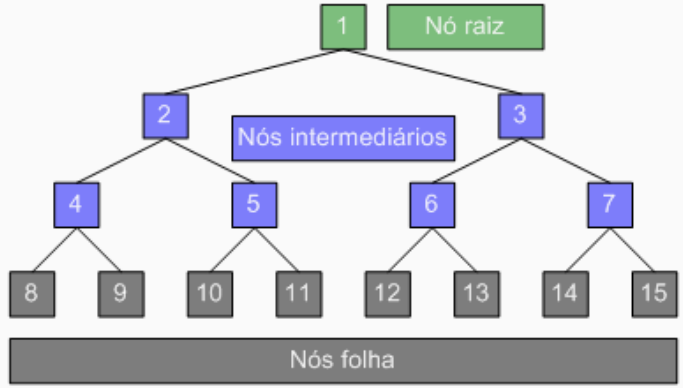
COMPUTAÇÃO GRÁFICA

Modelagem Hierárquica

Prof. Moisés Henrique Ramos Pereira

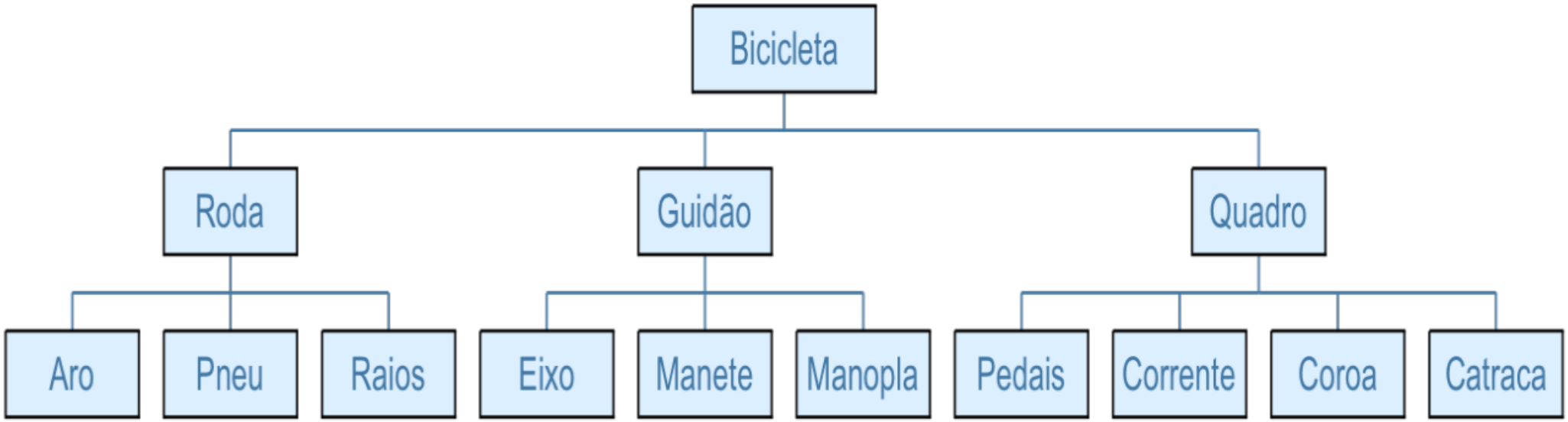


Introdução

- Para a definição de objetos ou sistemas complexos, em geral, é mais vantajoso dividir o processo em duas partes:
  - ✓ 1. Especificar as subpartes do objeto ou sistema.
  - ✓ 2. Descrever como estas subpartes são agrupadas, formando assim o objeto ou sistema.
- Exemplo:
  - ✓ Bicicleta

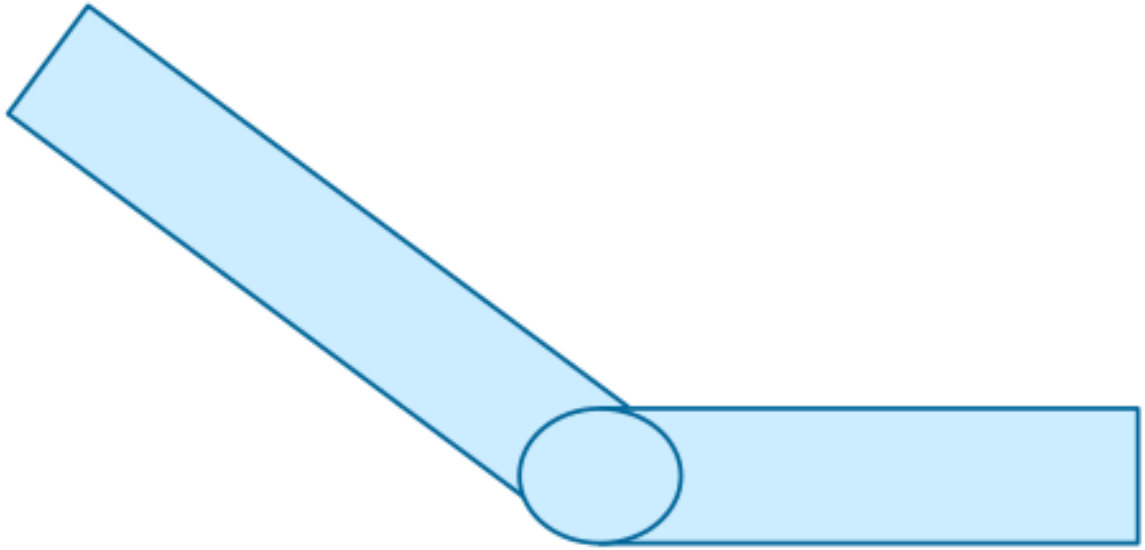
Introdução

- É a descrição de um objeto complexo composto de diversas partes, onde cada parte possui relação hierárquica com outra parte.
  - ✓ A descrição de cada uma das partes é aninhada em outra parte criando uma organização em árvore.



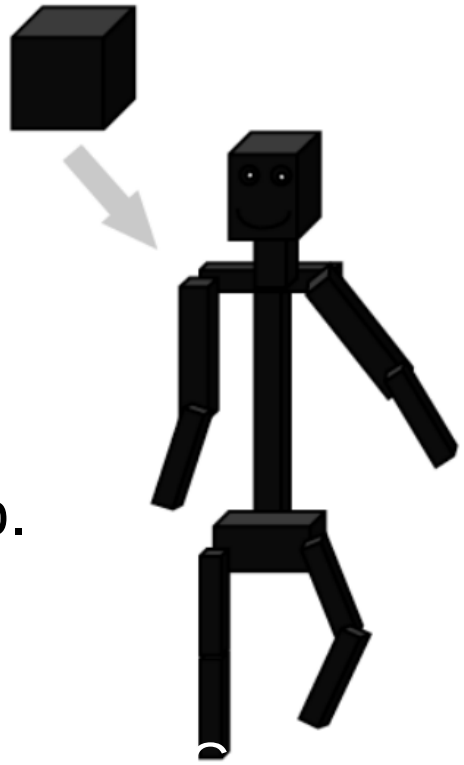
Estrutura ou Agrupamento?

- É mais conveniente criar e tratar um grupo de objetos como se fossem um único objeto do que tratá-los de forma separada.



Estrutura ou Agrupamento?

- Aprenderemos todos os passos para criarmos um boneco palito utilizando cubos.
  - ✓ Utilizando rotações, translações, escalas em cada um dos cubos é possível dar a cada um dos cubos a exata:
    - Posição
    - Tamanho
    - Orientação
  - ✓ Funciona perfeitamente até movermos o boneco.

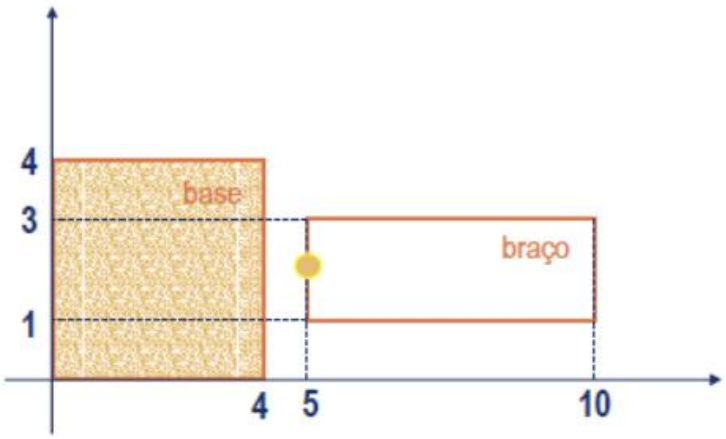


Estrutura ou Agrupamento?

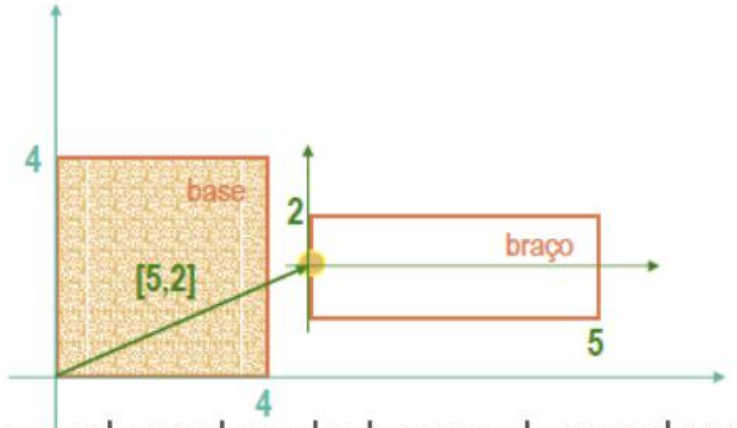
- Assim que tentamos mudar alguma parte, nosso modelo desmorona:
  - ✓ O objeto, ao ser modelado, possui limitações de movimentação, mas o modelo não tem ciência disso.
  - ✓ É preciso definir uma maneira que torne fácil a movimentação do boneco palito, mas sem desrespeitar suas limitações:
    - Modelagem hierárquica

Métodos para Modelagem Hierárquica

- De forma geral, modelos são construídos a partir de instâncias de formas geométricas.
  - ✓ Cada instância é definida com relação ao sistema de coordenadas do mundo do modelo, também chamado de coordenadas locais do objeto.
  - ✓ Transformações de modelagem resultam no posicionamento do objeto com relação a outros objetos.



coordenadas locais



coordenadas do braço dependem da base

No OpenGL

- O OpenGL possui duas formas de realizar a modelagem hierárquica:

  1. Pilha de matrizes (matriz stack) - quando um novo componente geométrico é adicionado a uma estrutura, sua matriz de transformação é empilhada na estrutura.

  2. Display lists - permitem encapsular os atributos de cada componente, tornando a exibição mais eficiente.

Pilha de Matrizes

- Transformações Modelview e Projection correspondem a uma pilha de matrizes 4 x 4 (lembre-se que o OpenGL utiliza, internamente, coordenadas homogêneas)
  - ✓ glMatrixMode(mode)
  - ✓ mode = GL_MODELVIEW, GL_PROJECTION, GL_TEXTURE
  - ✓ glPushMatrix(void)
  - ✓ glPopMatrix(void)
  - ✓ glLoadIdentity(void)
  - ✓ glLoadMatrix{fd}(const TYPE *M)
  - ✓ glMultMatrix{fd}(const TYPE *M)

Pilha de Matrizes

- Exemplo: modelo de carro
  - ✓ (www.glprogramming.com/red/chapter03.html)
  - ✓ Desenhar um carro, seu quatro pneus com cinco parafusos:
    1. Desenhar o carro.
    2. Armazene sua posição e translade para a posição da roda direita dianteira.
    3. Desenhe a roda (e os parafusos) e jogue fora a translação de forma a retornar à posição inicial.
    4. Armazene sua posição e translade para a roda esquerda dianteira.

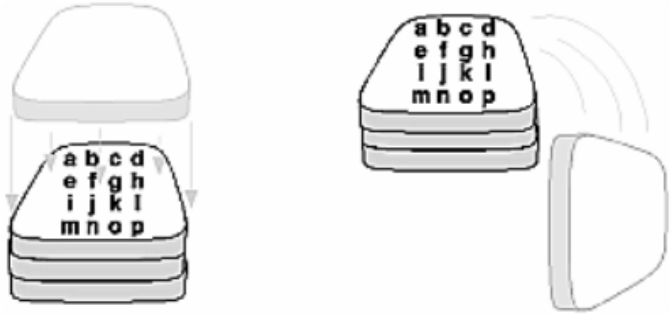
Pilha de Matrizes

- Uma vez que transformações são representadas por matrizes, uma pilha de matriz é o mecanismo perfeito para estas sucessivas operações de armazenar posição, transladar e retornar.

- O controle de qual matriz de transformação esta no topo da pilha é dado pelas seguintes funções:

  - ✓ glPushMatrix(void) - copia a matriz corrente e adiciona a cópia ao topo da pilha de matrizes.

  - ✓ glPopMatrix(void) - descarta a matriz no topo da pilha.



Pilha de Matrizes

- Fragmento de código:

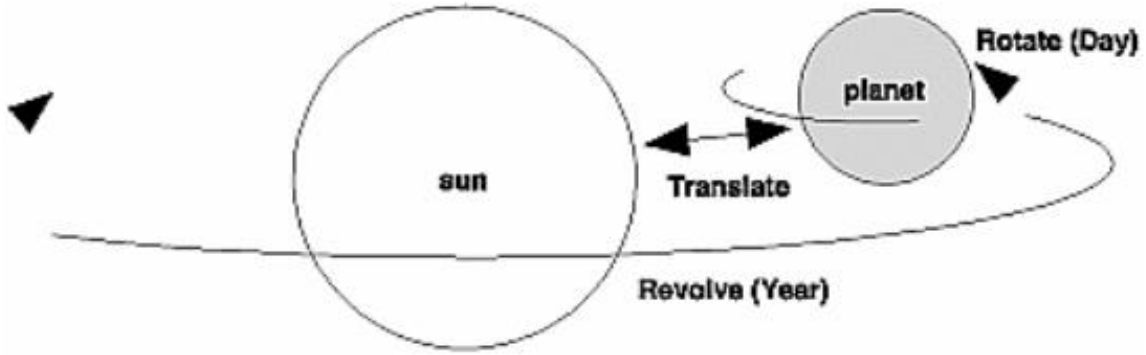
```
draw_body_and_wheel_and_bolts()
{
  draw_car_body();
  glPushMatrix();
    glTranslatef(40,0,30);    /*move to first wheel position*/
    draw_wheel_and_bolts();
  glPopMatrix();
  glPushMatrix();
    glTranslatef(40,0,-30);   /*move to 2nd wheel position*/
    draw_wheel_and_bolts();
  glPopMatrix();
  ...                 /*draw last two wheels similarly*/
}
```

```
draw_wheel_and_bolts()
{
  long i;

  draw_wheel();
  for(i=0;i<5;i++){
    glPushMatrix();
      glRotatef(72.0*i,0.0,0.0,1.0);
      glTranslatef(3.0,0.0,0.0);
      draw_bolt();
    glPopMatrix();
  }
}
```

Pilha de Matrizes

- Exemplo: Sistema solar
  - ✓ Desenhar o sol e um planeta que rotaciona em torno de seu próprio eixo e translada em torno do sol.
    - Utilizamos glRotate() para a rotação em torno do eixo do planeta e glTranslate() para a translação;
    - Utilizamos glutWireSphere() para a criação dos planetas.



Pilha de Matrizes

- Fragmento de código:

```
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>

static int year = 0, day = 0;


int main(int argc, char** argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);
  glutInitWindowSize (500, 500);
  glutInitWindowPosition (100, 100);
  glutCreateWindow (argv[0]);
  init ();
  glutDisplayFunc(display);
  glutReshapeFunc(reshape);
  glutKeyboardFunc(keyboard);
  glutMainLoop();
  return 0;
}
```

```
void reshape (int w, int h)
{
  glViewport (0, 0, (GLsizei) w, (GLsizei) h);
  glMatrixMode (GL_PROJECTION);
  glLoadIdentity ();
  gluPerspective(60.0, (GLfloat) w/(GLfloat) h,
1.0, 20.0);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  gluLookAt (0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0,
1.0, 0.0);
}
```

Pilha de Matrizes

- Fragmento de código:

```
void init(void)
{
  glClearColor (0.0, 0.0, 0.0, 0.0);
  glShadeModel (GL_FLAT);
}

void display(void)
{
  glClear (GL_COLOR_BUFFER_BIT);
  glColor3f (1.0, 1.0, 1.0);

  glPushMatrix();
  glutWireSphere(1.0, 20, 16);   /* draw sun */
  glRotatef ((GLfloat) year, 0.0, 1.0, 0.0);
  glTranslatef (2.0, 0.0, 0.0);
  glRotatef ((GLfloat) day, 0.0, 1.0, 0.0);
  glutWireSphere(0.2, 10, 8); /* draw smaller
planet */
  glPopMatrix();
  glutSwapBuffers();
}
```

```
void keyboard (unsigned char key,
                    int x, int y)
{
  switch (key) {
    case `d':
      day = (day + 10) % 360;
      glutPostRedisplay();
      break;
    case `D':
      day = (day - 10) % 360;
      glutPostRedisplay();
      break;
    case `y':
      year = (year + 5) % 360;
      glutPostRedisplay();
      break;
    case `Y':
      year = (year - 5) % 360;
      glutPostRedisplay();
      break;
    default:
      break;
  }
}
```