

Computação Gráfica – Animação

Profa. Mercedes Gonzales
Márquez

Animação

Do latim Animare: dar vida, movimento, coragem, entusiasmo, alma.

Os passos para a produção de uma animação em CG são basicamente:

- Desenhar ou esculpir
- determinar os movimentos
- Retratar o espírito da criatura ou cena a ser animada.

Animação - Aplicações

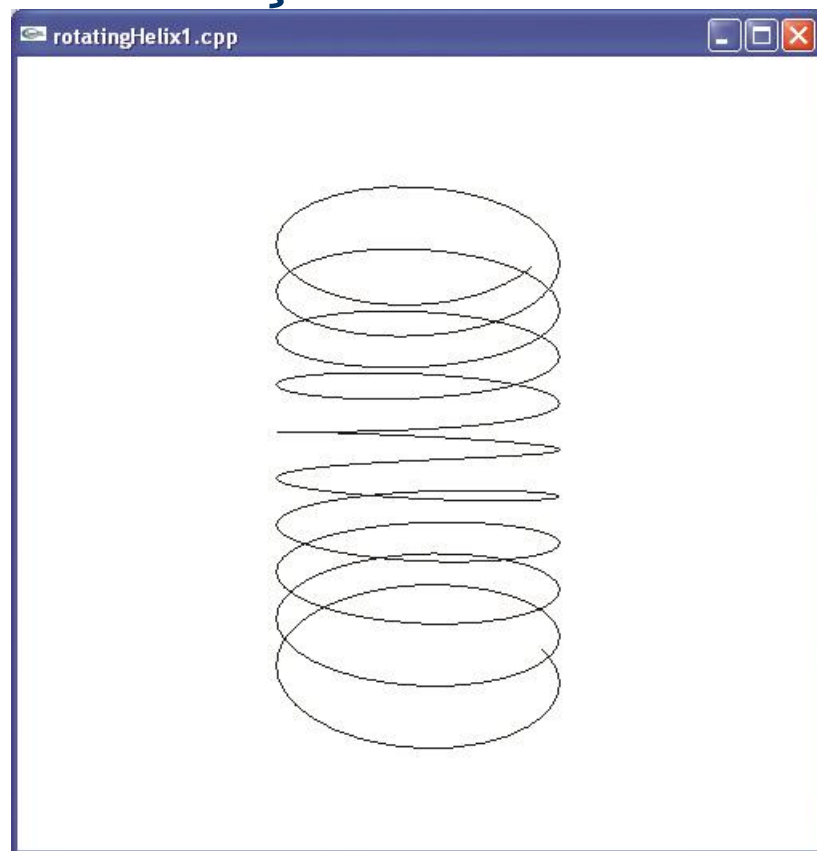
- Mídias: Filmes e propagandas.
- Engenharia: Testes de resistência e impactos.
- Medicina: Entender os movimentos e órgãos do corpo humano.
- Outros.

Animação - OpenGL

● Três métodos de Controle da Animação

1. Interativamente:

- via entrada por mouse ou teclado, com a ajuda das correspondentes funções callback para invocar as transformações.
- Exemplo: programa `rotatingHelix1.cpp` onde cada apertado da tecla espaço chama a rotina `increaseAngle()` para girar a hélice. O comando `glutPostRedisplay()` em `increaseAngle()` solicita que a tela seja redeseenhada.



Animação - OpenGL

- Três métodos de Controle da Animação

2. Automaticamente:

- especificando a função idle com o comando `glutIdleFunc(idle_function)`.
- A função `idle_function` é chamada sempre que nenhum evento em OpenGL está pendente.
- Exemplo programa `rotatingHelix2.cpp`, uma ligeira modificação de `rotatingHelix1.cpp`, onde apertando a tecla espaço faz com que as rotinas `increaseAngle()` e `NULL` (fazer nada) sejam alternadamente especificadas como funções idle.

Animação - OpenGL

- Três métodos de Controle da Animação

- 3. **Automaticamente:**

- especificando a rotina `timer_function`, com a chamada a `glutTimerFunc(period, timer_function, value)`.
 - A função `timer_function` é chamada `period` milisegundos depois que o comando `glutTimerFunc` é executado e com o parâmetro `value` sendo passado.
 - Exemplo: No programa `rotatingHelix3.cpp`, a `timer-function` `animate()` é primeiro chamada desde a função principal `main`, 5 miliseg. depois que o comando `glutTimerFunc(5,animate,1)` é executado. O parâmetro `value 1` que é passado para `animate()` não é usado neste programa. Chamadas subsequentes a `animate()` são feitas recursivamente desde a mesma rotina `glutTimerFunc`.

Animação - OpenGL

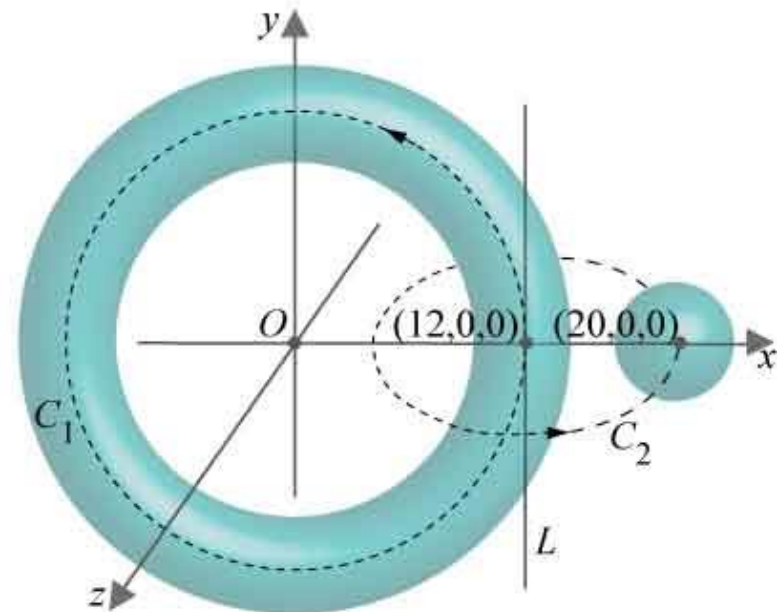
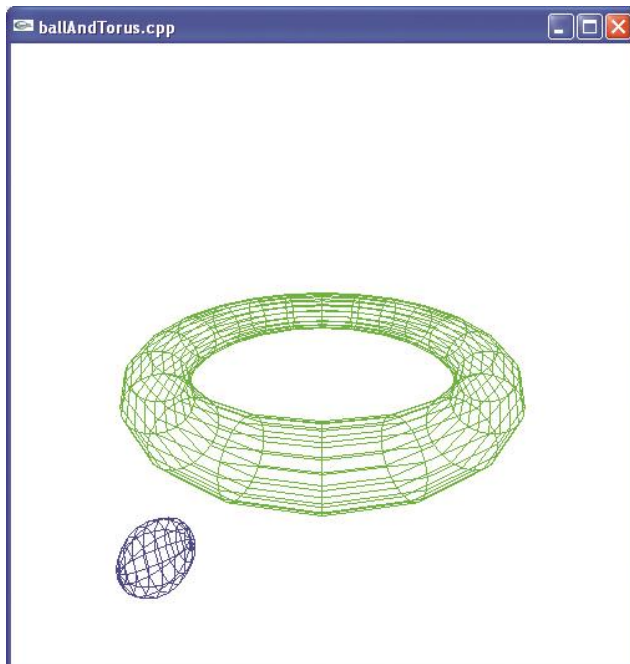
- Buffer duplo

- Buffer duplo grandemente melhora a qualidade da animação ocultando, ao observador, a transição entre os sucessivos frames. Pelo outro lado, com um único buffer, o observador “ve” o próximo frame sendo desenhado no mesmo buffer que contém o atual. O resultado pode ser indesejável *ghosting*, já que a primeira imagem persiste enquanto a próxima imagem está sendo criada.
- O modo de display buffer duplo é habilitado chamando `glutInitDisplayMode()` em `main()` com `GLUT_DOUBLE` (em lugar de `GLUT_SINGLE`) e inserindo a chamada a `glutSwapBuffers()` no final da rotina de desenho (no lugar de `glFlush()`).
- Experimento: Desabilite o buffer duplo em `rotatingHelix2.cpp`.

Animação - OpenGL

- Exemplos de animação

- Rode o programa ballAndTorus.cpp. Aperte a tecla espaço para iniciar o rotação latitudinal e longitudinal de uma bola ao redor do torus.



Animação - OpenGL

- Exercício: Rode o programa throwBall.cpp o qual simula o movimento de uma bola jogada com uma velocidade inicial específica e sujeita à força da gravidade.

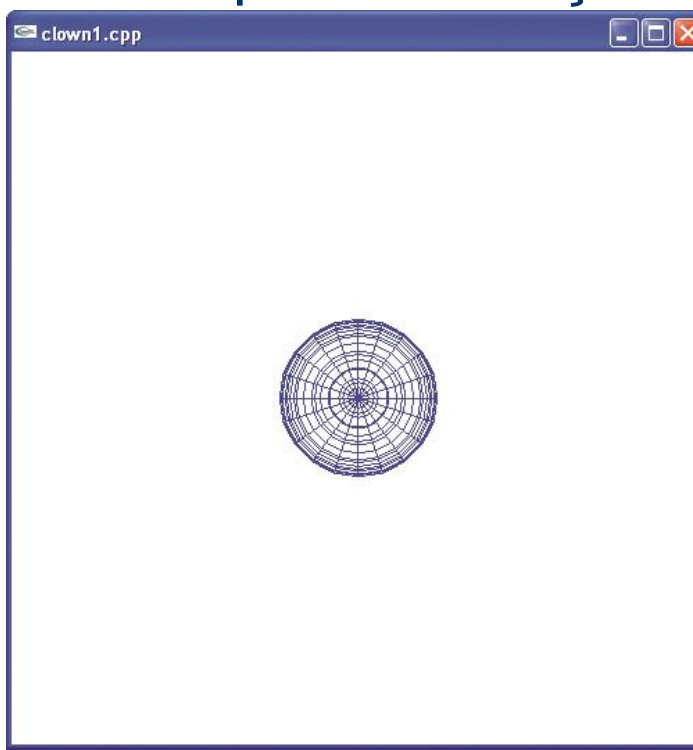
Considere:

(a) a equação que determina o movimento horizontal da bola em termos de t : $x(t)=ht$. Onde h é a componente horizontal da velocidade inicial.

b) a equação que determina o movimento vertical, $y(t)=vt-(g/2)t^2$, onde v é a componente vertical da velocidade inicial e g é a aceleração gravitacional.

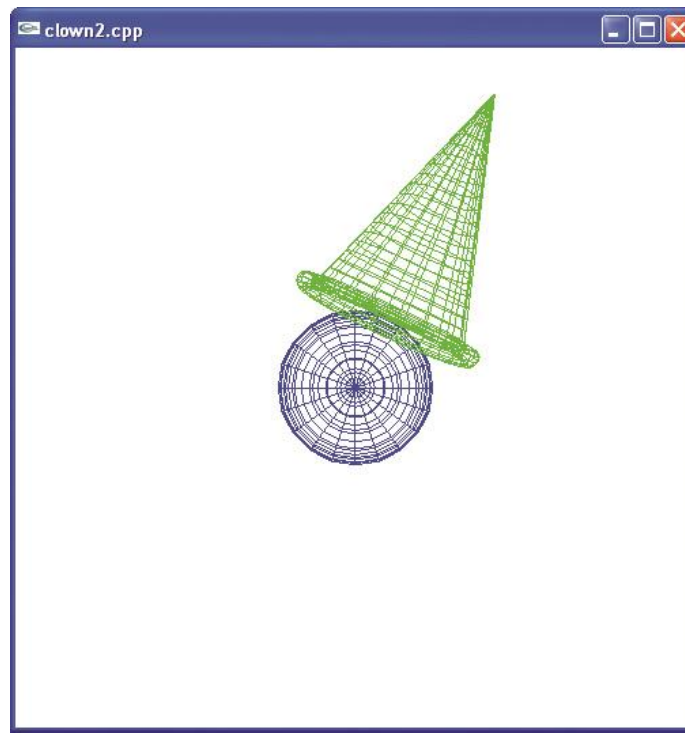
Animação - OpenGL

- Veja os passos para realizar a animação da cabeça de um palhaço. (a) Iniciamos com o desenho de uma esfera azul para representar de forma simples a cabeça do palhaço, programa `clown1.cpp`.



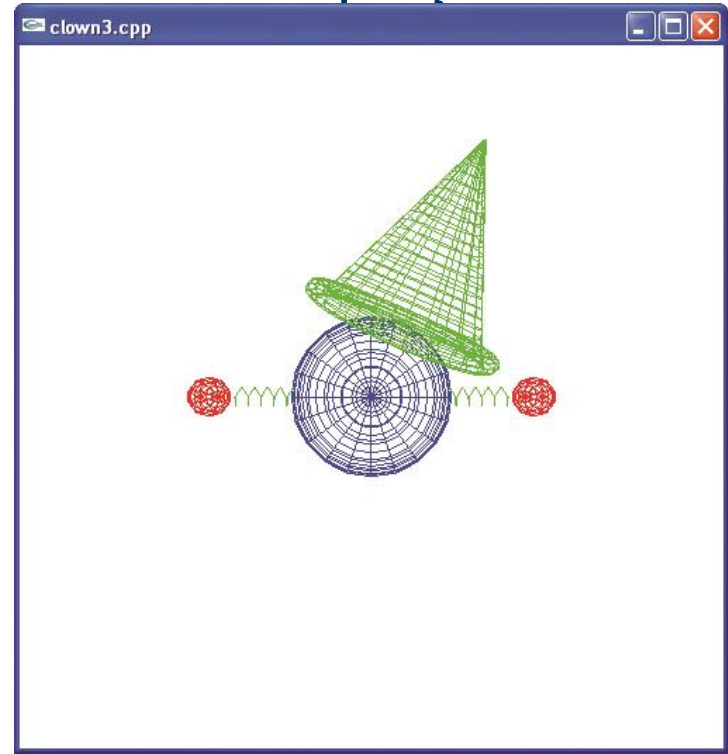
Animação – OpenGL

- Depois em clown2.cpp, desenha-se um chapéu de formato de cone com uma borda como base e gira-se esse chapéu na cabeça do palhaço. Veja Figura (b).



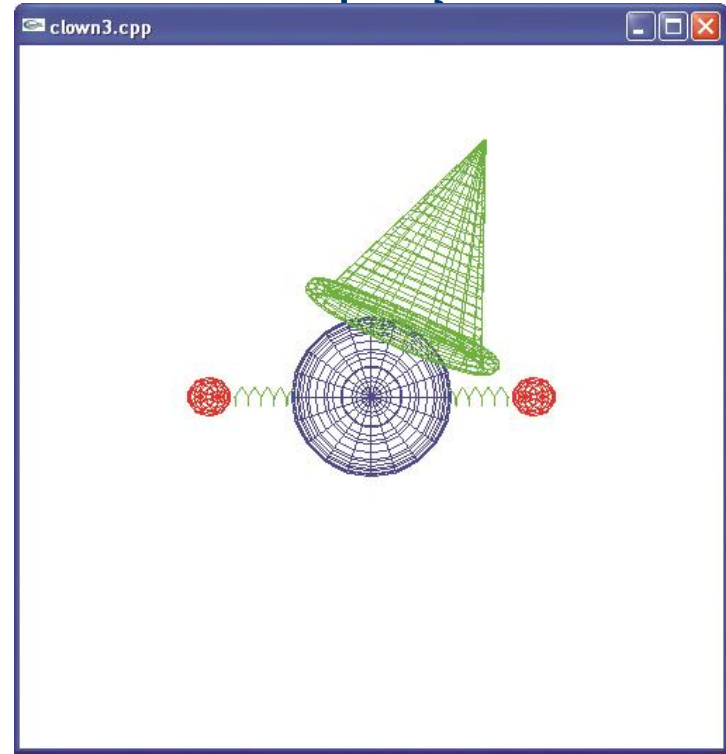
Animação - OpenGL

- Finalmente completa-se a animação, desenhando duas orelhas, as quais realizam uma animação ligadas à cabeça através de duas molas. Para lembrar da equação da hélice, veja o programa `helix.cpp`.



Animação - OpenGL

- Finalmente completa-se a animação, desenhando duas orelhas, as quais realizam uma animação ligadas à cabeça através de duas molas. Para lembrar da equação da hélice, veja o programa `helix.cpp`.



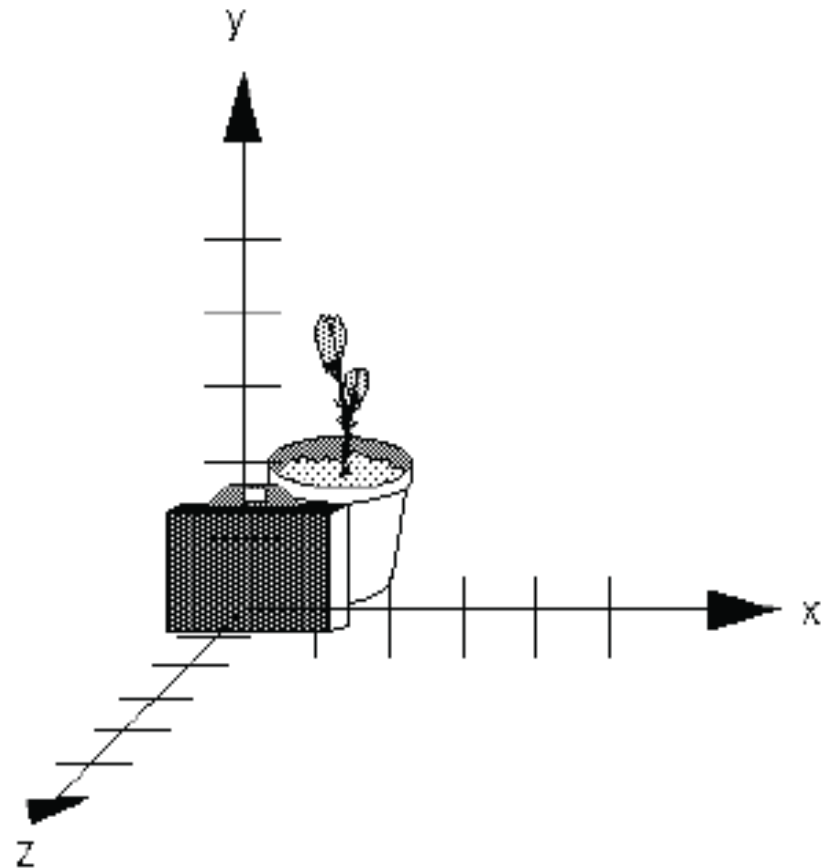
Orientar a câmera em direção da cena (transformação de visualização)

A câmera em OpenGL “por default” tem sua posição na origem de coordenadas (0,0,0) e a sua orientação é com vetor $up=(0,1,0)$. Existem duas opções para mudar sua posição e orientação:

- (1) Usar `glTranslate*()` e `glRotate*()`. Move a camera ou move todos os objetos em relação a uma camera fixa;
- (2) `gluLookAt()`

Visualizando devidamente o objeto (Exemplo)

- Objeto e câmera na origem

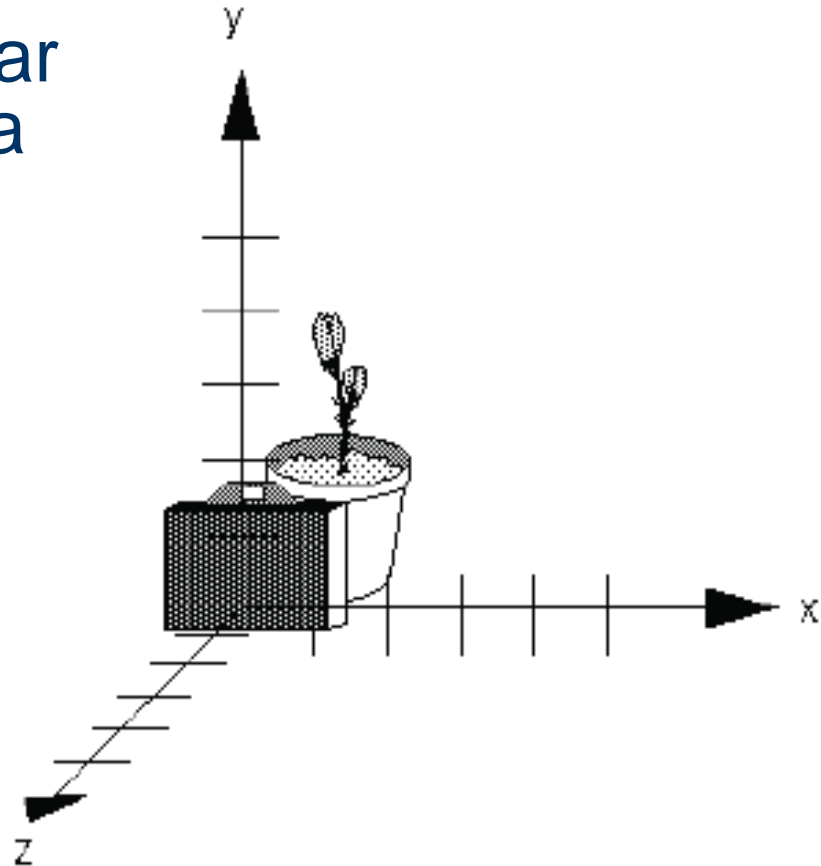


Visualizando devidamente o objeto

Com a câmera na origem $(0,0,0)$ não posso visualizar devidamente um objeto na posição $(0,0,0)$

Para visualizá-lo tenho duas opções:

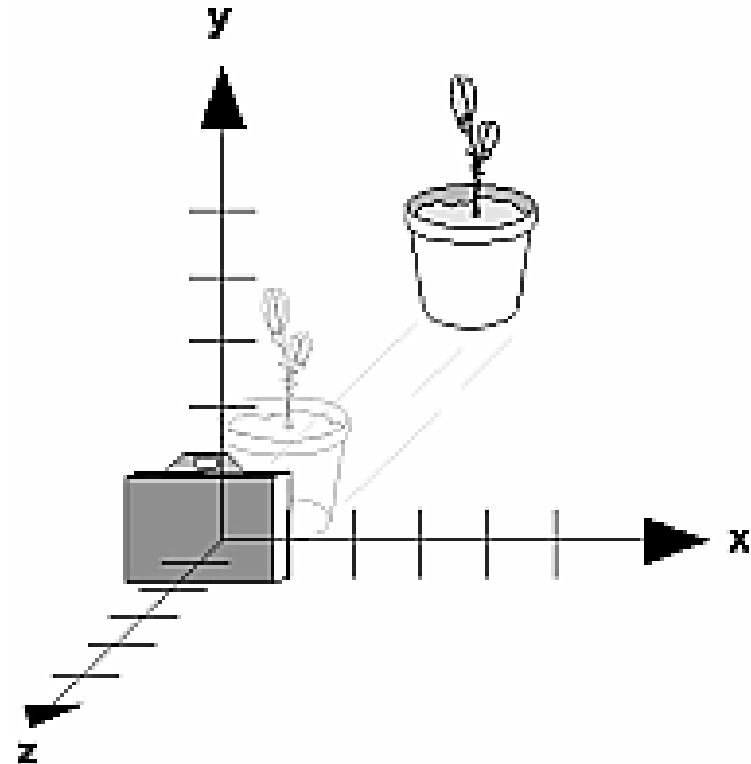
- (a) Mudar a câmera, ou
- (b) Mudar o objeto



Usando `glTranslate()` e `glRotate()`

(b) Mudando o objeto

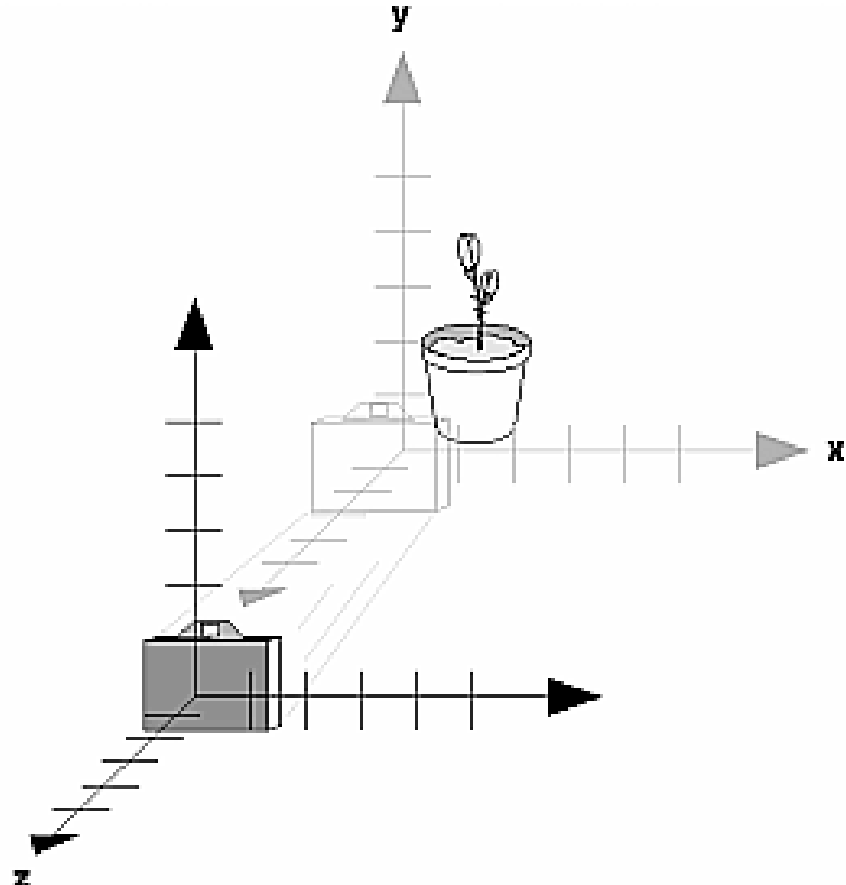
```
glTranslatef(0.0, 0.0, -5.0);
```



Usando gluLookAt

(a) Mudando a câmera

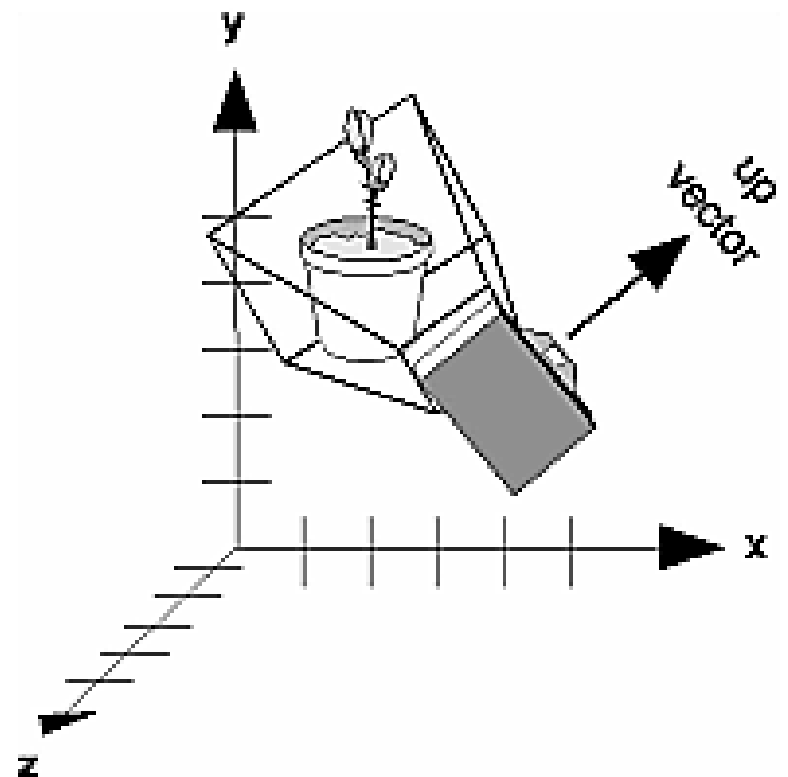
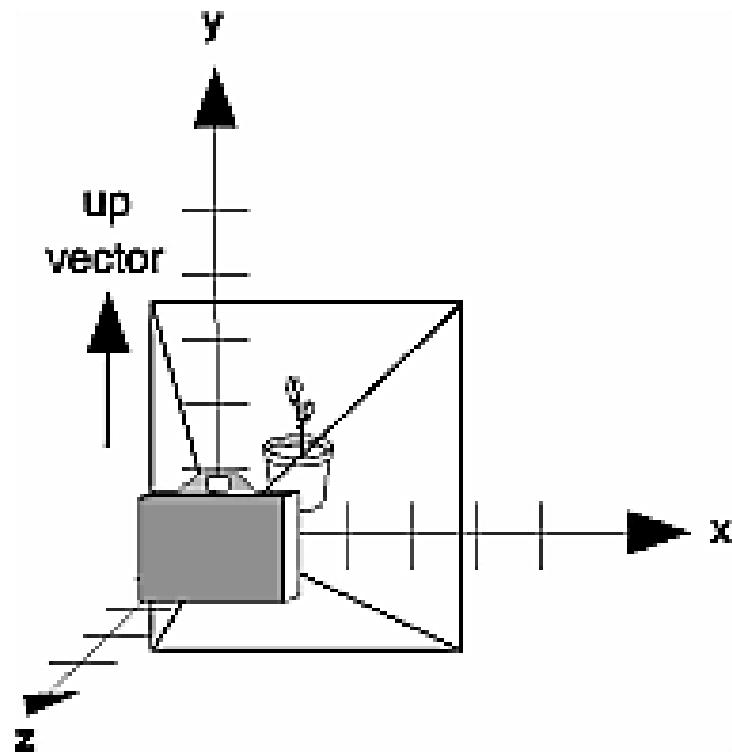
```
gluLookAt (eyex, eyey, eyez  
          centerx, centery, centerz,  
          upx, upy, upz)
```



gluLookAt

- A cena é construída na origem e definimos uma posição arbitrária para a câmera
- `void gluLookAt (eyex, eyey, eyez, centerx, centery, centerz, upx, upy, upz);`
 - Eye: localização da camera
 - Center: para onde a camera aponta
 - Up: vetor de direção de topo da camera

gluLookAt



Exemplo – Cubo (Programa cube.c)

Um cubo é escalado pela transformação de modelagem `glScalef (1.0, 2.0, 1.0)`. A transformação de visualização `gluLookAt()`, posiciona e orienta a câmera em direção do cubo. As transformações de projeção e viewport são também especificadas.

