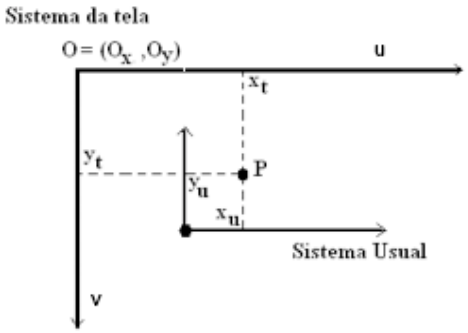
Exercícios (07/04/2010)

Considere, conforme os exercícios e códigos anteriores, a definição do tipo *ponto2D* e *vetor2D* em C:

```
typedef struct{
          float x;
          float y;
} ponto2D

typedef ponto2D vetor2D;
```

1. Implemente a função ***ConverteCoord*** que recebe um ponto ***P*** em coordenadas (x_u,y_u) do sistema de coordenadas cartesianas usual e converte para coordenadas (x_t, y_t)) da tela. O usuário deve fornecer também na entrada da função a origem ***O*** = (O_x, O_y) da tela e os vetores da base do sistema de coordenadas da tela $\{u,v\}$, da seguinte maneira:
   `ponto2D ConverteCoord (ponto2D p, ponto2D O, vetor2D u, vetor 2D v);`



2. Chamamos de curva poligonal à curva composta por uma seqüência de pontos ligados por segmentos de reta. Implemente a função ***DesenhaElipse*** para desenhar, usando uma curva poligonal, uma elipse de raios ***rx*** e ***ry***, centrada na origem. A função deve receber como parâmetro, além dos raios, o número de pontos da curva poligonal***:***
   `void DesenhaElipse (float rx, float ry, int num_pontos);`

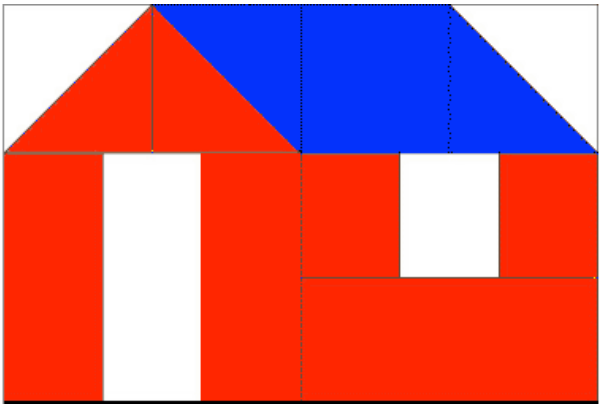
   Dicas:
   *1 - a representação paramétrica da elipse é f(θ) = (rx*cos(θ), ry*sen(θ))*
   2 – use o iterador ***for****.*

3. Considere a função **DesenhaQuadrado** e **DesenhaTriangulo** abaixo:

```
void DesenhaQuadrado()
{
     glBegin(GL_QUADS);
          glVertex2f(-1.0,-1.0);
          glVertex2f(1.0,-1.0);
          glVertex2f(1.0,1.0);
          glVertex2f(-1.0,1.0);
     glEnd();
}


void DesenhaTriangulo()
{
     glBegin(GL_TRIANGLES);
          glVertex2f(0.0,0.0);
          glVertex2f(1.0,0.0);
          glVertex2f(0.0,1.0);
     glEnd();
}
```

Usando apenas as funções glPushMatrix(), glPopMatrix(), glTranslatef(), glRotatef(), glScalef(), além das funções DesenhaQuadrado() e DesenhaTriangulo(), implemente a função display() para obter um resultado semelhante à figura abaixo:



Obs.: não é necessário desenhar as listras pretas.