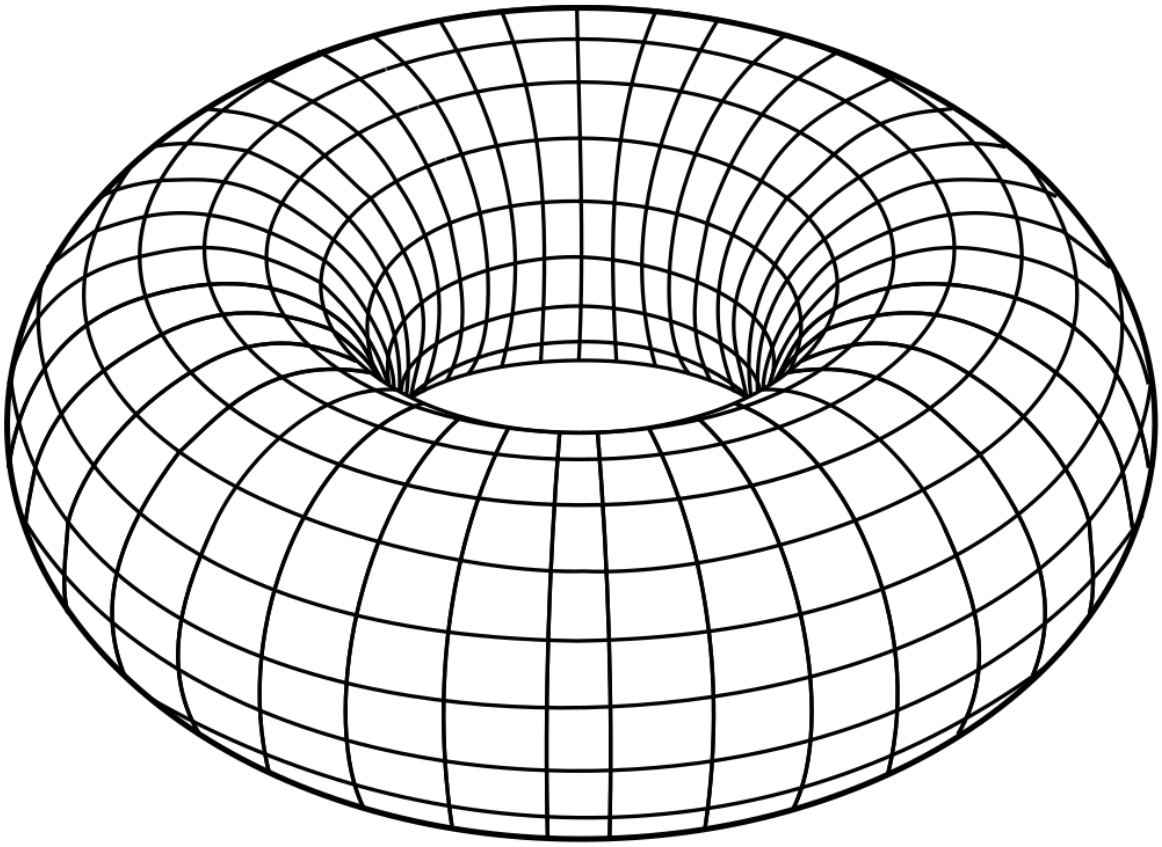
Universidad Nacional Autónoma de México

Primitiva Geométrica: Toroide

Alumno: Alfonso Murrieta Villegas

Análisis similar al hecho en clase para el dibujo de la envolvente del toroide con ciclos for y también pseudocódigo para dibujar en OpenGL los triángulos

Un Toroide es la superficie de revolución generada por un polígono o una curva cerrada simple que gira alrededor de una recta exterior o eje de rotación con la que no se intercepta.



Para poder aterrizar o realizar el toroide anterior a través de OpenGL primero es necesario contemplar las funciones paramétricas asociadas a esta figura, las cuales son las siguientes:

$$\begin{cases} x = (R + r\cos\ \alpha)\cos\ \beta \\ y = (R + r\cos\ \alpha)\sin\ \beta \\ z = r\sin\ \alpha \end{cases}$$

Tal que

R = Es el radio que va del dentro del toroide y a centro del "cilindro" o toro revolucionado.

r = Es el radio del conducto del toroide, es decir el radio de la circunferencia revolucionada para generar el toro

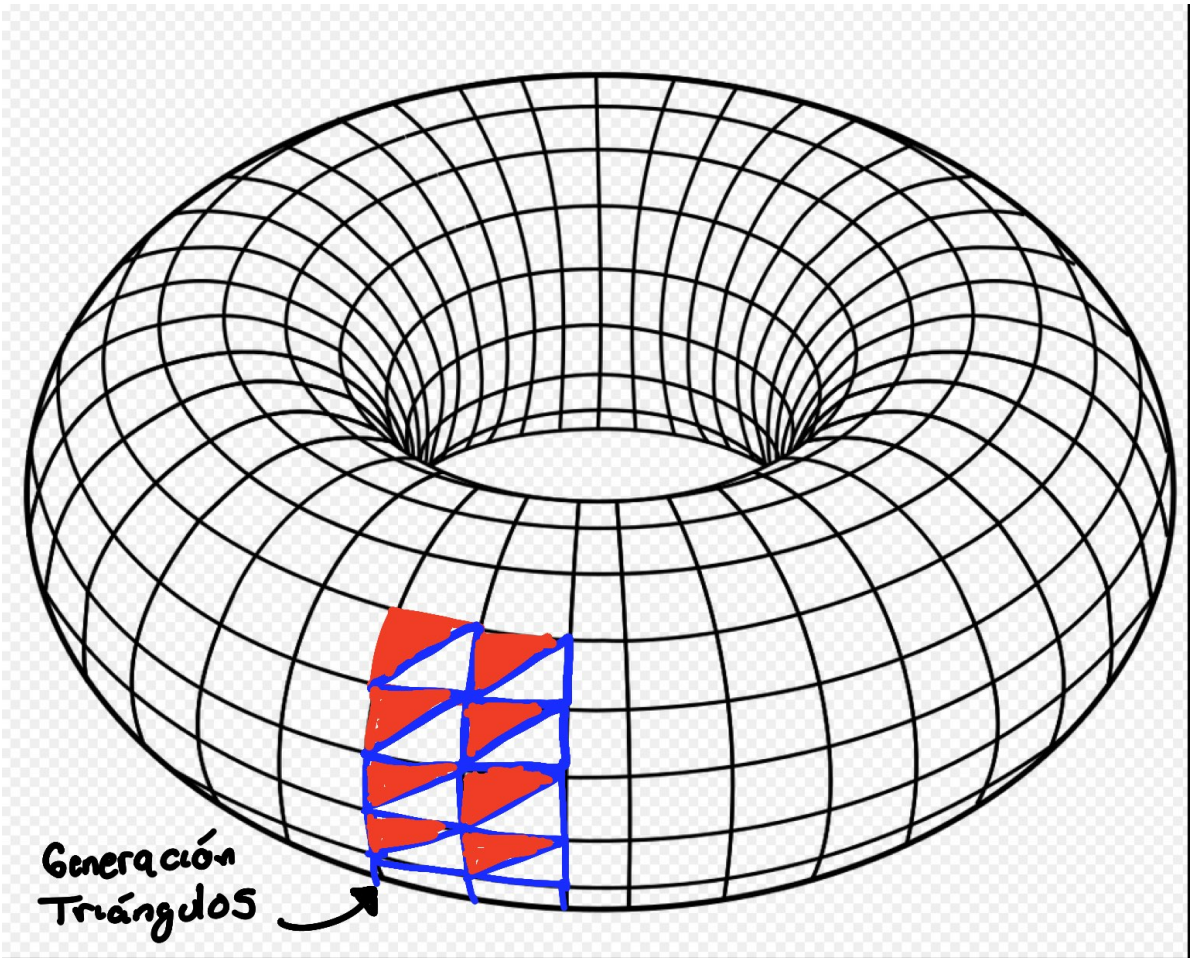
Generación de Pseudo-código para el cilindro

Con base a las variables o entradas anteriores, y considerando que cualquier figura geométrica esférica debe tener meridianos y paralelos, es como se partió y generó el siguiente pseudo-código:

```
// a y b son los ángulos esféricosy que cubren toda la superficie
//ia e ib son indices para el manejo correspondiente de cada meridiano y
paralelos

for ( ib = 0; ib < PARALELOS; ib++)
    for ( ia = 0; ia < MERIDIANOS; ia++){

        //Generación del cilindro

        //Basados en las paramétricas
        x = (10+cos(b))*cos(a);
        y = (10+cos(b))*sin(a);
        z = sin(b);

        //Sería la parte encargada de la posición
        //de cada indice en sus distintas coordenadas
        toroide_position[ix + 0] = x * r;
        toroide_position[ix + 1] = y * r;
        toroide_position[ix + 2] = z * r;
    }
}
```

Generación de Pseudo-código para los triángulo

Como bien sabemos los triángulos son la base para la generación de entidades o primitivas, en este caso concreto podemos asimilar de la siguiente forma la generación de triángulos para la superficie de nuestro toroide:



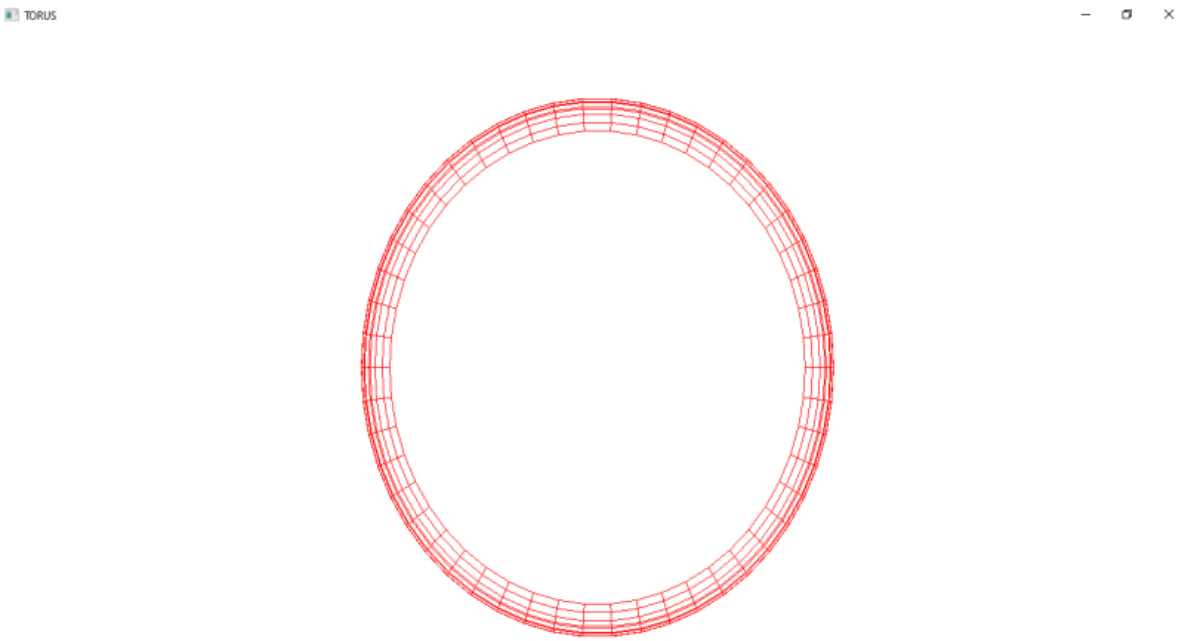
Con base a los datos o variables considerados en la generación de cada uno de los puntos de la superficie del toroide, a continuación se plantea el pseudo-código para la generación de triángulos y posteriormente rectángulos con la finalidad de crear una superficie:

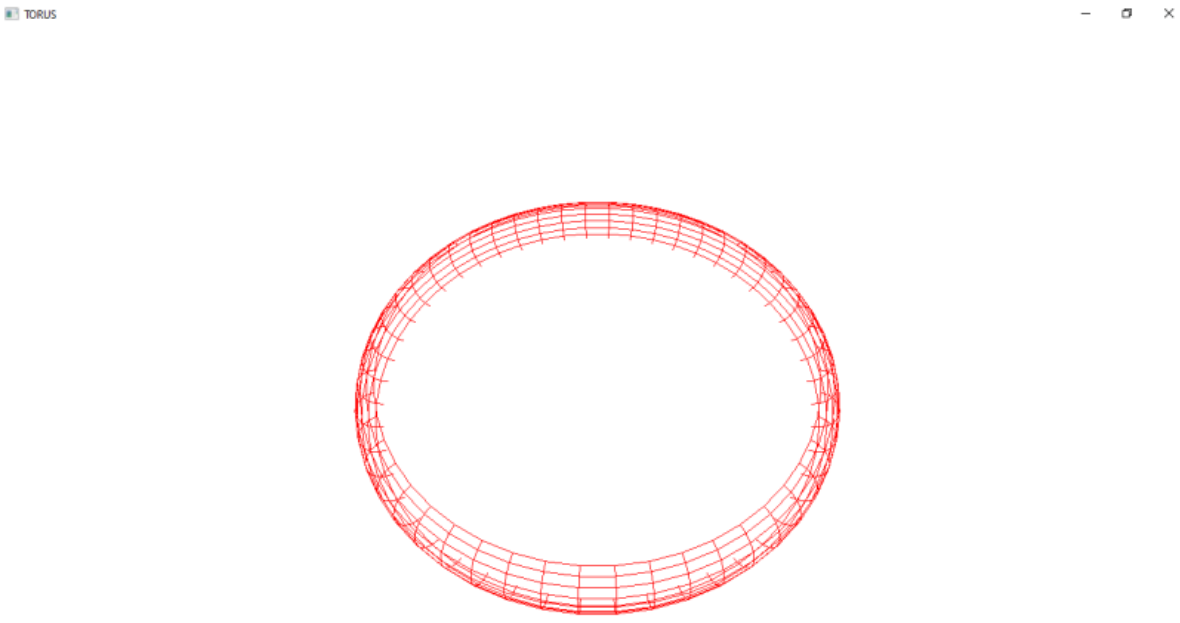
```
//NOTAS:
//ix e iy son los indices de cada una de las coordenadas de nuestras
circunferencias

for (ib = 1; ib < PARALELOS; ib++){

    for (ia = 1; ia < MERIDIANOS; ia++, iy++){
        //Incrementamos el indice 'y' cada vez que entre en el for esto para ir
        //desplazandonos respecto al dibujo de cada triángulo

        // Primera mitad de los rectángulos
        toroide_index[ix] = iy;
        ix++;
        toroide_index[ix] = iy + 1;
        ix++;
        toroide_index[ix] = iy + MERIDIANOS;
        ix++;

        // Segunda mitad de los rectángulos
        toroide_index[ix] = iy + MERIDIANOS;
        ix++;
        toroide_index[ix] = iy + 1;
        ix++;
```

```
            toroide_index[ix] = iy + MERIDIANOS + 1;
            ix++;
        }
    }
```

Primer acercamiento mediante OpenGL

A continuación se muestra el resultado obtenido mediante OpenGL con base al pseudo-código previamente realizado:





NOTA: Se puede observar que por el momento sólo se tiene aterrizado la mitad del cilindro revolucionado o toro.

Referencias

Norm Prokup. *Parametric Torus*. GeoGebra. Recuperado el 11 de Noviembre de 2020, de https://www.geogebra.org/m/hbYvw5Sp

Megabyte Softworks. C++, *OpenGL and more...* .Recuperado el 11 de Noviembre de 2020, de http://www.mbsoftworks.sk/tutorials/opengl4/011-indexed-rendering-torus/