TP 2b - Robot

Geometria
Transformações Geométricas

DEI – 2021/22 - Computação Gráfica

Jorge Henriques
André Perrotta

Objectivos

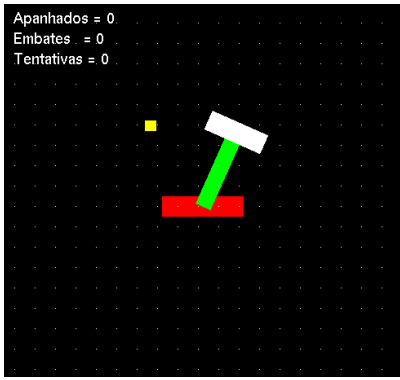
- Estudar como o OpenGL permite implementar operações básicas de transformação :
  - Translação, Rotação, Escala
  - Ordem das transformações

- ***Em particular neste trabalho***
  - **Gestão de push / pop**
  - **Combinação de transformações**
  - **Matriz model_View**

Objectivos

- Além disso
  - Definição e criação de janelas de visualização
  - Sistema de **coordenadas mundo** e sistema de **coordenadas da janela** de visualização
  - Bibliotecas típicas em OpenGl e sua utilização
  - Utilização de variáveis e comandos em OpenGL
  - Desenho de primitivas básicas
  - Especificação de cores
  - Gestão de eventos (teclado)
  - Desenho de caracteres (fontes bitmap)

Objectivos



JOGO – Regras

- ***Quadrado***= parte de uma posição semi-aleatória,

- ***Robot***
  - uma **base**, que se move apenas na horizontal (setas esquerda e direita, por exemplo)
  - uma **ligação**, que roda em torno da base (utilizando por exemplo as setas cima e baixo)
  - uma **extremidade=escudo** que roda livremente

- JOGO : <u>OPCIONAL</u> ! Não é fundamental implementar !

  Regras
  - O quadrado (amarelo) percorre o ecrã podendo acontecer uma de duas coisas:
  - O jogador apanha-o com a extremidade (**apanhados++**)
  - O quadrado embate na base (**embates++**)

  Final
  - O jogo termina quando ***embates>=apanhados+3 ??***
  - O jogo termina quando ??? Inventar um critério ???

openGL

1. Desenhar o objecto ROBOT

- Objecto "complexo" constituído por 3 objectos simples
  - Quadrados, cubo, …

- Transformações = operações sobre os objectos
  - Translações, rotações, escalas

  - Atenção:
    - Operações podem não ser comutativas:
      Ex: (Rotação + Translação) ***DIFERENTE DE*** (Translação + Rotação) !!!
    - Em openGl a "***ordem é inversa***" (baixo para cima) !!!

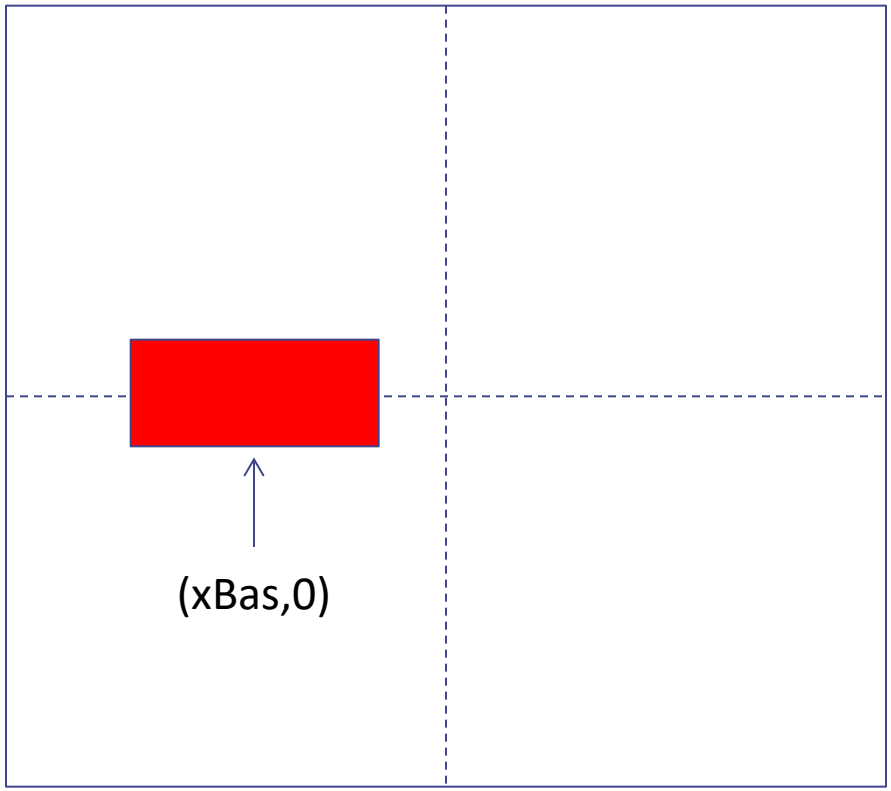
Às vezes dá jeito que as transformações sejam *globais* a todos os objectos

1. *Desenhar* o ROBOT

- Construído com base em a partir de objectos simples/básicos
  - Quadrados (2D), **Cubos (3D)**
- Uma possibilidade de definir o objecto vértice a vértice
  - Como nas aulas anteriores
- Pode-se usar alguns objectos já definidos na GLUT
  - **glutSolidCube(1.0);**
  - **Desenha um cubo de dimensão = 1.0, centrado na origem!**
  - **Vamos usar o cubo como objecto básico !**
  - **Se preferir, use um quadrado 2D, centrado na origem.**

3. ROBOT: composição de operações

- Transformações comuns / individuais




```
glColor3f(1,0,0);
glTranslatef (xBas, 0.0, 0.0);   //GLOBAL a todos os objectos
glPushMatrix();
        glScalef (wSup, hSup, 1.0);
        glutSolidCube(1.0);
glPopMatrix();
```

xBas

Se tecla → xBas = xBas + deslocaX

Se tecla ← xBas = xBas - deslocaX

openGL

Uso de push e pop ?

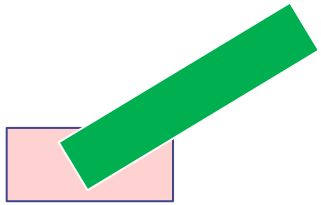
Movimentos independentes ?

Push
- Transforma e desenha **base**

Pop

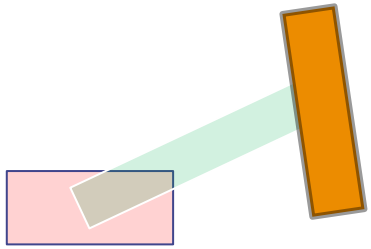
Push
- Transforma e desenha **Ligação**

Pop



Push
- Transforma e desenha **Extremidade**

Pop



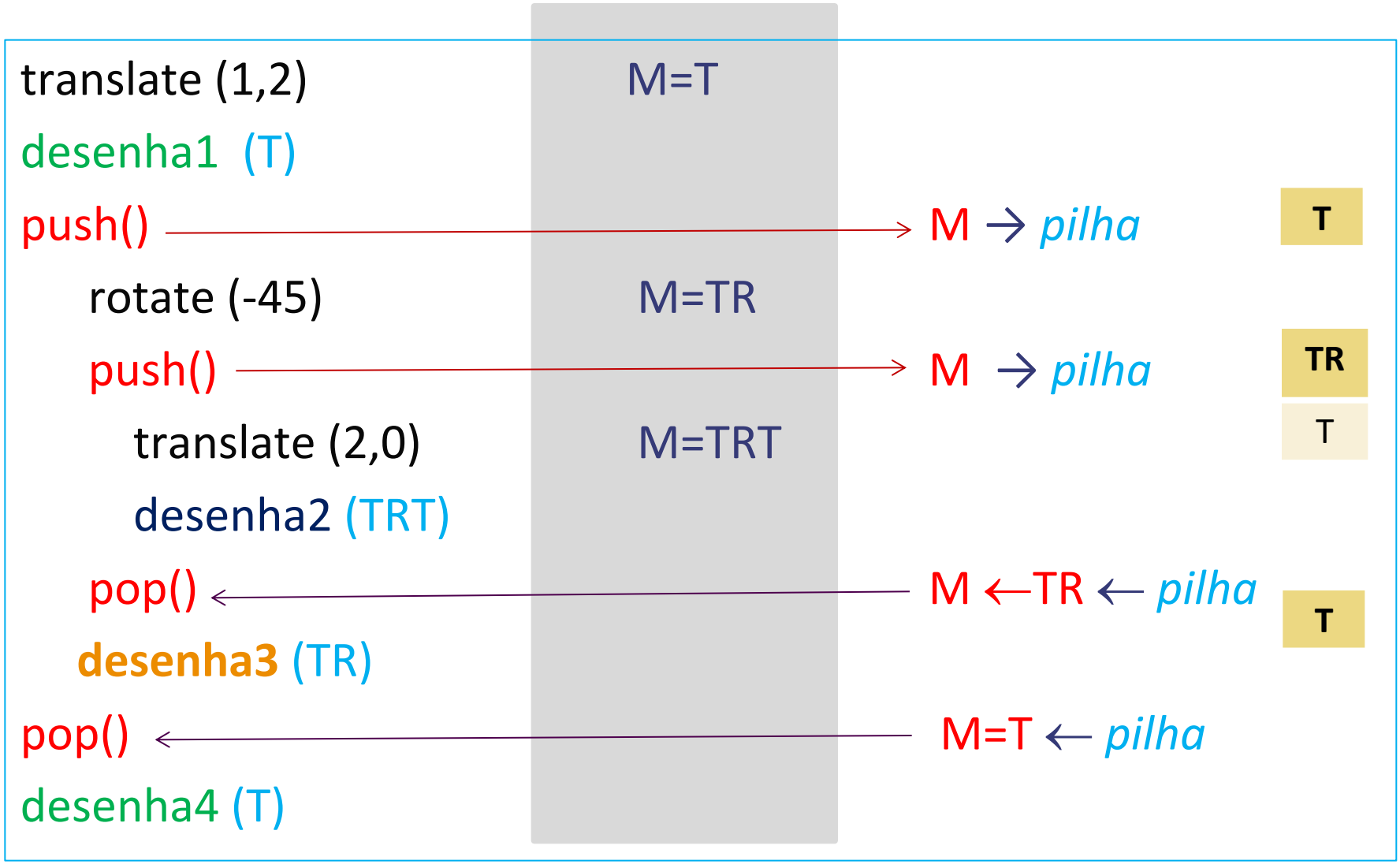
Ordem operações

desenha1	M=I	
push()		M → *pilha = I*
rotate (-45)	M=R	
translate (2,0)	M=RT	
desenha2		
pop()		M=I ← *pilha*
desenha3		

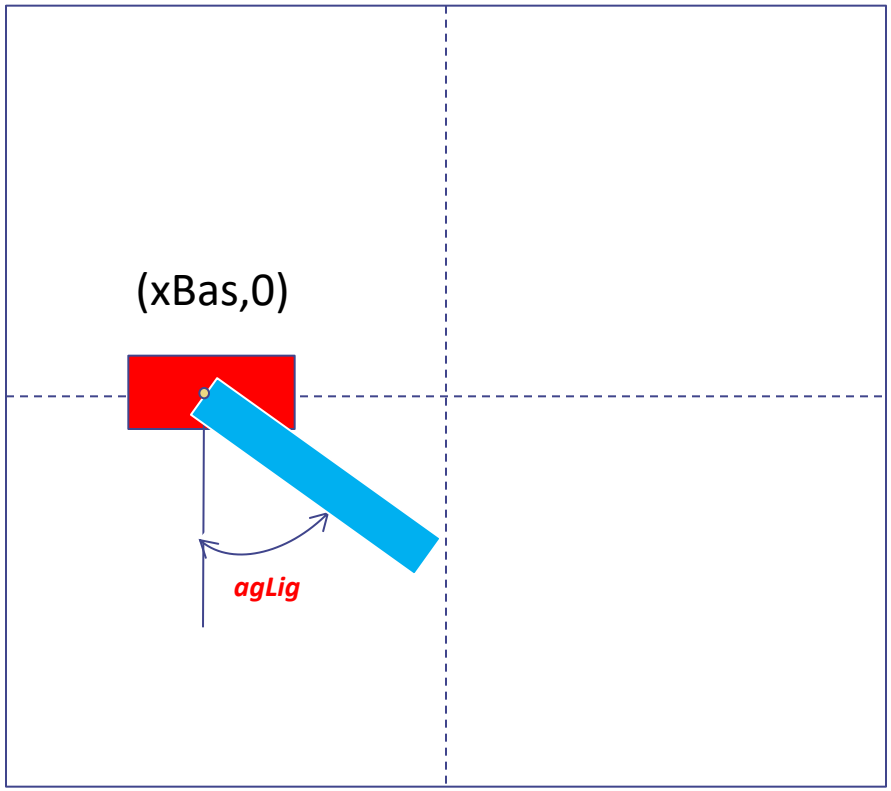
desenha2 = 1ºT + 2ºR

desenha1=desenha3 !!!

Ordem operações



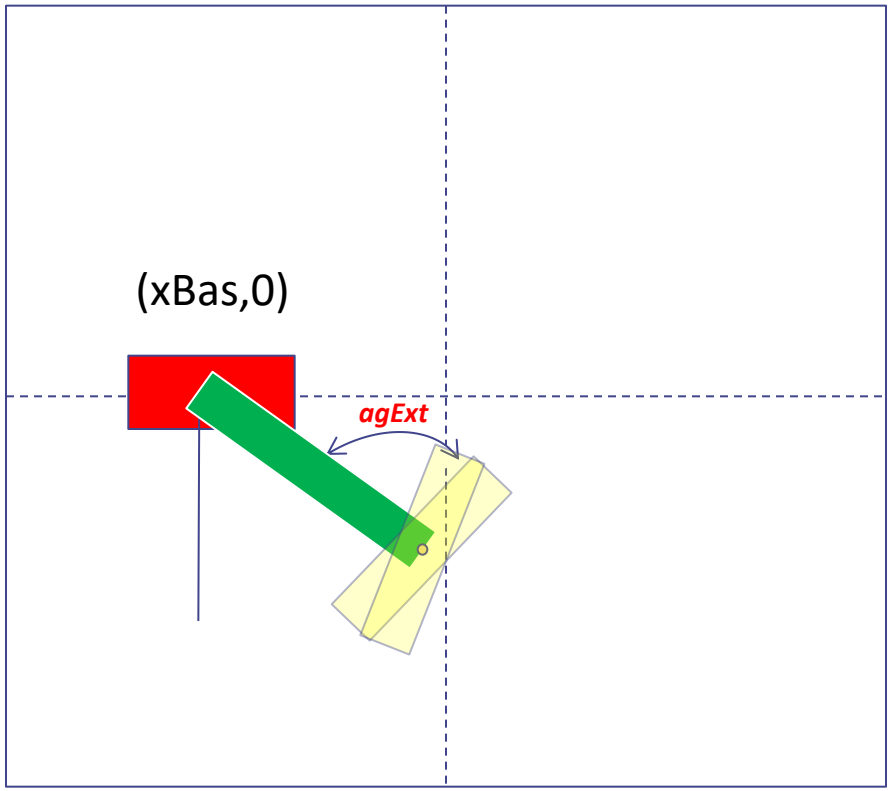
3. ROBOT: composição de operações




```
Translate(xBas,0)    // Comum a tudo
Push
   Scale(Base)
    desenhaBase
Pop

>> O que vai ser comum à Ligação e
à extremidade ?
push
    >> O que é especifico à Ligação  ?
    desenhaLigacao
push
```

3. Composição de operações



E a extremidade ?
Como aproveitar as operações da base e da ligação ?

- De novo em relação aos trabalhos anteriores
  - A GLUT considera o tratamento de "***teclas***" e "***teclas especiais***" de forma independente

```
main {
        …
           glutKeyboardFunc(teclado);
           glutSpecialFunc(teclasNotAscii);
}
```

```
void teclasNotAscii (int key, int x, int y)
{
           if(key == GLUT_KEY_LEFT) {
               xSup= xSup- incSup;
               glutPostRedisplay();
           }
           if(key == GLUT_KEY_RIGHT) {
               xSup= xSup+ incSup;
               glutPostRedisplay();
           }
…
}
```

openGL

2. Apanhar os objectos = *coordenadas dos objectos*

- Como saber se:
  - quadrado ***embate*** na base ?
  - quadrado ***apanhado*** pela extremidade ?

- Extremidade
  - Como saber "*aonde anda*" a extremidade ??

openGL

2. Coordenadas dos objectos

- **GL_MODELVIEW_MATRIX**
  - **Matriz que integra: "Modelos" + "Visualização"**

- GLfloat Matriz[4][4];
- glGetFloatv(GL_MODELVIEW_MATRIX, &Matriz[0][0]);

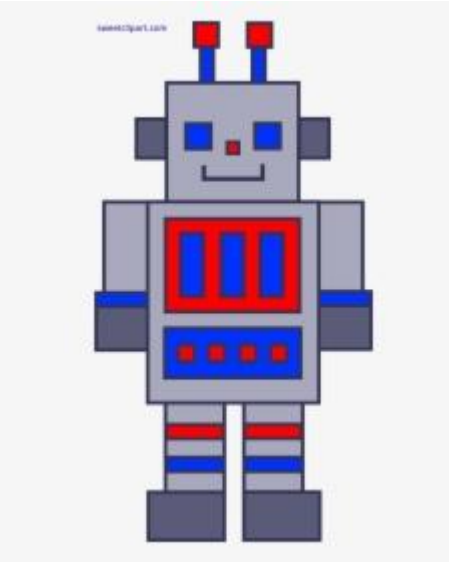
$$M = \begin{bmatrix} * & * & * & tx \\ * & * & * & ty \\ * & * & * & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad M = \begin{bmatrix} * & * & * & 0 \\ * & * & * & 0 \\ * & * & * & 0 \\ tx & ty & tz & 1 \end{bmatrix}$$

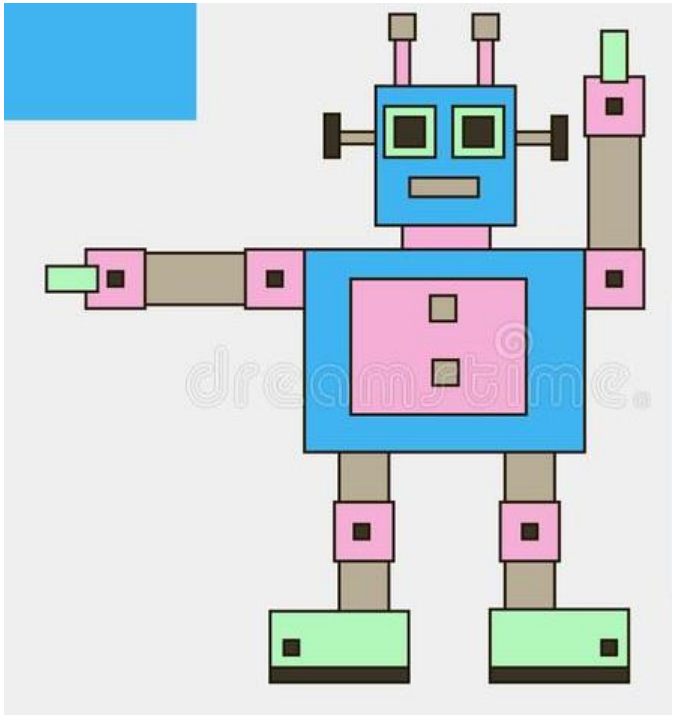
Em openGL matriz modelView

- ***Transposta***
- *Normalizada* ***[-1,1]***

- **tx=M[3][0]**
- **ty=M[3][1]**

- Melhoramentos do trabalho
  - **Modelo hierárquico !**





PROJETO

Exemplo do que vai ser a primeira meta –

Objecto 3D – próxima aula

- **Geometria -** Definir um objecto -
  - Exemplo mesa
  - Requisitos: um tampo e 4 pernas

- **Animação – Transformações geométricas**
  - Requisitos: duas rotações e uma translação
  - Ex: uma gaveta que abre (translação)
  - Ex: uma porta que abre (rotação)