividade 2

Como foi apresentado em sala de aula, a OpenGL não possui uma câmera propriamente dita. Na verdade, um conjunto de transformações geométricas globais são aplicadas para emular a ideia de câmera. Essas operações são efetuadas através da função *gluLookAt*.

Questões propostas

1. Implemente sua própria função *LookAt* que receba as coordenadas dos pontos 3D $\vec{v_e}$ (posição da câmera), $\vec{v_t}$ (direção para onde a câmera está apontada) e do vetor 3D $\vec{up}$ (orientação da câmera no eixo y). A próxima figura descreve tanto alguns desses itens quanto os vetores utilizados para cálculo das matrizes de transformação.



Sua função deve efetuar uma translação e uma rotação sobre os pontos a partir dos vetores especificados acima. As matrizes de translação T e de rotação R devem ser geradas com as informações dos vetores passados para a função da forma especificada abaixo.

$$\vec{f} = \frac{\vec{v_e} - \vec{v_t}}{\|\vec{v_e} - \vec{v_t}\|} \qquad \vec{l} = \frac{\vec{up} \times \vec{f}}{\left\|\vec{up} \times \vec{f}\right\|} \qquad \vec{u} = \vec{f} \times \vec{l}$$

$$T = \begin{bmatrix} 1 & 0 & 0 & -x_{ve} \\ 0 & 1 & 0 & -y_{ve} \\ 0 & 0 & 1 & -z_{ve} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad R = \begin{bmatrix} x_l & y_l & z_l & 0 \\ x_u & y_u & z_u & 0 \\ x_f & y_f & z_f & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

obs.: Desenhe algum objeto 3D na tela e efetue testes modificando os parâmetros da sua função para verificar o seu funcionamento. Para ter certeza se a sua função está funcionando corretamente, compare os resultados dela com os do *gluLookAt*.