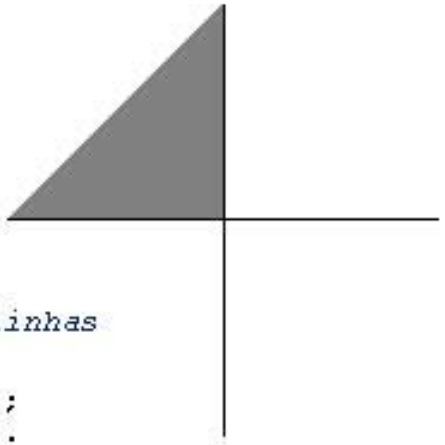
Aula 06

Desenhos Bidimensionais

- Utilizando a primitiva gráfica GL_TRIANGLE_FAN
- Detalhando o código:
  - Desenhando uma seqüência de triângulos
  - Desenhando uma seqüência de linhas

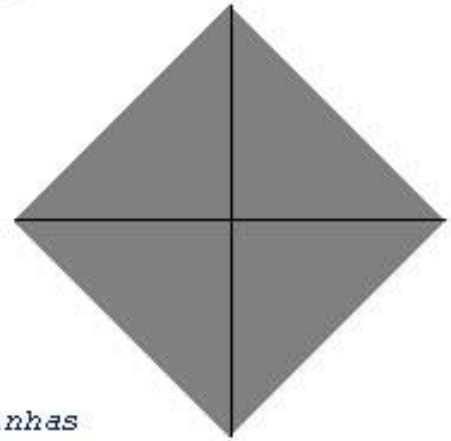
```
// Desenha uma seqüencia de triângulos
glBegin(GL_TRIANGLE_FAN);
    glVertex2f(  0.0f,  0.0f);
    glVertex2f(-20.0f,  0.0f);
    glVertex2f(  0.0f, 20.0f);
    //glVertex2f( 20.0f,  0.0f);
    //glVertex2f(  0.0f,-20.0f);
    //glVertex2f(-20.0f,  0.0f);
glEnd();
```



```
// Desenha uma seqüencia de linhas
glBegin(GL_LINES);
    glVertex2f(  0.0f,  0.0f);
    glVertex2f(-20.0f,  0.0f);
    glVertex2f(  0.0f,  0.0f);
    glVertex2f(  0.0f, 20.0f);
    glVertex2f(  0.0f,  0.0f);
    glVertex2f( 20.0f,  0.0f);
    glVertex2f(  0.0f,  0.0f);
    glVertex2f(  0.0f,-20.0f);
glEnd();
```

Desenhos Bidimensionais

- Utilizando a primitiva gráfica GL_TRIANGLE_FAN
- Detalhando o código:
  - Desenhando uma seqüência de triângulos
  - Desenhando uma seqüência de linhas



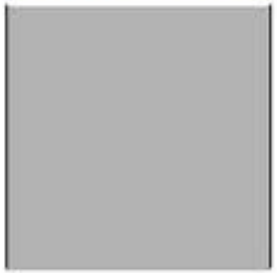
```
// Desenha uma seqüencia de triângulos
glBegin(GL_TRIANGLE_FAN);
    glVertex2f(  0.0f,  0.0f);
    glVertex2f(-20.0f,  0.0f);
    glVertex2f(  0.0f, 20.0f);
    glVertex2f( 20.0f,  0.0f);
    glVertex2f(  0.0f,-20.0f);
    glVertex2f(-20.0f,  0.0f);
glEnd();
```

```
// Desenha uma seqüencia de linhas
glBegin(GL_LINES);
    glVertex2f(  0.0f,  0.0f);
    glVertex2f(-20.0f,  0.0f);
    glVertex2f(  0.0f,  0.0f);
    glVertex2f(  0.0f, 20.0f);
    glVertex2f(  0.0f,  0.0f);
    glVertex2f( 20.0f,  0.0f);
    glVertex2f(  0.0f,  0.0f);
    glVertex2f(  0.0f,-20.0f);
glEnd();
```

Desenhos Bidimensionais

- Utilizando a primitiva gráfica GL_QUAD_STRIP
- Detalhando o código:
  - Desenhando uma seqüência de quadrados
  - Desenhando uma seqüência de linhas

```
// Desenha uma seqüência de quadrados
glBegin(GL_QUAD_STRIP);
    glVertex2f(-45.0f, 15.0f);
    glVertex2f(-45.0f,-15.0f);
    glVertex2f(-15.0f, 15.0f);
    glVertex2f(-15.0f,-15.0f);
    //glVertex2f( 15.0f, 15.0f);
    //glVertex2f( 15.0f,-15.0f);
    //glVertex2f( 45.0f, 15.0f);
    //glVertex2f( 45.0f,-15.0f);
glEnd();
```

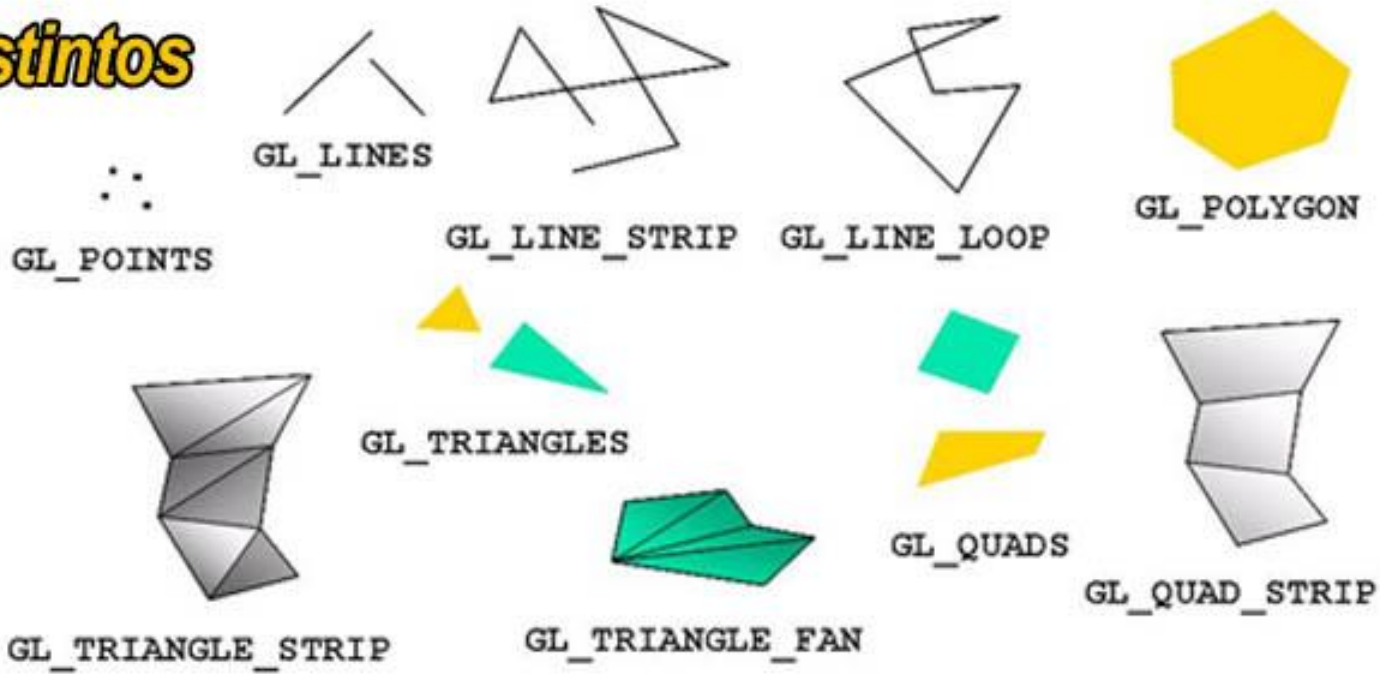


```
// Desenha linhas
glBegin(GL_LINES);
    glVertex2f(-45.0f, 15.0f);
    glVertex2f(-45.0f,-15.0f);
    glVertex2f(-15.0f, 15.0f);
    glVertex2f(-15.0f,-15.0f);
    glVertex2f( 15.0f, 15.0f);
    glVertex2f( 15.0f,-15.0f);
    glVertex2f( 45.0f, 15.0f);
    glVertex2f( 45.0f,-15.0f);
glEnd();
```

Desenhos Bidimensionais

- Cores e Estilos de Traçado de Primitivas:

- Desenho de primitivas com diversas aparências:
  - Cores
  - Estilos
  - Padrões distintos



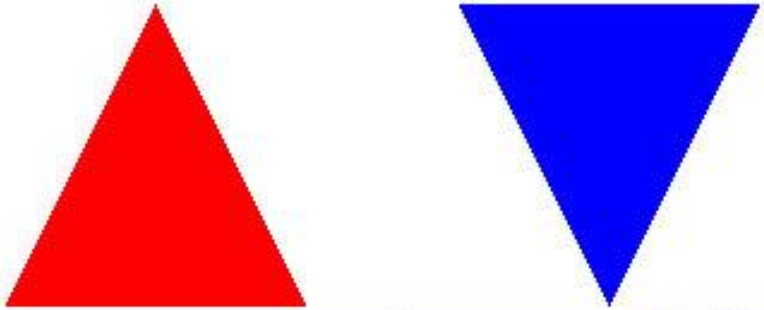
Desenhos Bidimensionais

- Cores e Estilos de Traçado de Primitivas:

- Alterando a cor de um desenho bidimensional:
  - Função: glColor3f
  - Detalhando o código:
    - Alterando a cor do desenho para vermelho
    - Alterando a cor do desenho para azul

glColor3f(r,g,b)

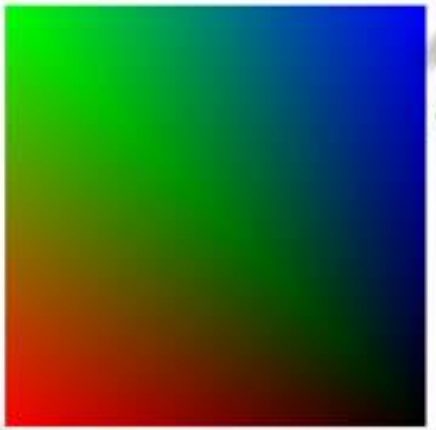
```
// Desenha dois triângulos
glBegin(GL_TRIANGLES);
    // Altera a cor do desenho para vermelho
    glColor3f(1.0f, 0.0f, 0.0f);
    glVertex2f(-35.0f,-14.0f);
    glVertex2f(-21.0f, 14.0f);
    glVertex2f( -7.0f,-14.0f);
    // Altera a cor do desenho para azul
    glColor3f(0.0f, 0.0f, 1.0f);
    glVertex2f(  7.0f, 14.0f);
    glVertex2f( 21.0f,-14.0f);
    glVertex2f( 35.0f, 14.0f);
glEnd();
```



Desenhos Bidimensionais

- Cores e Estilos de Traçado de Primitivas:

- Interpolação de Cores:
  - Permite interpolar cores no momento de desenhar uma primitiva
  - Definir os vértices das primitivas (pontos, linhas, triângulos ou quadriláteros) com cores distintas

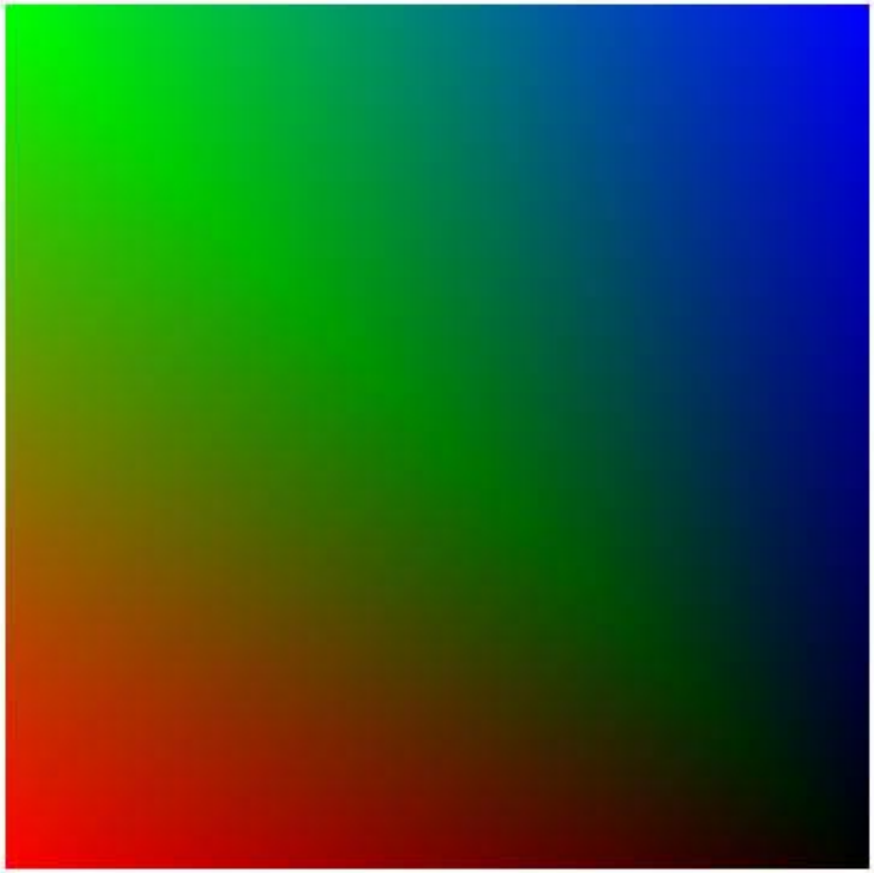


Desenhos Bidimensionais

- Cores e Estilos de Traçado de Primitivas:

- **Desenhando um quadrado cujos vértices possuem cores distintas**
- **Detalhando o código:**

```
// Desenha um quadrado no qual cada
// vértice possui uma cor diferente
glBegin(GL_QUADS);
    // Altera a cor do vértice para vermelha
    glColor3f(1.0f, 0.0f, 0.0f);
    glVertex2f(-40.0f,-40.0f);
    // Altera a cor do vértice para verde
    glColor3f(0.0f, 1.0f, 0.0f);
    glVertex2f(-40.0f,40.0f);
    // Altera a cor do vértice para azul
    glColor3f(0.0f, 0.0f, 1.0f);
    glVertex2f(40.0f,40.0f);
    // Altera a cor do vértice para preto
    glColor3f(0.0f, 0.0f, 0.0f);
    glVertex2f(40.0f,-40.0f);
glEnd();
```



Desenhos Bidimensionais

- Linhas (Cor, espessura e estilo):

- **Espessura:** função **glLineWidth** (em pixels)
- **Estilo:** função **glLineStipple**
- **Cor:** função **glColor3f**

glLineStipple (GLint factor, GLushort pattern)

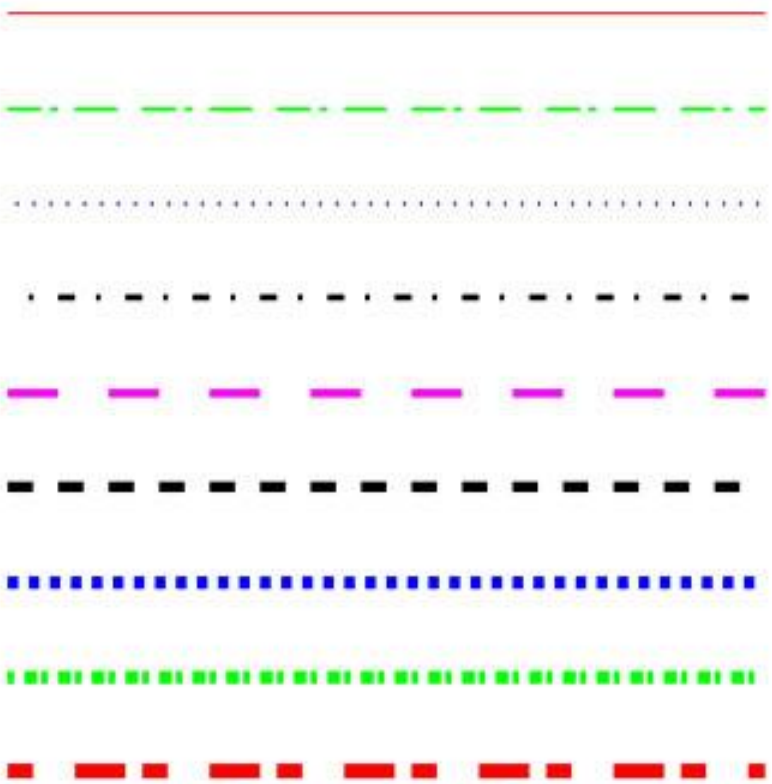
- valor (1 - 256)
- Fator de multiplicação para alterar o número de pixels do tracejado

- especifica o tipo de tracejado

Desenhos Bidimensionais

- Linhas (Cor, espessura e estilo):

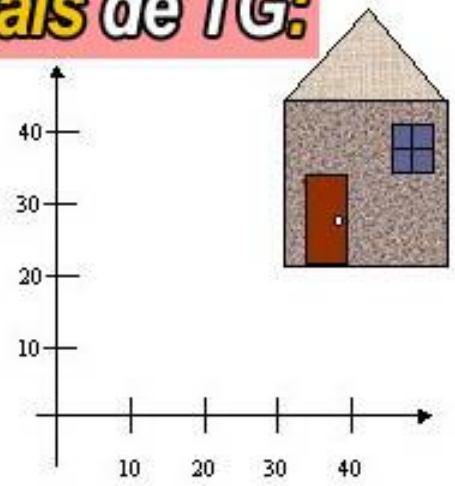
- Detalhando o código:

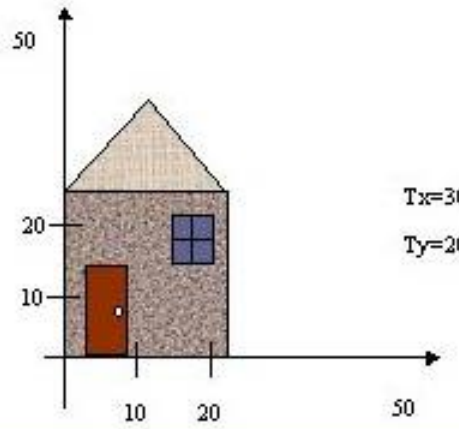


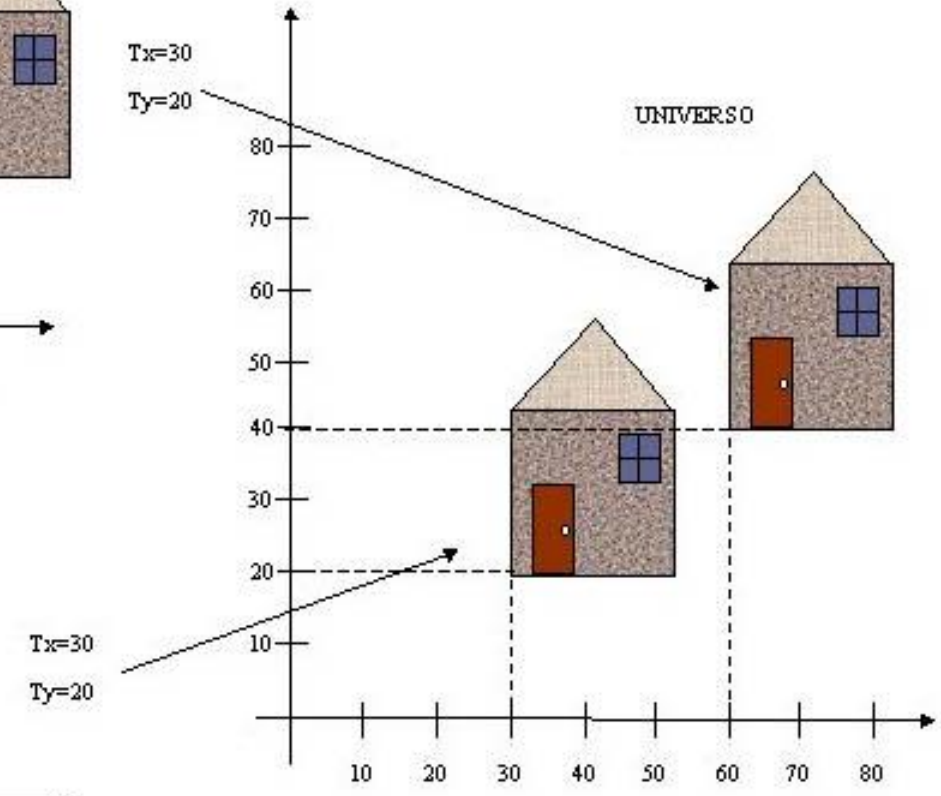
```
// Habilita a alteração do estilo das linhas
glEnable(GL_LINE_STIPPLE);

// Altera a cor do desenho para vermelho
glColor3f(1.0f, 0.0f, 0.0f);

// Altera a espessura da linha
glLineWidth(0.8);

// Altera o estilo da linha
glLineStipple(1, 0xFFFF);
glBegin(GL_LINES);
    glVertex2f(-40.0f, 40.0f);
    glVertex2f(40.0f, 40.0f);
glEnd();

// Altera a cor do desenho para verde
glColor3f(0.0f, 1.0f, 0.0f);

// Altera a espessura da linha
glLineWidth(1.6);

// Altera o estilo da linha
glLineStipple(4, 0x1F2F);
glBegin(GL_LINES);
    glVertex2f(-40.0f, 30.0f);
    glVertex2f(40.0f, 30.0f);
glEnd();
```

Transformações Geométricas

- Manipular e alterar as características dos objetos
- Tipos fundamentais de TG:
  - Translação
  - Rotação
  - Escala

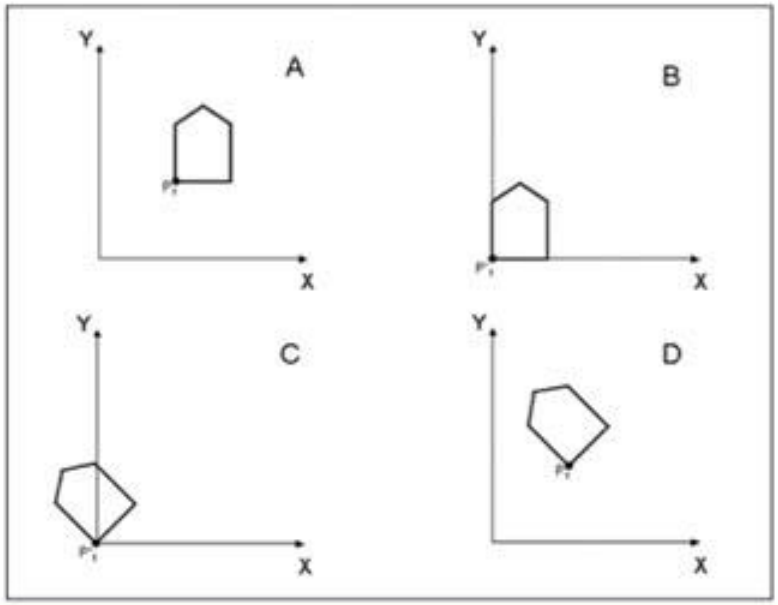






Transformações Geométricas

- Primitivas Gráficas:
  - Constituídas por meio de coordenadas cartesianas (vértices)
  - TG: consistem em operações de soma e multiplicação realizadas sobre essas coordenadas



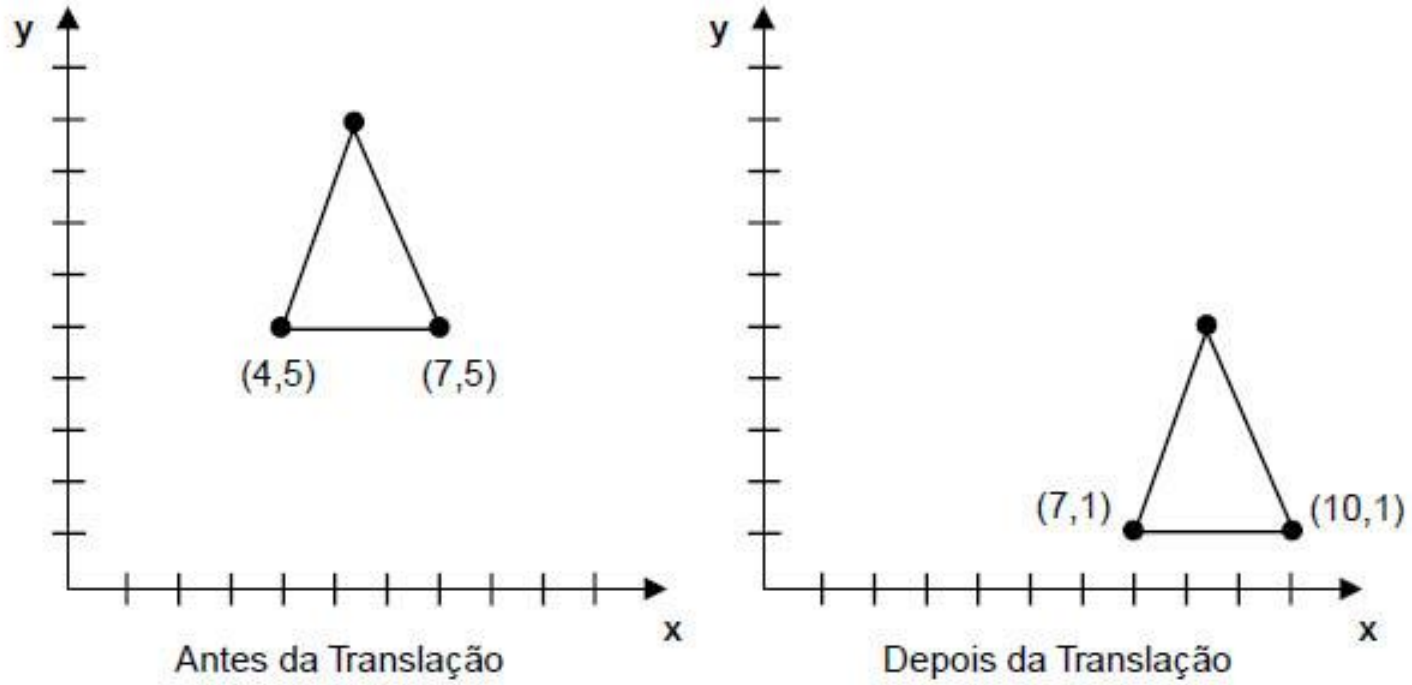
Transformações Geométricas

- Translação:
  - Usada para definir a posição de um objeto
  - Matematicamente: adicionar constantes de deslocamento a todas as coordenadas
  - Função: glTranslatef(x, y, z)

eixos

Transformações Geométricas

- Translação:

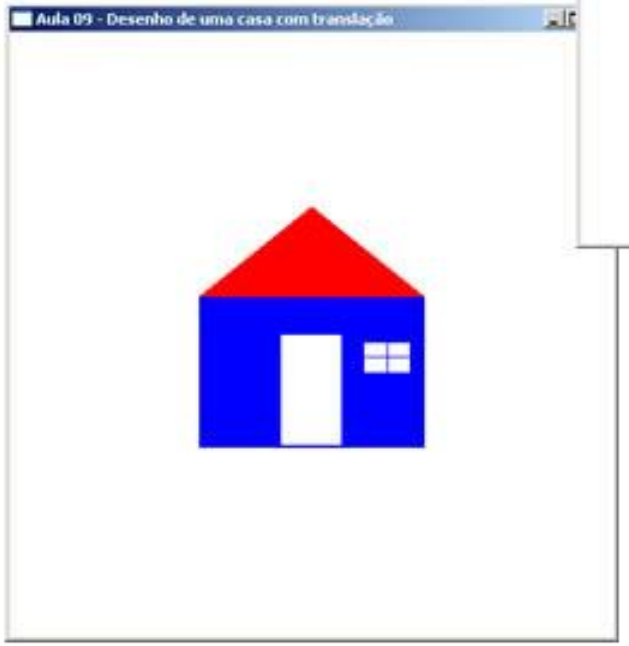


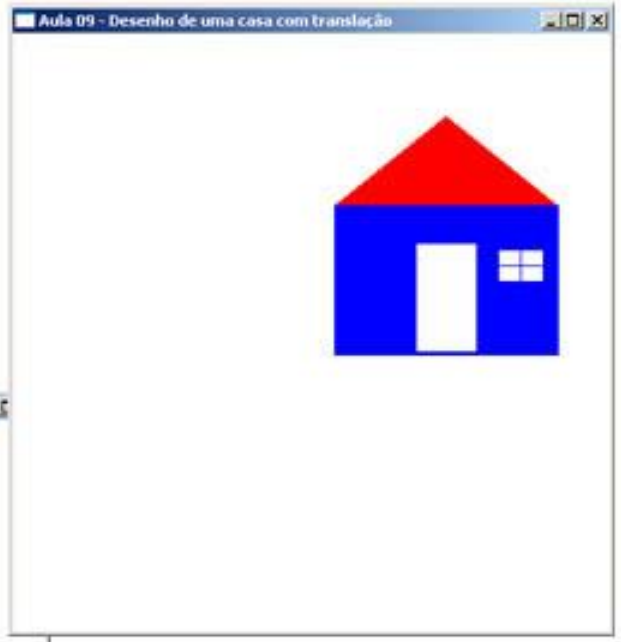
Transformações Geométricas

- Translação:

- Detalhando o código

```
// Aplica uma translação sobre a casinha que será desenhada
glTranslatef(18.0f, 12.0f, 0.0f);

// Desenha uma casinha composta de um quadrado e um triângulo

// Altera a cor do desenho para azul
glColor3f(0.0f, 0.0f, 1.0f);
// Desenha a casa
glBegin(GL_QUADS);
    glVertex2f(-15.0f,-15.0f);
    glVertex2f(-15.0f,  5.0f);
    glVertex2f( 15.0f,  5.0f);
    glVertex2f( 15.0f,-15.0f);
glEnd();
```





Transformações Geométricas

- Escala:
  - Usada para definir a escala a qual será utilizada para exibir um determinado objeto
  - Matematicamente: consiste em multiplicar um valor de escala por todas as coordenadas
  - Aumentar ou diminuir o tamanho
  - Função:

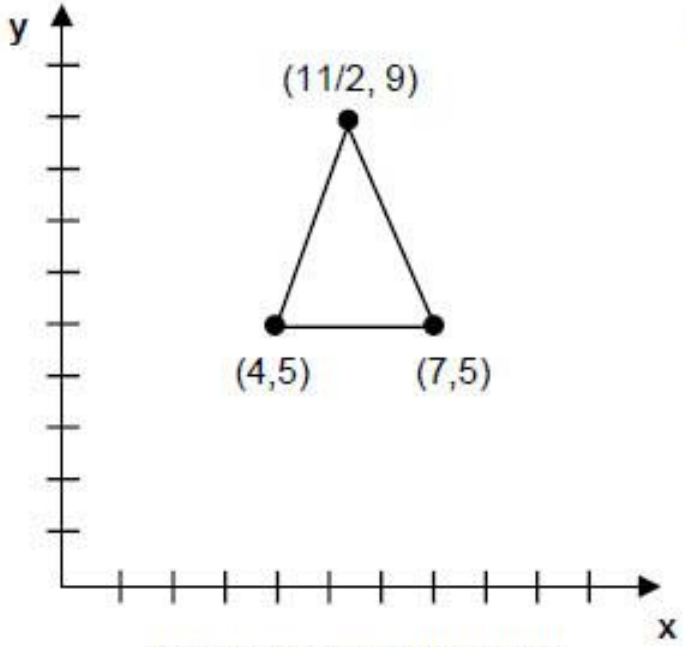
```
glScalef( x, y, z )
glScaled( x, y, z )
```

float
double

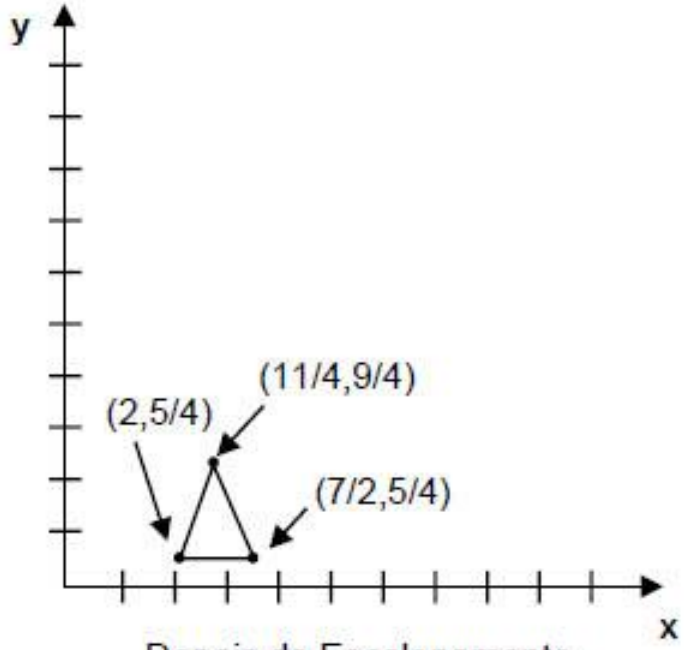
  - Observação: em 2D o terceiro argumento (valor 1)

Transformações Geométricas

- Escala:



Antes do Escalonamento



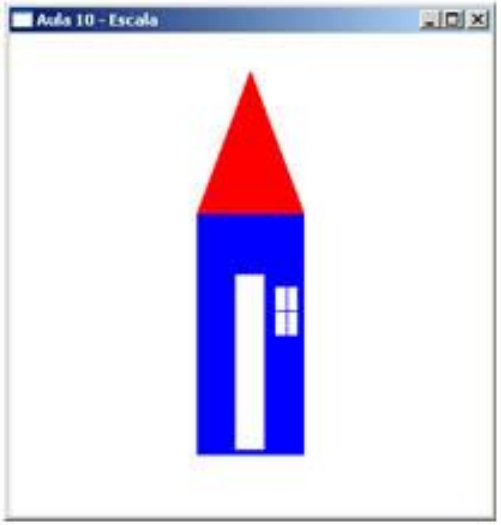
Depois do Escalonamento

Transformações Geométricas

- Escala:

- Detalhando o código

```
// Aplica uma escala sobre a casinha que será desenhada
glScalef(0.4f, 2.0f, 1.0f);

// Desenha uma casinha composta de um quadrado e um triângulo

// Altera a cor do desenho para azul
glColor3f(0.0f, 0.0f, 1.0f);
// Desenha a casa
glBegin(GL_QUADS);
    glVertex2f(-15.0f,-15.0f);
    glVertex2f(-15.0f,  5.0f);
    glVertex2f( 15.0f,  5.0f);
    glVertex2f( 15.0f,-15.0f);
glEnd();
```



Transformações Geométricas

- Rotação:
  - Usada para definir o valor do ângulo o qual será utilizada para girar o objeto
  - Matematicamente: consiste em aplicar uma composição de cálculos utilizando o seno e cosseno do ângulo de rotação a todas as coordenadas
  - Função:

    glRotatef(ângulo, x, y, z)
    glRotated(ângulo, x, y, z)

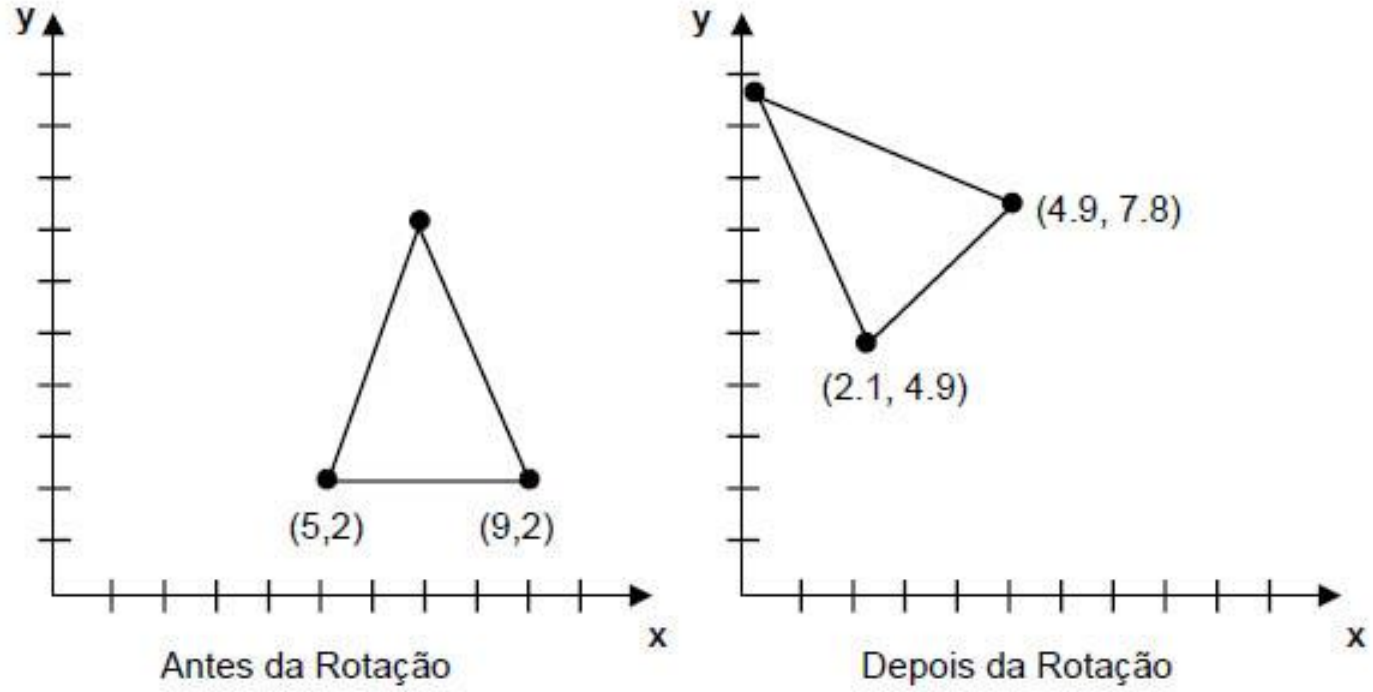
    float

    double
  - Observação: em 2D, a rotação é feita ao redor do eixo z

    Além do ângulo, deverá ser utilizado os valores (0, 0, 1) como argumentos (x, y e z)

Transformações Geométricas

- Rotação:

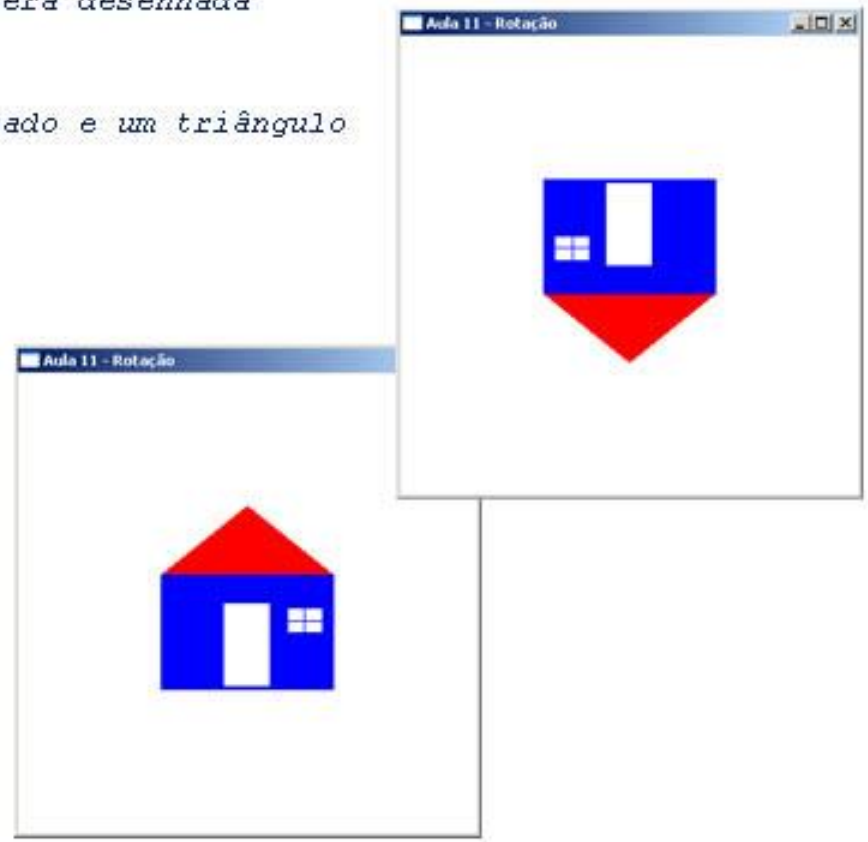


Transformações Geométricas

- Rotação:

- Detalhando o código

```
// Aplica uma rotação sobre a casinha que será desenhada
glRotatef(180.0f, 0.0f, 0.0f, 1.0f);

// Desenha uma casinha composta de um quadrado e um triângulo

// Altera a cor do desenho para azul
glColor3f(0.0f, 0.0f, 1.0f);
// Desenha a casa
glBegin(GL_QUADS);
    glVertex2f(-15.0f,-15.0f);
    glVertex2f(-15.0f,  5.0f);
    glVertex2f( 15.0f,  5.0f);
    glVertex2f( 15.0f,-15.0f);
glEnd();
```



Animação

- Tópicos:
  - Animação (Conceitos Básicos)
    - Animação e Computadores
    - Animação Tridimensional

Animação

- **Animação:**
  - Gráficos em seqüências

- **Animação por Computador:**
  - Processo de exibição (seqüência de objetos visuais gerados por computador)
  - Taxa suficiente para provocar sensação de movimento (25 / s)
  - Objeto animado: variação de
    - Posição
    - Tamanho
    - Forma

Animação

- Vantagens:
  - Transmite grandes quantidades de informação e são por si só, uma fonte de motivação
  - Relativamente ao Vídeo:
    - Conjunto de imagens
    - Ocupa menor espaço
    - Constituída por um conjunto de gráficos
  - Animação por Computador:
    - Aplicada em filmes como também em apresentações assistidas por computador permitindo a criação de efeitos visuais

Animação

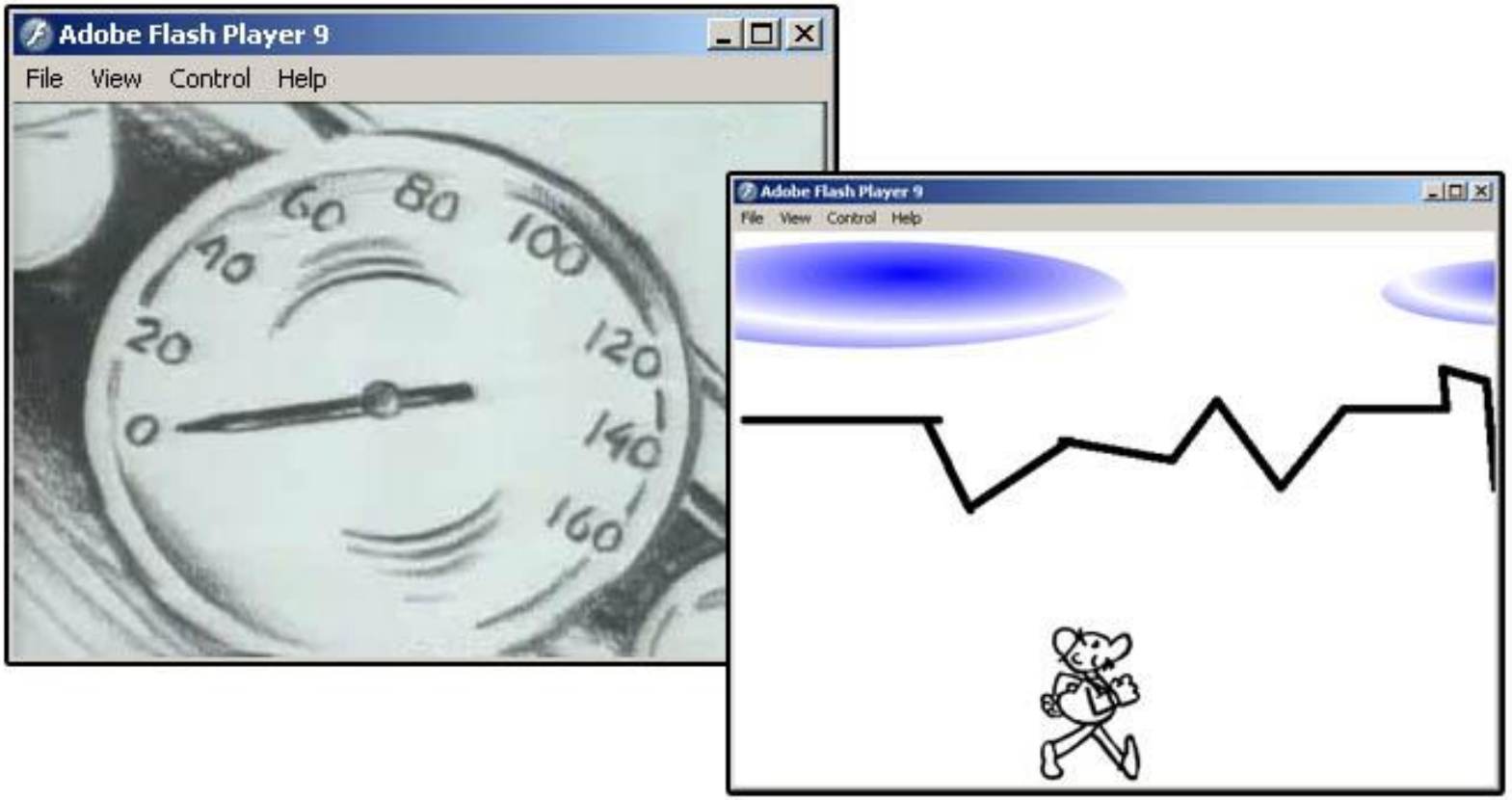
- Exemplos:
  - Educação Médica
  - Entretenimento

Animação

- Observação:
  - Uma técnica de animação que se tornou famosa pela Disney foi o uso progressivo de diferentes desenhos gráficos em cada frame de um filme (24 frames/s) chamada de animação por células (cell animation)
  - Um minuto de animação requer até 1.440 frames/s
  - O termo célula deriva do material usado para produzir a frame – folhas de celulóide, atualmente substituídas por plástico

Animação

- Exemplos (Frame a Frame)



Animação

- Tipos de Animação por Computador:
  - Convencional Assistida por Computador
  - Bidimensional
  - Tridimensional

Animação

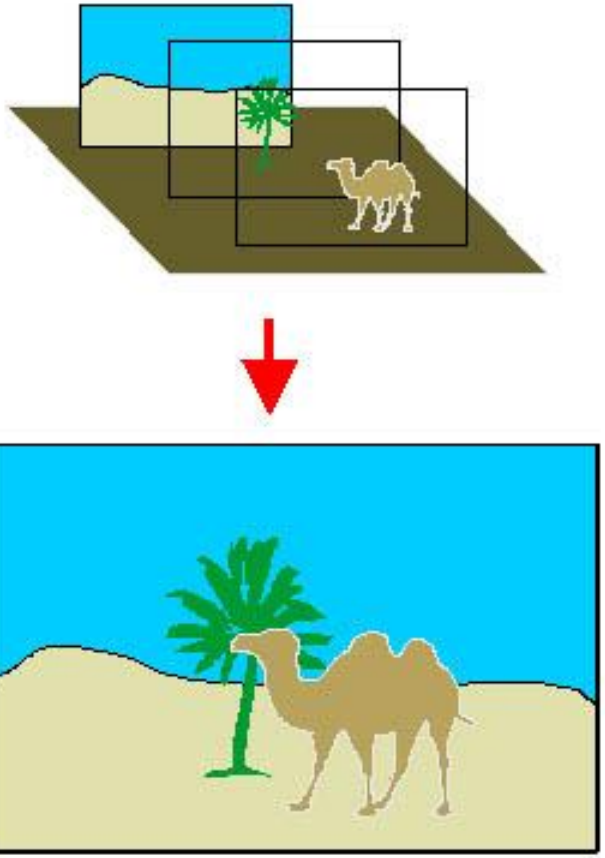
- Convencional Assistida por Computador:

  - Função principal: armazenamento, composição e edição de desenhos

  - Outras funções: colorização, efeitos especiais, sonorização

  - Procura manter o visual de animação manual

- Construção de uma Animação Convencional:



layers

Animação

- **Animação 2D ou bidimensional:**
  - Tipo de animação mais comum
  - Tem apenas largura (eixo do X) e altura (eixo do Y)
  - Cenas construídas por objetos:
    - Entidades gráficas simples ou compostas
  - Objetos com capacidade de movimento e deformação:
    - Gráficos que compõem um objeto são transladadas, giradas ou deformadas
  - Transformações variam de quadro a quadro:
    - Transmitindo ilusão de movimento

Animação

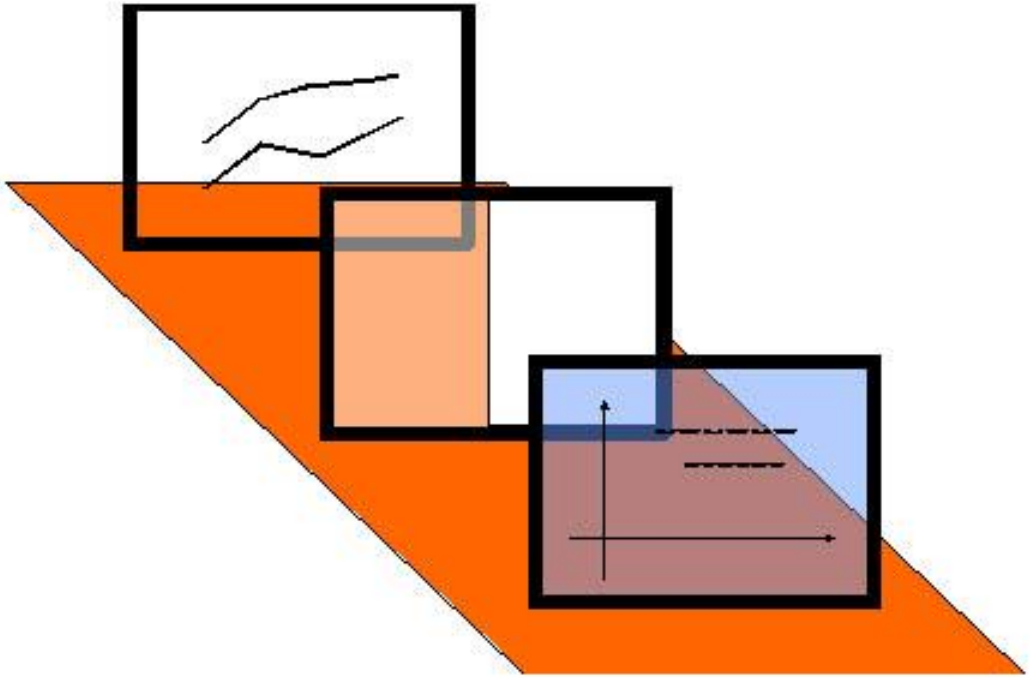
- Animação constituída de quadro a quadro:

Animação

- Exemplo de Interpolação:

Animação

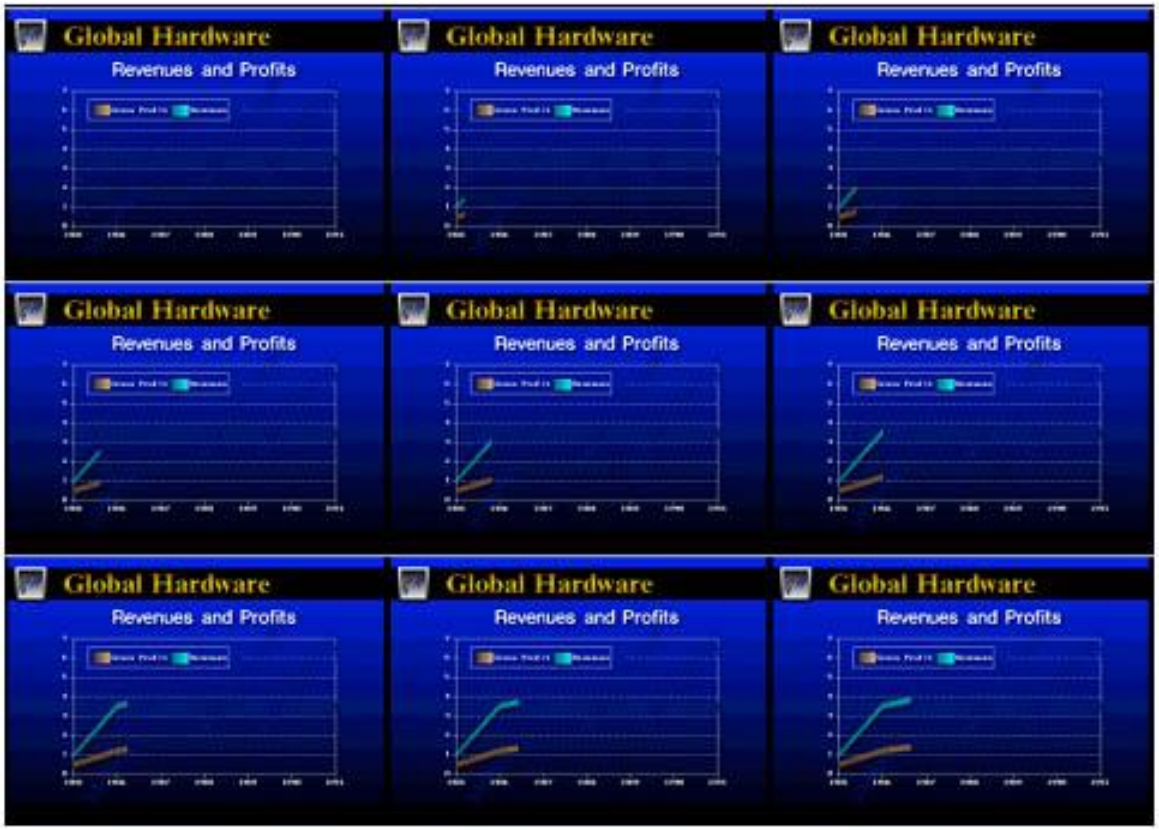
- Realização de composição com “ matte ” (silhueta):



- Matte:
  - Utilizadas para esconder objetos em uma animação
  - Silhueta altera ou se desloca ao longo do tempo

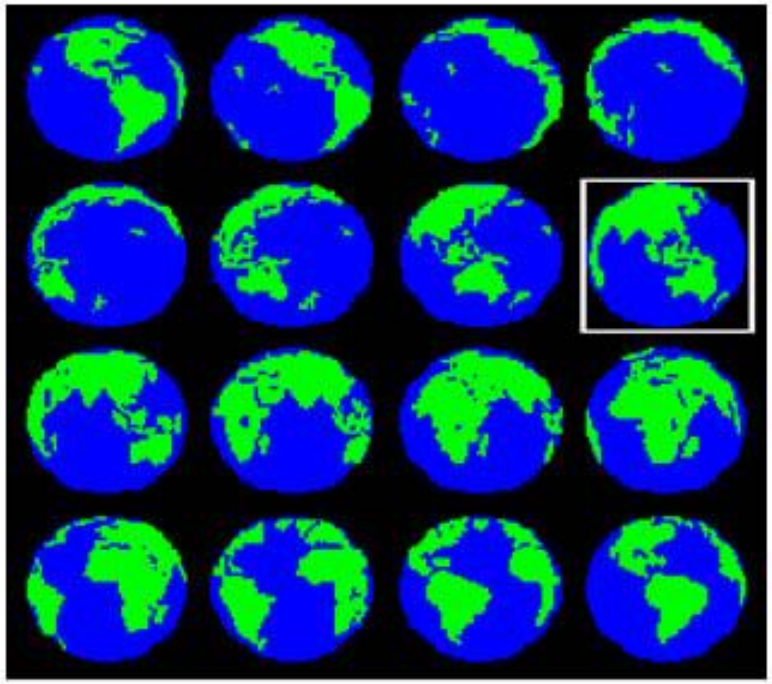
Animação

- Efeito de "matte" digital:



Animação

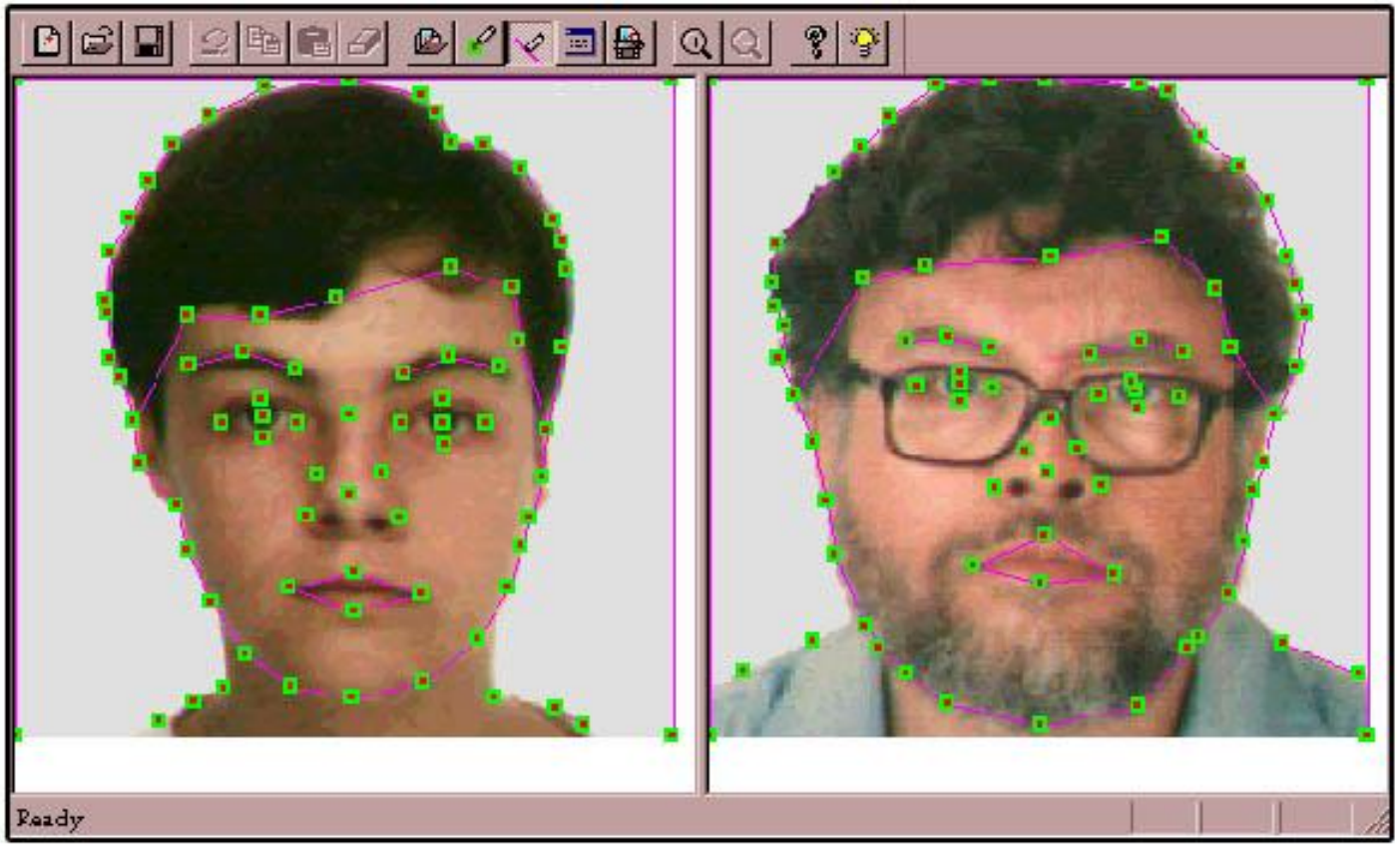
- Animação por troca de imagens acionada por script:



Animação

- Morfos:
  - Interpolações bidimensionais
  - Captura dos setores da imagem de origem
  - Aplicação nos correspondentes do destino

- Edição de morfos:



Animação

- Resultado de morfo:

Animação

- **Animação 3D ou tridimensional:**
  - Desenhar uma imagem a três dimensões implica considerar os vários ângulos sob os quais essa imagem pode ser observada
  - Criar efeitos especiais no cinema e nos jogos de computador
  - Procura simular filmagens no mundo real
  - Visualização de objetos 3D por todos os ângulos
  - Modelagem requer técnicas avançadas

Animação

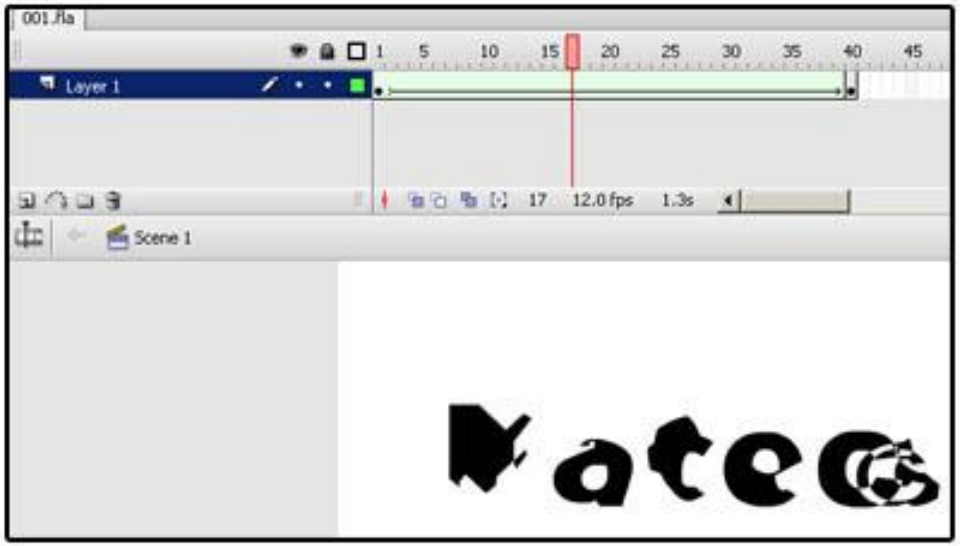
- Aplicações da animação tridimensional:
  - Publicidade e vinhetas
  - Entretenimento: filmes, jogos e parques
  - Visualização técnica e científica:
    - Simulações
    - Imagens médicas
    - Maquetes virtuais

Animação

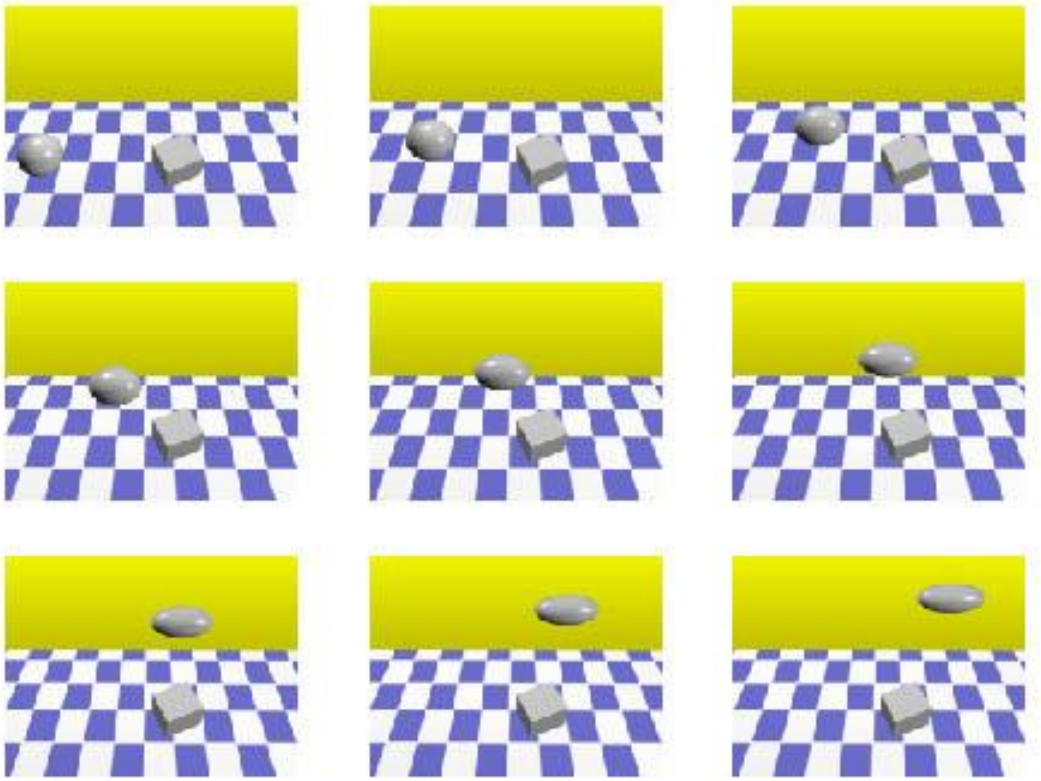
- O ciclo da animação tridimensional:
  - Modelagem
  - Elaboração
  - Coreografia
  - Pós-produção

Animação

- Coreografia - Técnicas básicas:
  - Interpolação entre quadros-chave modelados manualmente
  - Os quadros-chave devem captar os movimentos essenciais

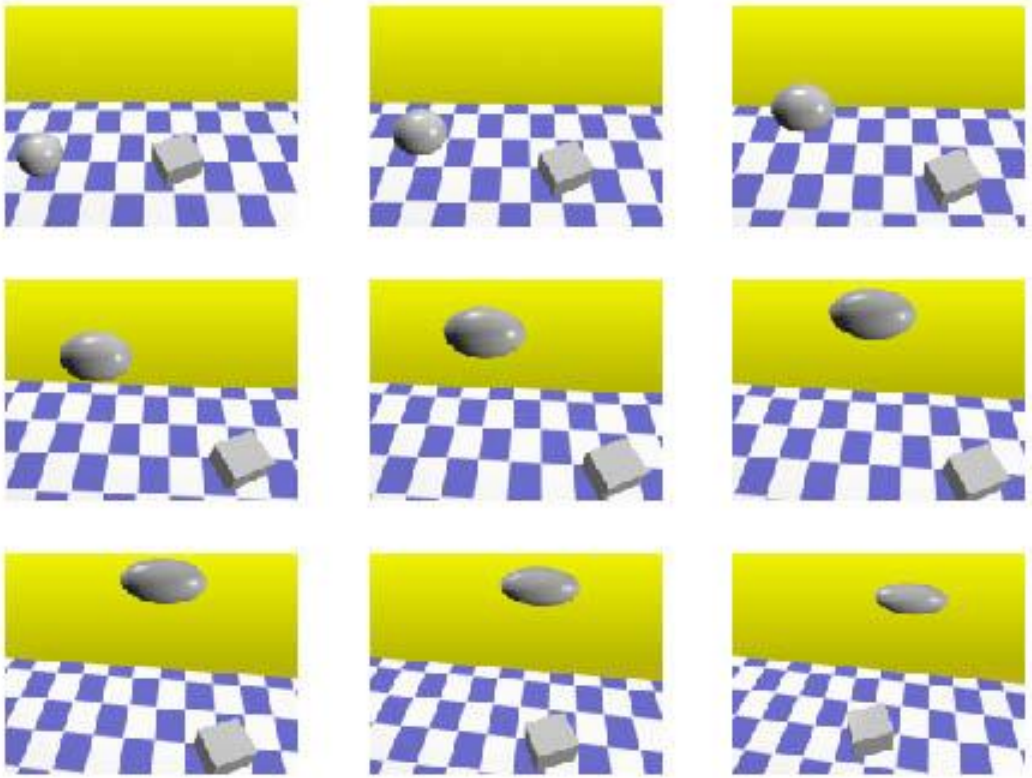


- Interpolação tridimensional simples:



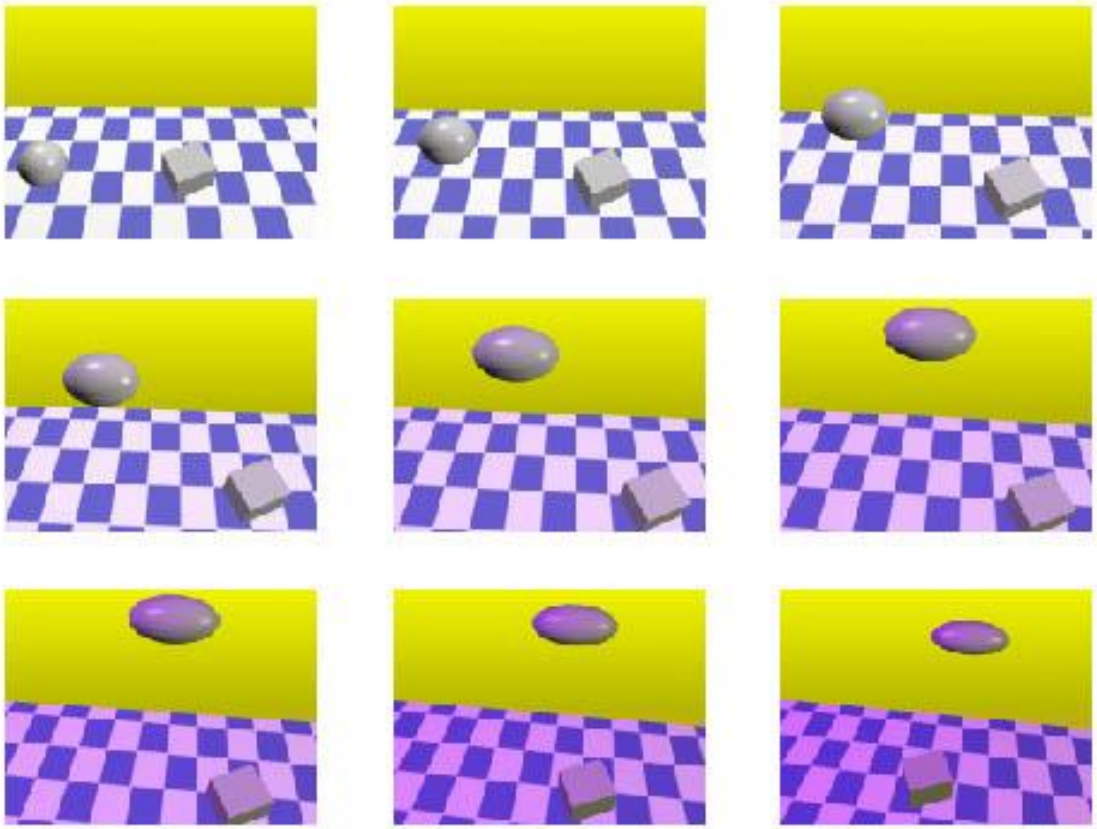
Animação

- Coreografia com movimento de câmera:



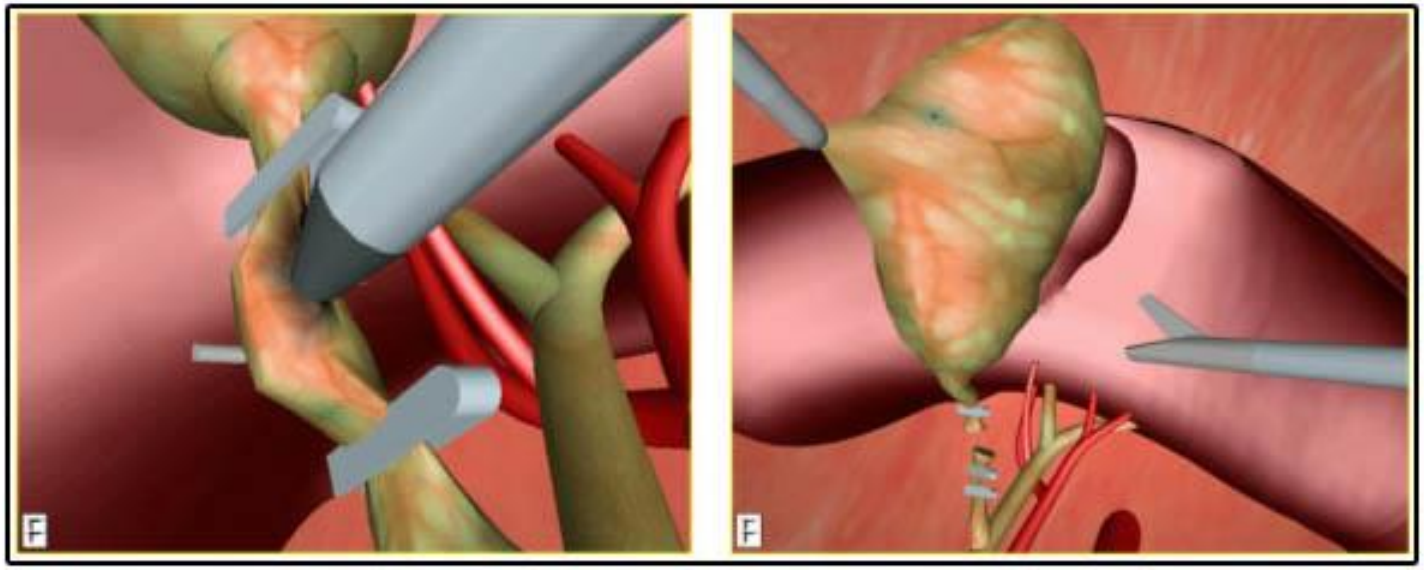
Animação

- Coreografia com variação de luzes:



Animação

- Animação em tempo real:
  - Limitada pela velocidade de processamento
  - Utilizada em aplicativos com interfaces multimídia
  - Aplicações avançadas em simuladores e realidade virtual



Animação - OpenGL

- Biblioteca GLUT: animação 2D e 3D
  - Redesenhar a cena continuamente (seqüencia / frame)
  - Exibição rápida (ilusão de movimento)
  - O simples fato de alterar cor e forma dos objetos pode ser considerada uma animação
  - 3D: simples movimento do observador produz efeitos de animação, mesmo que os objetos estejam estáticos

Animação - OpenGL

- Animação 2D
  - Limitada ao movimento de objetos ou mudança de forma / cores
  - Função callback
    - chamada em intervalos de tempo predeterminados
    - força a atualização contínua da tela
  - Evitar / imagem piscando
    - utilização de 2 buffers de exibição
    - glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    - substituir a função glFlush por glutSwapBuffers

Animação - OpenGL

- Função:
  glutTimerFunc(10,Anima2D, 1);

- Associação de uma função callback a ser executada de tempos em tempos

- Primeiro Parâmetro: define o tempo (milissegundos)

- Segundo Parâmetro: nome da função de callback

- Terceiro Parâmetro: valor inteiro passado para a função de callback como parâmetro

Animação - OpenGL

```
#include <stdlib.h>
#include <GL/glut.h>

// Variáveis que guardam a translação que será aplicada
// sobre a casinha
GLfloat Tx;
GLfloat Ty;

// Variáveis que guardam os valores mínimos de x e y da
// casinha
GLfloat minX, maxX;
GLfloat minY, maxY;

// Variáveis que guardam o tamanho do incremento nas
// direções x e y (número de pixels para se mover a
// cada intervalo de tempo)
GLfloat xStep;
GLfloat yStep;

// Variáveis que guardam a largura e altura da janela
GLfloat windowXmin, windowXmax;
GLfloat windowYmin, windowYmax;
```

Animação - OpenGL

```
// Função callback de redesenho da janela de visualização
void Desenha(void)
{
    // Muda para o sistema de coordenadas do modelo
    glMatrixMode(GL_MODELVIEW);
    // Inicializa a matriz de transformação corrente
    glLoadIdentity();

    // Limpa a janela de visualização com a cor
    // de fundo definida previamente
    glClear(GL_COLOR_BUFFER_BIT);

    // Aplica uma translação sobre a casinha
    glTranslatef(Tx, Ty, 0.0f);
```

Animação - OpenGL

```
// Função responsável por inicializar parâmetros e variáveis
void Inicializa (void)
{
    // Define a cor de fundo da janela de visualização como branca
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

    // Inicialização das variáveis globais
    xStep = yStep = 1.0f;
    Tx = Ty = 0.0f;
    minX = -15.0f;
    maxX =  15.0f;
    minY = -15.0f;
    maxY =  17.0f;
    windowXmin = windowYmin = -40.0f;
    windowXmax = windowYmax = 40.0f;
}
```

Animação - OpenGL

```
// Função callback chamada pela GLUT a cada intervalo de tempo
void Anima2D(int value)
{
    // Muda a direção quando chega na borda esquerda ou direita
    if( (Tx+maxX) > windowXmax || (Tx+minX) < windowXmin )
        xStep = -xStep;

    // Muda a direção quando chega na borda superior ou inferior
    if( (Ty+maxY) > windowYmax || (Ty+minY) < windowYmin )
        yStep = -yStep;

    // Move a casinha
    Tx += xStep;
    Ty += yStep;

    // Redesenha a casinha em outra posição
    glutPostRedisplay();
    glutTimerFunc(150,Anima2D, 1);
}
```

Exercício