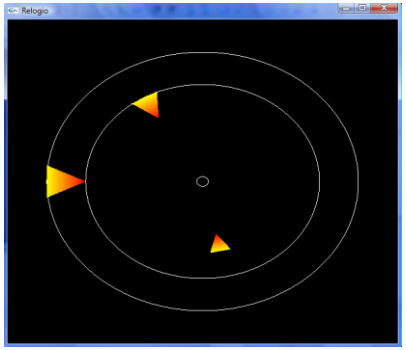
1º openGL
Relógio



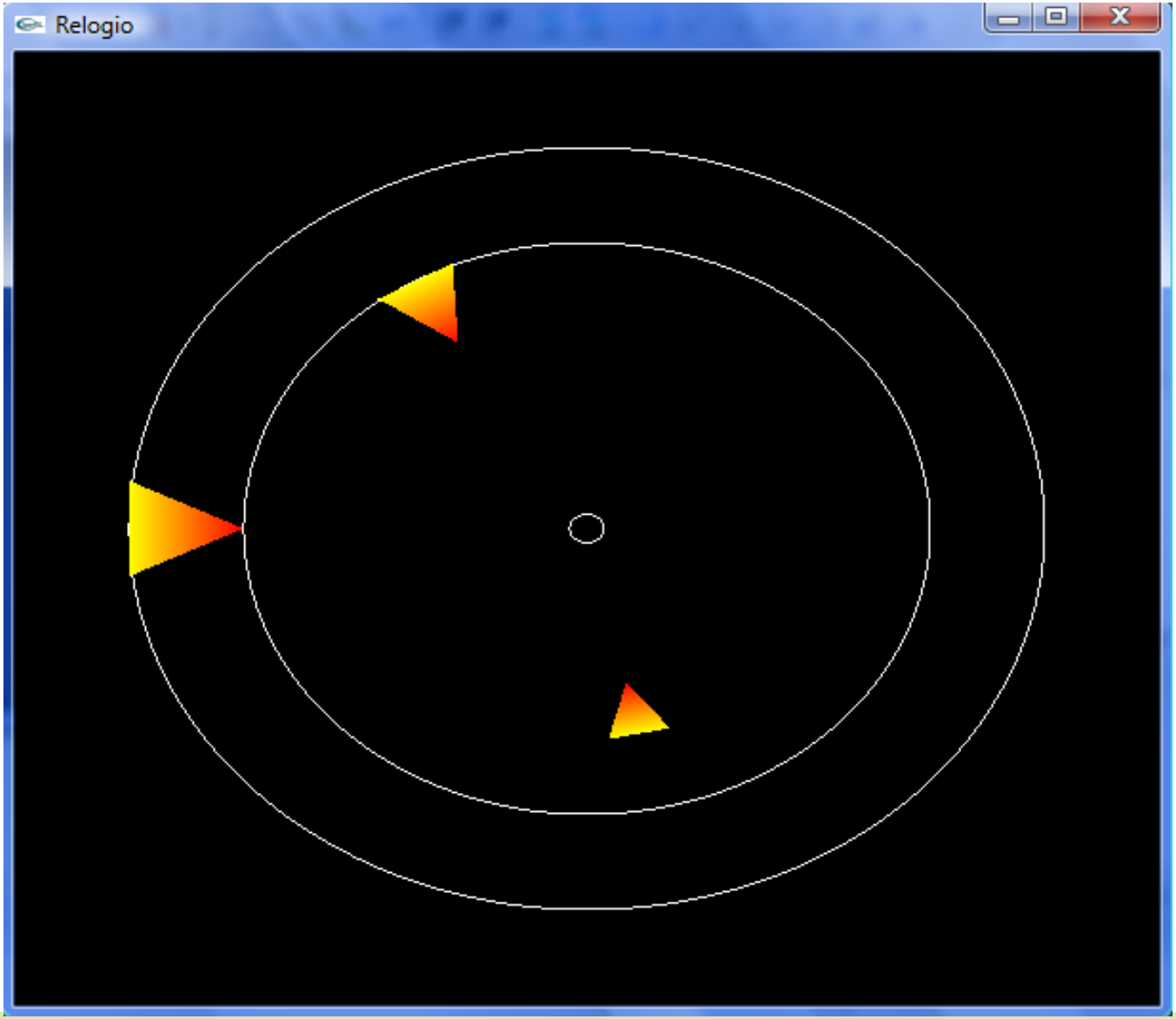
2019/20 - Computação Gráfica

Jorge Henriques
Evgheni Polisciuc

Objectivos

- Este trabalho tem como objectivos consolidar:
  - Utilização de variáveis e comandos em OpenGL
  - Desenho de primitivas básicas
  - Especificação de cores
- Pretende-se ainda introduzir como o OpenGL permite implementar operações básicas de transformação de objectos 2D, em particular:
  - Translação
  - Rotação
  - Escala

1º exemplo



Objectivos

O que é preciso para saber as horas, minutos e segundos ?

```
#include <time.h>
struct tm timeinfo;
GLint hour, minute, second;

{
…
    time_t t = time(0);           //time now
    time(&t);
    localtime_s(&timeinfo, &t);
    hour = timeinfo.tm_hour;
    minute = timeinfo.tm_min;
    second = timeinfo.tm_sec;
```

**código disponibilizado no código base*

Objectivos

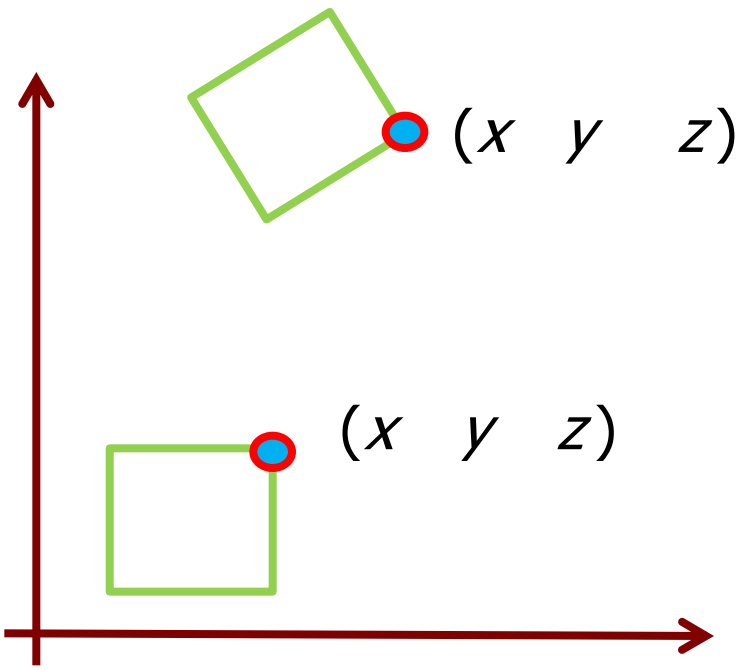
Para fazer continuamente a actualização da janela ?

- Actualização de msecDelay em msecDelay

```
main (..)
{
      glutTimerFunc( msecDelay, Timer, 1);     //.. função callback
 }


void Timer(int value)
{
    glutPostRedisplay();
    glutTimerFunc( msecDelay ,Timer, 1);   //.. Espera msecDelay milisegundos
}
```

Transformações geométricas

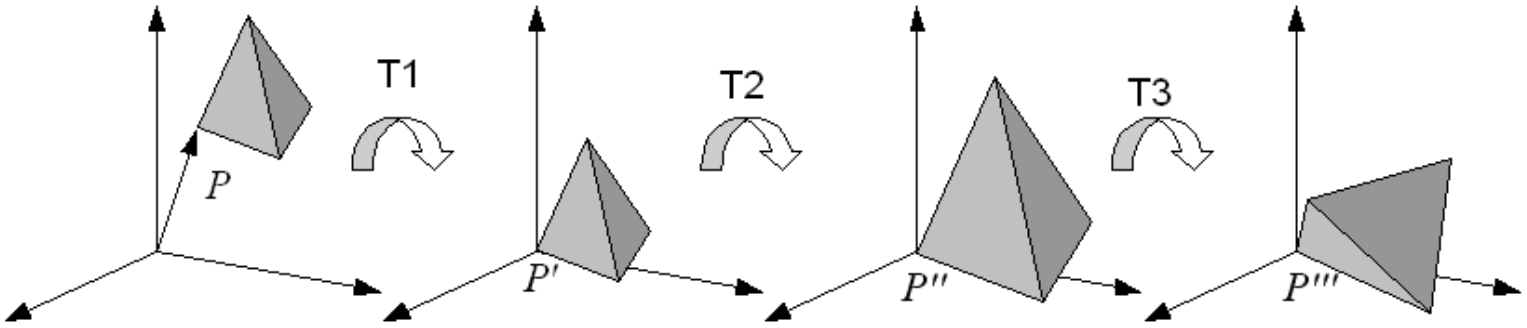
- Ideia geral
- Coordenadas homogéneas



$$\begin{bmatrix} \tilde{x} & * & * & * & * \\ \tilde{y} & * & * & * & * \\ \tilde{z} & * & * & * & * \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Ordem de operações

- Aplicação de uma transformação T1 a um ponto P
  - P’ = T1 P
- Aplicação de uma transformação T2 a um ponto P’
  - P’’ = T2 P’
- Aplicação de uma transformação T3 a um ponto P’’
  - P’’’ = T3 P’’



Ordem operações

- P''' = T3 P'' = T3 T2 P' = T3 T2 T1 P
  - **P''' = M P**
  - **M= T3xT2xT1**

- Composição de Transformações = Produto de Matrizes
  - **Não é comutativa** (o produto de matrizes também não é comutativo)
  - Transformações associam-se da direita para a esquerda (**ordem inversa**).

- ***Eficiência***: aplica-se uma transformação composta (matriz M). em vez de várias transformações sucessivas …

Ordem / matriz acumulação =modelView=M

Rectângulo – coordenadas iniciais P0?

```
desenha 1
rotate(45)
desenha 2
translate(2,0)
desenha 3
```

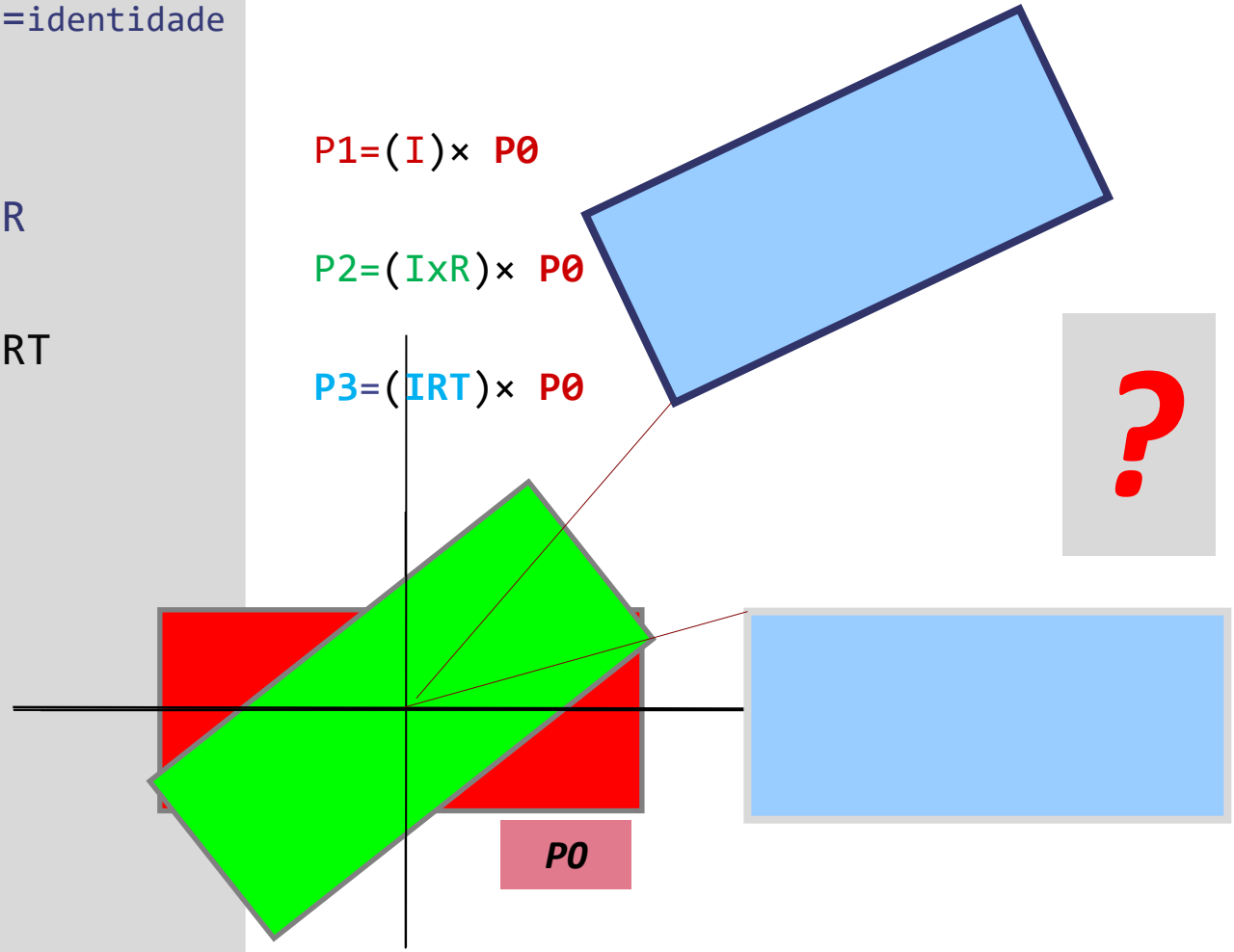
Ordem ?

```
M=I=identidade

M=IR

M=IRT
```

P1=(I)× **P0**

P2=(IxR)× **P0**

P3=(IRT)× **P0**



Ordem operações

1. Ordem de operações

- Em openGL é ao contrário “do que pensamos” !!

2. Uma transformação é aplicada a todos os objectos !

- Como ultrapassar o problema?

- Para evitar este efeito cumulativo, e permitir que uma transformação tenha efeito apenas num determinado espaço do programa, usa-se:
  - **glpushMatrix**()
  - ***glpopMatrix***()
- Permite guardar e restaurar o valor da matriz de transformação
- Ideal para isolar e transformar apenas alguns objectos

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

translate (1,2)	M=T	
desenha1 (T)		
push()		M → *pilha = T*
rotate (-45)	M=RT	
push()		R → *pilha* = RT | T
translate (2,0)	M=TRT	
desenha3 (TRT)		
pop()		M=RT ← *pilha = T*
desenha4 (RT)		
pop()		M=T ← *pilha*
desenha3 (T)		

Objectivos

Operações: ver apresentação openGL

- *Ordem das operações é importante !*
- *Operações cumulativas: push e pop*

Entre cada primitiva fazer push/pop (veremos qual o significado!!)

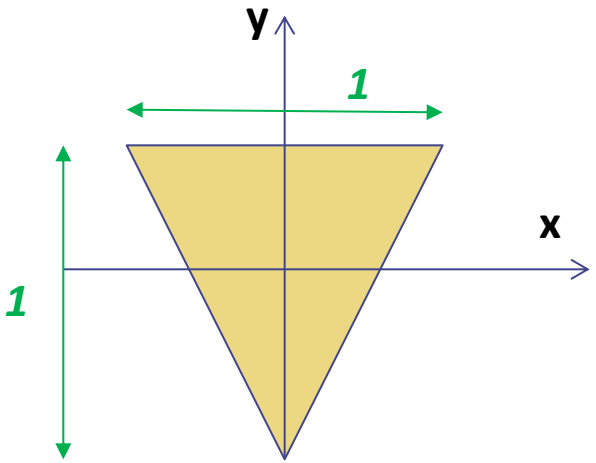
Qualquer coisa como (2D) …

```
glPushMatrix();
   glTranslatef(   posicaoX,  posicaoY,     0);     //.. Plano XY
   glRotatef( anguloGraus, 0, 0, 1);                //.. Roda em torno do eixo dos Z
   glScalef( scaleX, scaleY scaleZ );               //.. Aumento/diminuição do tamanho

   DesenhaTriangulo( ..);
glPopMatrix();
```

Rotação positiva

1º exemplo

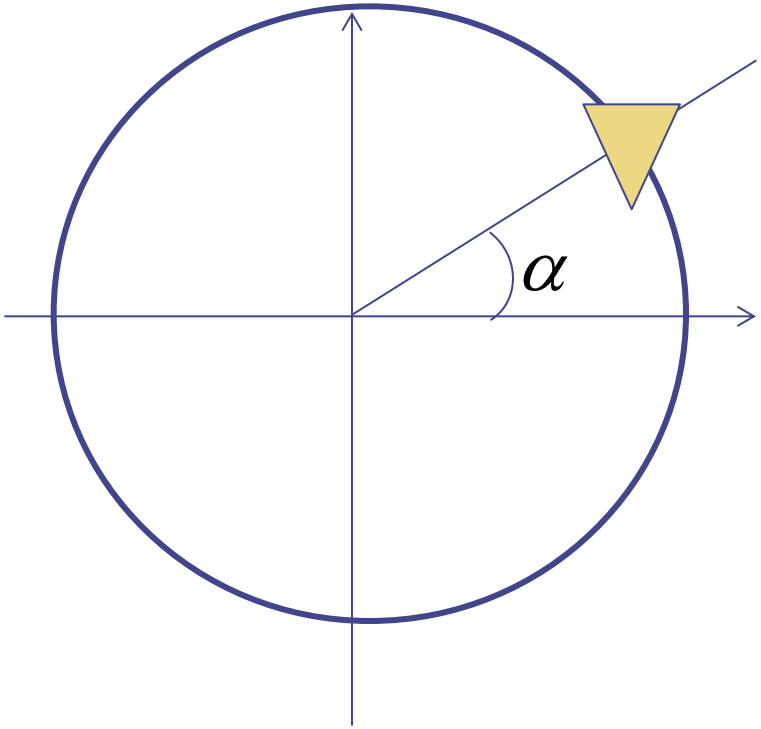
- ***Ponteiro dos segundos ???***

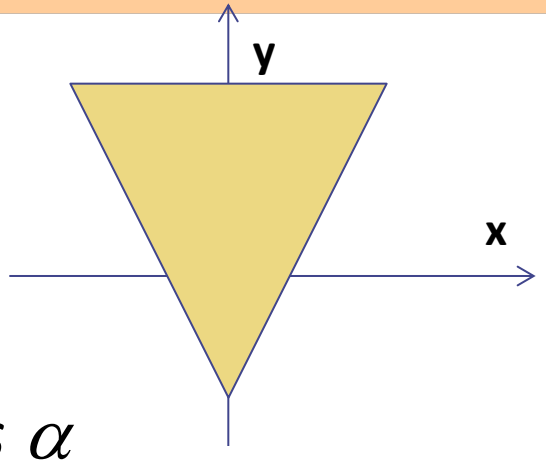


- *1. Desenhar **na origem** um triângulo (**normalizado**, i.e. tamanho unitário) !!!*
- *2. Posição / localização ?*
- *3. Rotação / orientação?*
- *4. Escala / tamanho ?*

1º exemplo

- *2. Posição*





$x = r \cos \alpha$

$y = r \sin \alpha$

segundos =0	→	α=90
segundos =60	→	α= -270

$\alpha = 90 - 6 \text{ sec}$

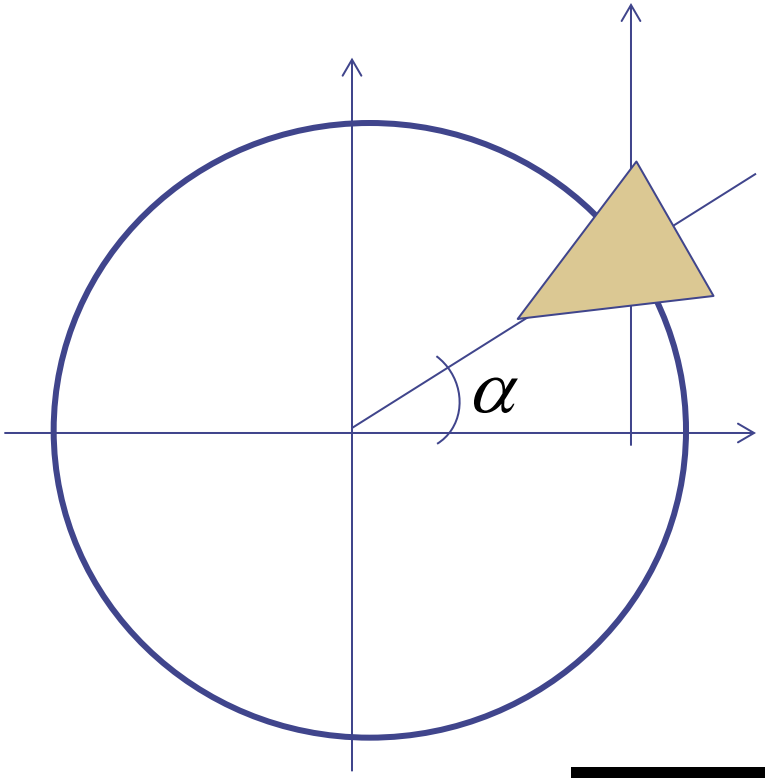
glTranslate(tx, ty, tz)

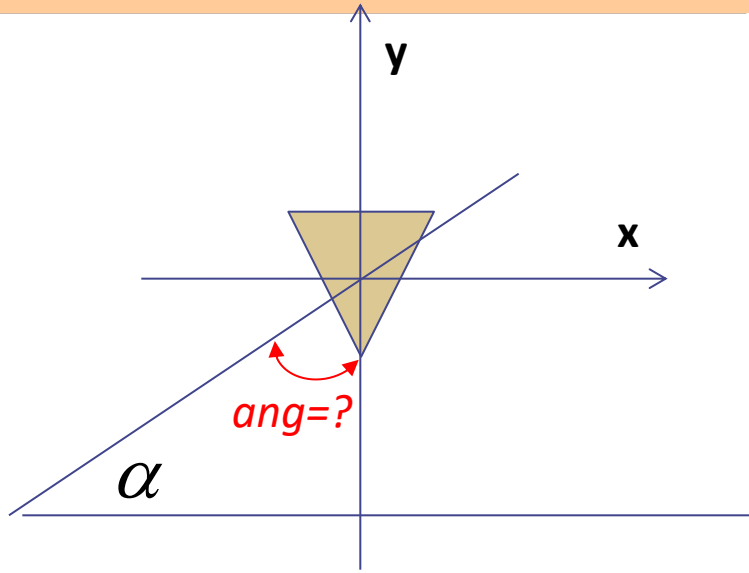
glTranslate(x, y, 0)

Segundos

```
second = current_time->tm_sec;

angulo= 90-6.0*second;              // graus
raio=5;
x =raio*cos(PI*angulo/180.0)        // radianos
y =raio*sin(PI*angulo/180.0)        // radianos

glPushMatrix();
      glTranslatef( x, y, 0);
      DesenhaTriangulo(10);
glPopMatrix();
```

3. Rotação





$ang = \alpha - 90$

glRotate(angulo,, eixo)

glRotate(ang, 0,0,1)

Segundos

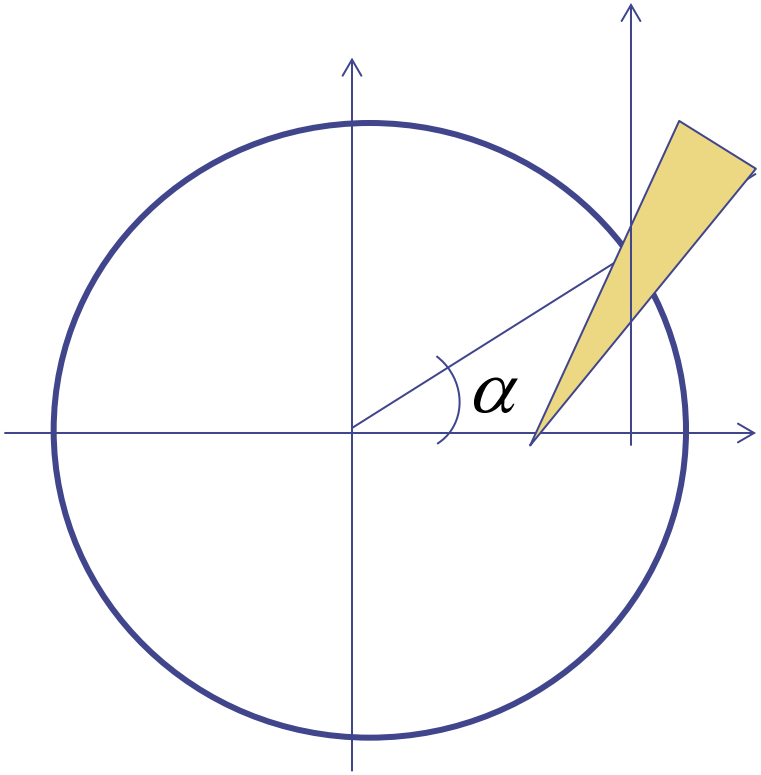
```
second = current_time->tm_sec;

angulo= 90-6.0*second;
raio=5;
x =raio*cos(PI*angulo/180.0)
y =raio*sin(PI*angulo/180.0)

glPushMatrix();
        glTranslatef( x, y, 0);
        glRotatef( angulo-90.0, 0, 0, 1);
        DesenhaTriangulo(10);
glPopMatrix();
```

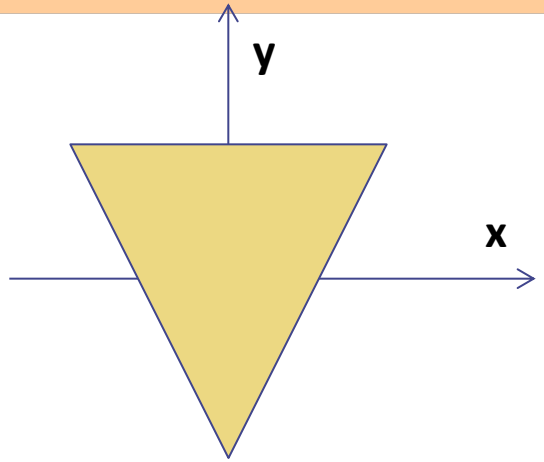
Ordem é importante

TR

1º exemplo

- *4. Escala*





Por exemplo: aumentar o comprimento do ponteiro

glScale(sx, sy, sz)

glScale(1, 3, 1)

Segundos

```
second = current_time->tm_sec;

angulo= 90-6.0*second;
raio=5;
x =raio*cos(PI*angulo/180.0)
y =raio*sin(PI*angulo/180.0)

glPushMatrix();
          glTranslatef( x, y, 0);
          glRotatef( angulo-90.0, 0, 0, 1);
          glScalef( 1, 2 ,1 );
          DesenhaTriangulo(10);
glPopMatrix();
```

Ordem correta ?

1º exemplo

S

- A completar
  - Ponteiro dos segundos ✓
  - Ponteiro das minutos
  - Ponteiro das horas

- Melhoramentos / inovações
  - Área (circulo) interior
  - Relógio “mais bonito”