COMPUTACIÓN GRÁFICA
Primer Parcial- 27 de abril de 2011

Resuelva cada problema en una hoja distinta. Escriba en cada hoja a entregar su nombre y apellido.

1. Implemente en C# el procedimiento ***setupViewport(...)*** , de modo que la Figura1 se vea completa y sin deformarse, aún cambiando el tamaño de la ventana de pantalla. Las llamadas a este procedimiento pueden verse en el código adjunto.

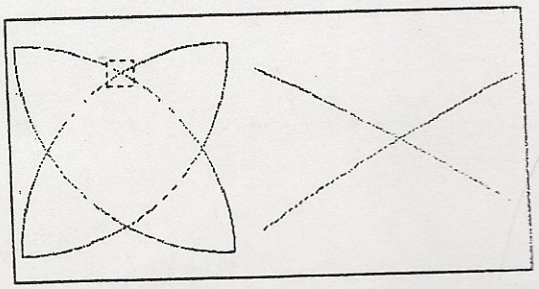


Figura 1

2. a) Implemente un método para suavizar normales, basado en el ángulo que conforman las normales que convergen en un vértice.

   b) Escriba las tablas resultantes de aplicar el método del inciso a) a la estructura de caras de la tabla, que representa al objeto de la Fig 2, para que una vez iluminado y sombreado, se vea como un cilindro con tapas. No tiene que efectuar cálculos, sino dejar las operaciones indicadas.

CARAS												
Ind_vértices						Ind_normales						
0	1	7	6			0	0	0	0			
1	2	8	7			1	1	1	1			
2	3	9	8			2	2	2	2			
3	4	10	9			3	3	3	3			
4	5	11	10			4	4	4	4			
5	0	6	11			5	5	5	5			
0	5	4	3	2	1	6	6	6	6	6	6	
6	7	8	9	10	11	7	7	7	7	7	7	

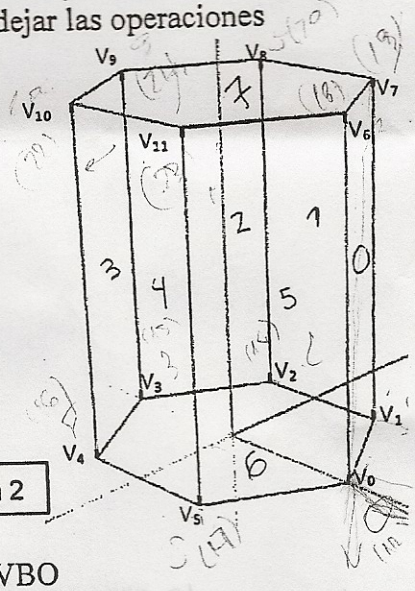


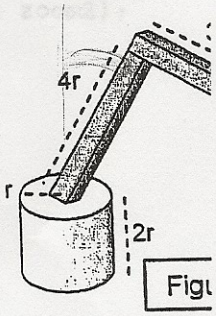
Figura 2

3. Se desea **dibujar** el cilindro con tapas del ejercicio anterior usando VBO

   a) Escriba la primitiva OpenTK –con sus parámetros- que usaría para ese fin.

   b) Escriba las tablas necesarias para crear ese VBO. Las tablas deben mostrar, al menos, las dos tapas y la primera y última caras laterales

4. Implemente el *Constructor* y el método *Dibujar* (...) de la clase *BrazoRobot* que muestre el brazo de robot de la Fig. 3. Tenga en cuenta que debe parametrizarse de modo tal que permita la rotación de las articulaciones sujeta, en cada caso, a las restricciones que considere convenientes.
   Dispone para ello de las clases *Cilindro* y *PoliedroRectangular*, con sus constructores: *Cilindro*(radio, altura) y *PoliedroRectangular*(anchoBase, profBase, largo). Los métodos *Dibujar*( ) de ambas clases dibujan a los objetos alineados con el eje *y*, con sus baricentros en el origen.



Figu

```
            miFigura = new Figura(origenX, origenY, ang1, ang2 , radio);

private void glControl1_Paint(object sender, PaintEventArgs e)
        {
            GL.Clear(ClearBufferMask.ColorBufferBit |
            ClearBufferMask.DepthBufferBit);

            setupVentana(  ,  ,  ,  );

            int w = glControl1.Width;
            int h = glControl1.Height;
            setupViewport(0, w/2, 0, h);

            GL.MatrixMode(MatrixMode.Modelview);
            GL.LoadIdentity();
            miFigura.dibujar();

            if (zoom)
            {
             areaZoom = new Cuadrado(  ,  ,  ,  );
            GL.Color3(0, 0, 0.7);
            GL.Enable(EnableCap.LineStipple);
            GL.LineStipple(5, 0x5555);

            areaZoom.dibujar();
            GL.Disable(EnableCap.LineStipple);

            setupVentana(  ,  ,  ,  );

            setupViewport(w / 2, w, 0, h);
            GL.MatrixMode(MatrixMode.Modelview);
            GL.LoadIdentity();
            miFigura.dibujar();

            }

            glControl1.SwapBuffers();
        }

        private void glControl1_MouseDown(object sender, MouseEventArgs e)
        {
            double zoomX;
            double zoomY;
            double zoomZ;

        Glu.gluUnProject(  ,  ,  ,  ,  ,  , out zoomX, out zoomY, out zoomZ);

        zoom = true;

        }
```