KnE Engineering

SIIPRIN-CITEGC
Ibero-American Symposium on Computer Programming jointly held with the International Congress on Technology Education and Knowledge Management
Volume 2018

Knowledge E
Engaging minds

Conference Paper

Mathematical Algorithms for the Texture and Lighting of 3D Surfaces in a Computer

Algoritmos Matemáticos para el texturizado e iluminación de superficies 3D en un Computador

Alonso Álvarez, Narcisa Salazar, and José Tinajero

Escuela Superior Politécnica de Chimborazo, Código Postal 060155, Riobamba, Chimborazo, Ecuador

Abstract

At present, the generation of 3D objects by computer has become a fundamental tool for the development of most sciences. Modeling a three-dimensional surface on a computer whose display is a two-dimensional graphic screen presents some challenges such as simulating the depth on the graphic screen. To overcome this drawback, the authors propose to use Vector Differential Analysis (Differential Geometry), since calculating the normal vector to the Surface eliminates hidden sections and differentiates external faces of internal faces to texturize differently. In the same way, taking advantage of the properties of the Vector Gradient, it is possible to simulate light intensities on the surfaces.

Resumen

En la actualidad la generación de objetos 3D por computador se ha convertido en una herramienta fundamental para el desarrollo de la mayoría de las ciencias. Modelar una superficie tridimensional en un computador cuyo despliegue es una pantalla grafica bidimensional presenta algunos desafíos como el de simular la profundidad en la pantalla gráfica. Para superar este inconveniente se propone utilizar el Análisis Diferencial Vectorial (Geometría Diferencial), ya que mediante el cálculo del vector normal a la superficie se puede eliminar secciones escondidas y diferenciar caras externas de caras internas para texturizar de diferente forma. De igual manera aprovechando las propiedades del Vector Gradiente se logra simular intensidades de luz sobre las superficies.

Keywords: Mathematics, algorithms, Surface, differential geometry, Vector Gradient

Palabras clave: Matemáticas, Algoritmos, Superficies, Geometría Diferencial, Vector Gradiente.

Corresponding Author:
Alonso Álvarez
aalvarez@espoch.edu.ec

Received: 4 December 2018
Accepted: 5 December 2018
Published: 27 December 2018

Publishing services provided by Knowledge E

Selection and Peer-review under the responsibility of the SIIPRIN-CITEGC Conference Committee.

1. Introduccion

En la actualidad la generación de objetos 3D por computador se ha convertido en una herramienta fundamental para el desarrollo de la mayoría de las ciencias, como es el caso de las ingenierías y la medicina, al igual que no se puede desconocer la importancia que representa en el cine [1], [2].

Una superficie en tres dimensiones se puede definir como el grafo cartesiano de una función continuamente diferenciable de dos variables [3],

$$z = F(x, y)$$

$$F : \Omega \subseteq R^2 \to R$$

o como el grafo de una función paramétrica, también llamada función vectorial

$$\alpha(t, h) = (x(t, h), y(t, h), z(t, h))$$

$$t \in [a, b];$$

$$h \in [c, d].$$

Un modelo matemático se puede considerar como la representación de cualquier fenómeno real utilizando estructuras matemáticas [4].

Modelar una superficie tridimensional (3D) en un computador cuyo despliegue es una pantalla grafica bidimensional (2D) presenta algunos desafíos: a) ¿Como puede ser desplegada la profundidad en una pantalla grafica?; b) ¿Cómo pueden ser identificadas y removidas las partes ocultas de la superficie?; c) ¿Cómo pueden contribuir la iluminación, el sombreado, la textura, y el color en la representación de la superficie? Quizá una de las mejores respuestas a estas interrogantes la encontramos en el Análisis Diferencial Vectorial [4].

Desde una perspectiva física, una superficie puede emitir luz por su propia emisión, como focos de luz, o reflejar luz de otras superficies que la iluminan. Algunas superficies pueden reflejar y emitir luz. El color que se ve en un punto de un objeto está determinado por las múltiples interacciones entre las fuentes de luz y superficies reflectivas [1]. Este es un proceso recursivo.

El problema consiste de dos aspectos:

1. Modelar las fuentes de luz en una escena.
2. Construir un modelo de reflexión que trate con las interacciones entre materiales y luz.

Para comprender el proceso de iluminación, se puede comenzar siguiendo los rayos de luz de un punto fuente, donde el observador ve solamente la luz que emite la fuente y que llega a los ojos; Probablemente a lo largo de complejos caminos y múltiples interacciones con objetos en la escena.

2. Desarrollo

2.1. Segmentación y Mallado

Para la segmentación y el mallado, se supone que Ω sea un rectángulo

$$\Omega = \{(x, y) \in R^2 / a \leq x \leq b, c \leq y \leq d\}$$

En caso de no ser así, se puede hacer una aproximación de Ω mediante una familia de rectángulos disjuntos.

Si la superficie está definida por *z=F(x,y)* con *(x,y)*$\epsilon\Omega$ y F continua y diferenciable. Sean *n, m* el número de particiones en el eje *x* y en el eje *y* respectivamente entonces, si

Para i=1 hasta n

Para j=1 hasta m

$$h_x = |b - a| / n$$

$$h_y = |d - c| / m$$

$$V_x[i, j] = a + i * h_x$$

$$V_y[i, j] = c + j * h_y$$

$$V_z[i, j] = F\left(V_x[i, j], V_y[i, j]\right)$$

Donde V_x, V_y, V_z son matrices a valores reales.

Ahora, la transformación 3D a 2D que se realiza físicamente en el sistema visual humano, se tiene que realizar mediante un proceso matemático en un sistema de Graficación por computadora. Este proceso se le conoce como Axonometrico [4].

$$Axonometria(x, y, z) \rightarrow (ax, ay)$$

$$\{ax = y - \rho * x * \cos(\alpha)$$

$$ay = z - \rho * x * \sin(\alpha)\}$$

$$\rho = 0.5; \alpha = \pi/4$$

Mediante este proceso se puede transformar

$$(V_x[i,j], V_y[i,j], V_z[i,j],) \rightarrow (A_x[i,j], A_y[i,j],)$$

Y finalmente se construye la aproximación de la superficie mediante el mallado

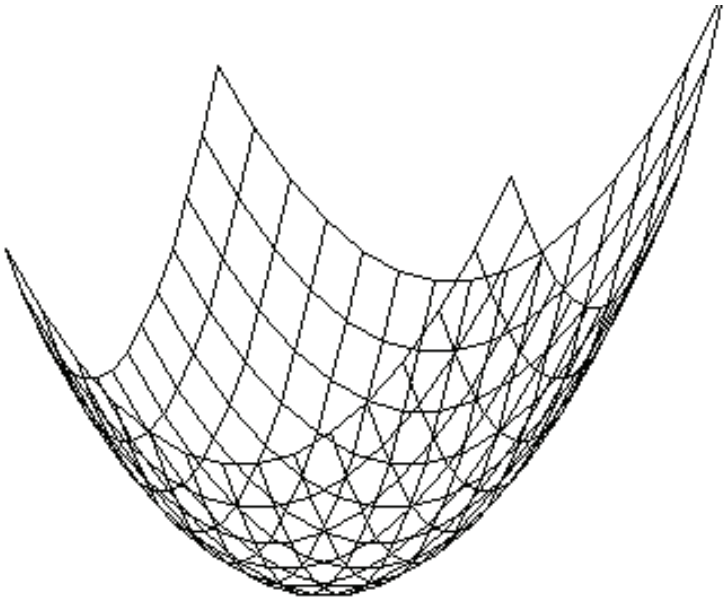


Figura 1: Paraboloide Mallado.

2.2. Remoción de Secciones Escondidas

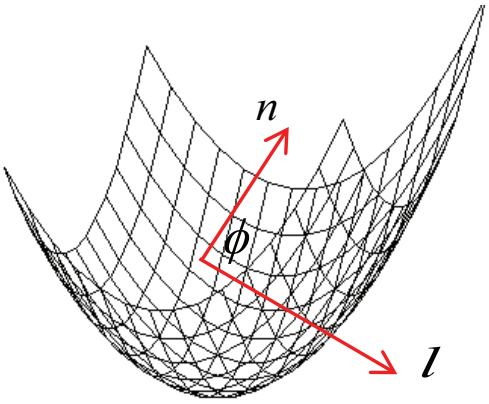
Uno de los problemas más desafiantes en la generación de objetos 3D, es la eliminación de secciones escondidas, con la finalidad de evitar cálculos de partes que no son visibles.

Un método para lograr esto es el algoritmo del pintor, el cual se basa en localizar las partes más cercanas al observador por lo tanto son las más visibles.

Otro algoritmo que se puede utilizar está basado en el Análisis Vectorial, el cual examina la dirección de los vectores normales a cada elemento del mallado. La prueba de visibilidad de un elemento del mallado se realiza al encontrar el ángulo entre el vector normal (**n**) y el vector de visión (punto del observador) (**l)** [1], [3],

$$\phi = \arccos\left(\frac{n \cdot l}{\|n\| \cdot \|l\|}\right)$$

Formula 1: Angulo entre el vector Normal y el Observador.



$$n = (n_1, n_2, n_3)$$
$$l = (l_1, l_2, l_3)$$
$$n.l = n_1 l_1 + n_2 l_2 + n_3 l_3$$
$$\|n\| = \sqrt{n_1^{\,2} + n_2^{\,2} + n_3^{\,2}}$$
$$\|l\| = \sqrt{l_1^{\,2} + l_2^{\,2} + l_3^{\,2}}$$

Si φ se encuentra entre o y 90 grados, la sección es visible y por lo tanto es desplegada caso contrario no.

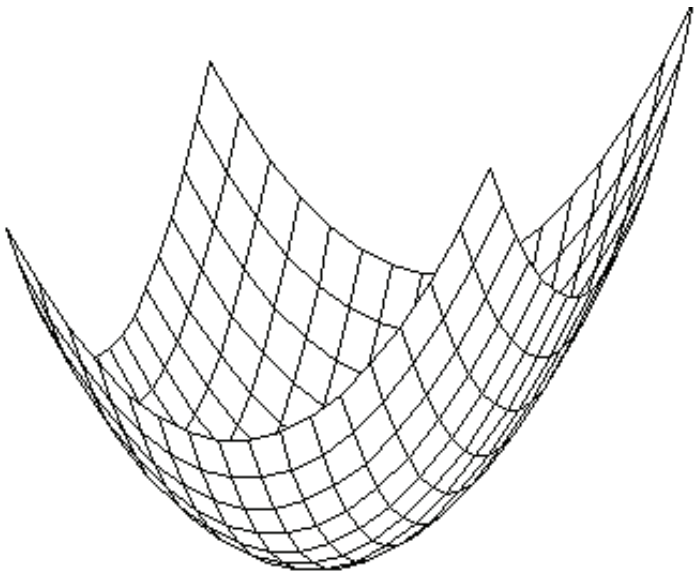


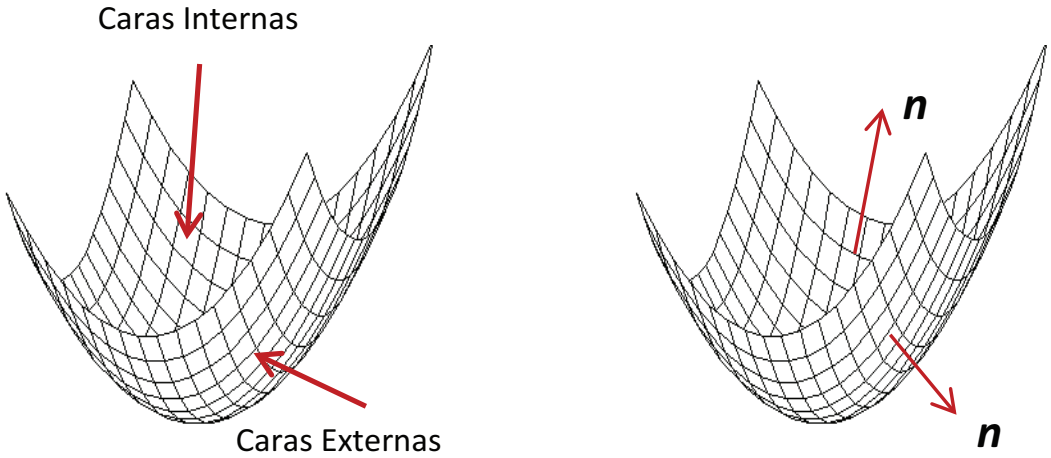
Figura 2: Paraboloide eliminando secciones escondidas.

2.3. Iluminación y texturizado

Finalizado el proceso de segmentación y mallado, se puede rellenar la malla utilizando algoritmos de modelos matemáticos para el sombreado, los cuales modelan los tipos de iluminación, dependiendo de las características de la superficie y de las propiedades de la iluminación que lo invade. Además, se debe considerar que, dentro de las caras visibles, se tiene caras externas y caras internas, por lo tanto, se necesita un procedimiento para diferenciar estas caras.

De igual forma se puede resolver este problema mediante el Análisis Vectorial. En cada punto de intersección de la malla se podrían lanzar vectores normales, y

dependiendo hacia donde apunta el vector se deduciría si es cara externa o interna y con lo cual se podrá texturizar o iluminar de manera diferente.



Este proceso se lo realiza mediante las fórmulas para encontrar el vector normal a la superficie [4],

Para i=1 hasta n

Para j=1 hasta m

$$a_x = \left(A_x[i+1, j] - A_x[i, j]\right)$$

$$n_x = a_x * \left(A_y[i+1, j+1] - A_y[i+1, j]\right)$$

$$a_y = \left(A_y[i+1, j] - A_y[i, j]\right)$$

$$n_y = a_y * \left(A_x[i+1, j+1] - A_x[i+1, j]\right)$$

Si $(n_x\text{-}n_y)>0$ entonces se tiene un sector de la cara externa entonces se pinta con Gris, caso contrario se pinta con azul.

Por último, se puede depurar el sombreado manejando diferentes intensidades de colores o texturas en la cara externa como de la interna.

Utilizando el Operador Diferencial Gradiente en la función *z=F(x,y)*, se obtiene el vector

$$Grad(z) = \left(\frac{\partial F(x,y)}{\partial x}, \frac{\partial F(x,y)}{\partial y}\right)$$

Formula 2: Vector Gradiente.

El cual representa la dirección en el espacio según la cual se aprecia una variación de una propiedad física [5], que en este caso se puede utilizar para degradar colores. Si de este vector se calcula la norma euclidea

$$\left\|\left(\frac{\partial F(x,y)}{\partial x}, \frac{\partial F(x,y)}{\partial y}\right)\right\| = \sqrt{\left(\frac{\partial F(x,y)}{\partial x}\right)^2 + \left(\frac{\partial F(x,y)}{\partial y}\right)^2}$$

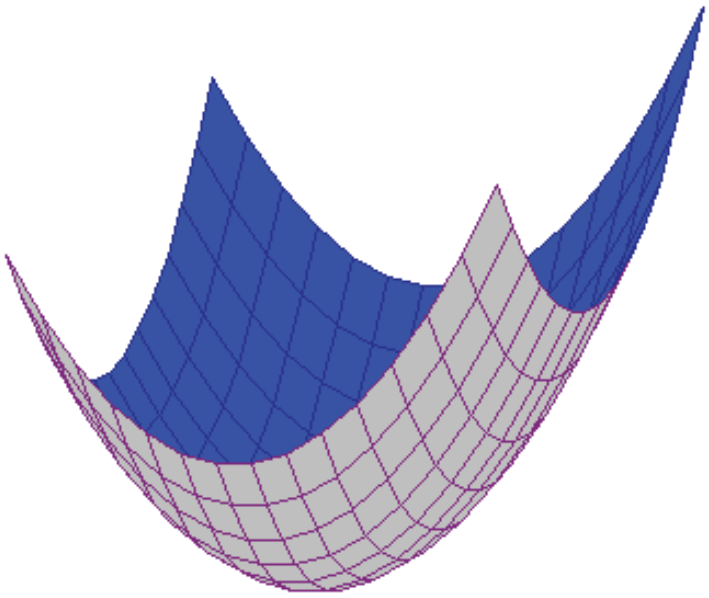


Figura 3: Paraboloide Texturizado.

Realizando una aproximación numérica de las derivadas parciales:

$$k_1 = \frac{F(x+h, y) - F(x, y)}{h}$$

$$k_2 = \frac{F(x, y+h) - F(x, y)}{h}$$

$$Color0(x, y) = \sqrt{k1^2 + k2^2}$$

$$h = 0,00001$$

Se obtiene una intensidad diferente para cada sector del mallado

3. Implementación

3.1. Ventana Real y Ventana Pantalla (ViewPort)

Dentro de los pasos en la línea de ensamblaje que se utiliza para generar imágenes en un computador es fundamental diferenciar entre ventana real y ventana del computador. Es decir, se tienen dos sistemas de referencia: el sistema de referencia real donde se desarrolla el fenómeno a estudiar, y el sistema de referencia de la pantalla [2], [6].

En el sistema de referencia real se identifica la ventana que se quiere transportar sobre la pantalla *(ventana real),* y sobre la pantalla se identifica el rectángulo en el

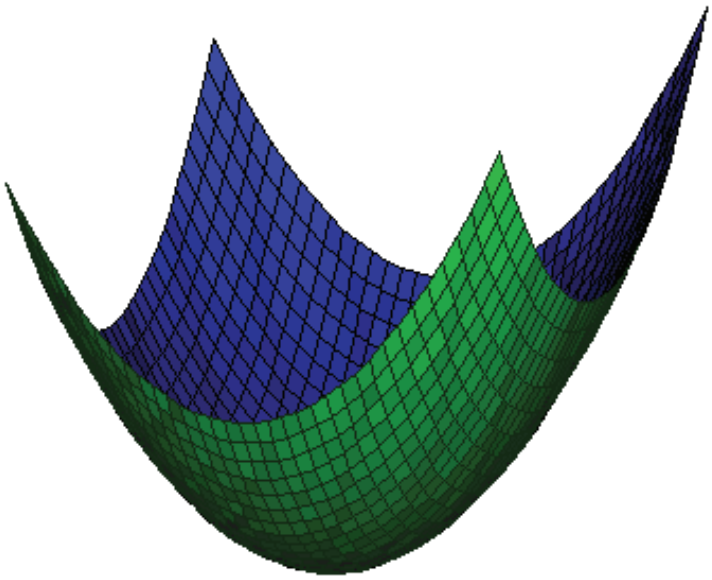


Figura 4: Paraboloide con iluminación.

que tal ventana vendrá transferida *(ventana pantalla)*. Es decir, se debe determinar la transformación (función) que asocia las coordenadas del sistema de referencia real a las coordenadas de referencia de la pantalla en modo que la ventana real venga transformada en la ventana pantalla y viceversa, estos sistemas se pueden apreciar en la Figura 5**.**

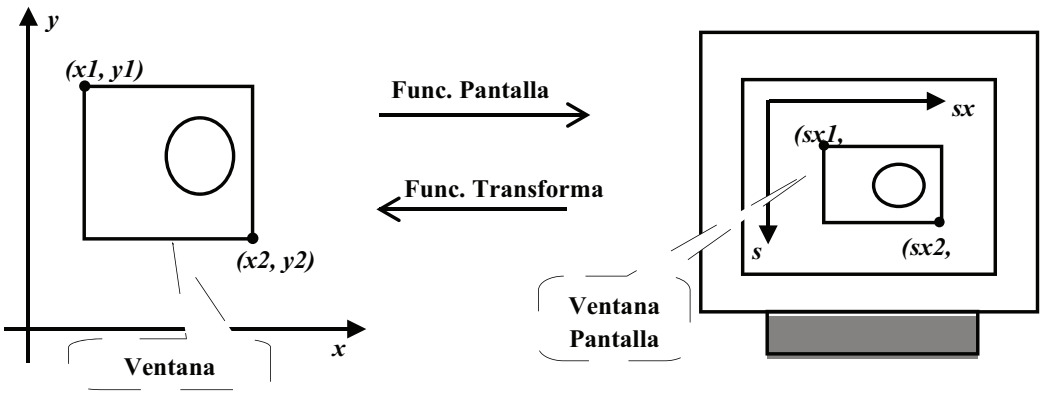


Figura 5: Ventana real y Ventana pantalla.

Se designan con (x_1, y_1) las coordenadas del ángulo superior izquierdo de la ventana real, y con (x_2, y_2) las coordenadas del ángulo inferior derecho de la ventana real. De igual forma con (sx_1, sy_1) las coordenadas del ángulo superior izquierdo de la ventana pantalla, y con (sx_2, sy_2) las coordenadas del ángulo inferior derecho de la ventana pantalla.

Se buscan relaciones que permitan expresar las coordenadas de referencia real con las coordenadas de referencia de la pantalla, para ello se utilizan dos transformaciones matemáticas: Traslación y Escalamiento [4]:

$$\begin{cases} x = \alpha.sx + \beta \\ y = \gamma.sy + \delta \end{cases}$$

Formula 3: Traslación y Escalamiento de Vectores.

con α y β tales que:

$$\begin{cases} x_1 = \alpha.sx_1 + \beta \\ x_2 = \alpha.sx_2 + \beta \end{cases}$$

y ϒ, δ tales que:

$$\begin{cases} y_1 = \gamma.sy_2 + \delta \\ y_2 = \gamma.sy_1 + \delta \end{cases}$$

Resolviendo los dos sistemas lineales se obtienen las relaciones

$$x = x_1 + \frac{(x_2 - x_1)(sx - sx_1)}{(sx_2 - sx_1)}$$
$$y = y_2 - \frac{(y_2 - y_1)(sy - sy_1)}{(sy_2 - sy_1)}$$

Las cuales se implementaron en el Proceso denominado *Pantalla*

3.2. Lenguaje de Programación C#

C# o C Sharp es un lenguaje de programación que está incluido en la Plataforma.NET y corre en el Lenguaje Común en Tiempo de Ejecución (CLR, Common Language Runtime). El primer lenguaje en importancia para el CLR es C#, mucho de lo que soporta la Plataforma.NET está escrito en C#, las características de este lenguaje permiten de una manera fluida la implementación de procesos matemáticos ya que su estructura es muy similar a la estructura matemática [7].

La orientación a los objetos es más pura y clara que en otros lenguajes. C# soporta todas las características del paradigma de la programación orientada a objetos, como son la encapsulación, la herencia y el polimorfismo [8].

Contiene dos categorías generales de tipos de datos integrados: tipos de valor y tipos de referencia. El término tipo de valor indica que esos tipos contienen directamente sus valores.

Además, el uso es sencillo C# elimina muchos elementos añadidos por otros lenguajes y que facilitan su uso y compresión. Es por ello que se dice que C# es auto contenido [7].

3.3. Procedimientos en C#

3.3.1. PROCESO PANTALLA

```
        int PXmin = 0;  int PYmin = 0;
        int PXmax = 560; int PYmax = 440;
        public double Xmax = 14;
        public double Xmin = -14;
        public double Ymax = 11;
        public double Ymin = -11;

public void Pantalla(double x, double y, out int Px, out int Py)
        {
            Px = (int)(((PXmax - PXmin) / (Xmax - Xmin)) * (x -
Xmax)) + PXmax;
            Py = (int)(((PYmax - PYmin) / (Ymin - Ymax)) * (y - Ymax))
+ PYmin;
        }
```

3.3.2. PROCESO VECTOR

```
    public virtual void Encender(Bitmap grafico)
        {
            int Px, Py;
            RepositorioMM m=new RepositorioMM();
            m.Pantalla(this.X0,this.Y0,out Px,out Py);
            if (Px > 0 && Px < 560 && Py > 0 && Py < 440)
                grafico.SetPixel(Px, Py, Color0);
        }

        public virtual void Apagar(Bitmap grafico)
        {
            Color0 = Color.White;
            Encender(grafico);
        }
```

3.3.3. PROCESO AXONOMETRIA

```
public  void Axonometria(double X, double Y, double Z, out double Ax,
out double Ay)
        {
            Ax = Y - X * 0.354;
            Ay = Z - X * (0.35);
        }
```

3.3.4. PROCESO VECTOR 3D

```
public override void Encender(Bitmap grafico)
        {
            double ax, ay;  int sx, sy;
            r.Axonometria(this.X0, this.Y0, this.Z0, out ax, out ay);
            r.Pantalla(ax, ay, out sx, out sy);
            if (sx > 0 && sx < 560 && sy > 0 && sy < 440)
            {
                grafico.SetPixel(sx, sy, Color0);
            }
        }
```

3.3.5. PROCESO SUPERFICIE

```
x = -10; dx = 0.4; i = 0;
do
  {
    y = -8; dy = 0.4; j = 0;
    do
      {
        v3d.X0 = x; v3d.Y0 = y;
        v3d.Z0 = (v3d.Y0 *Cos(v3d.X0) + v3d.X0 *Sin(v3d.Y0));
        Axonometria(v3d.X0, v3d.Y0, v3d.Z0, out rx, out ry);
        Mx[i, j] = rx;
        My[i, j] = ry;
        j = j + 1; y += dy;
       }
       while (y <= 8);
            i = i + 1; x += dx;
      }
      while (x <= 10);
       Ni=i; Nj=j;
 for (i=0; i<Ni; i++)
    {
     for (j=0; j<Nj; j++)
      {
          ax=(Mx[i+1, j]- Mx[i, j]);
          nx=ax*( My[i+1, j+1]- My[i+1, j]);
          ay=(My[i+1, j]- My[i, j]);
          ny=ay*(Mx[i+1, j+1]- Mx[i+1, j]);
          if (nx-ny)>0
             {
                 ColorS=rgb(0,Color0(Mx[i,j],My[i,j])%255,0);
              }
           else  ColorS=rgb(0,0,Color0(Mx[i,j],My[i,j])%255);
          Cuadrilatero(Mx[i,j],My[i,j],Mx[i+1, j],My[i+1, j],
           Mx[i+1,j+1],My[i+1,j+1],Mx[i,j+1],My[i,j+1],ColorS);
         }
}
```

4. EJEMPLO

Superficie: $z = x.\cos(y) + y.sen(x)$

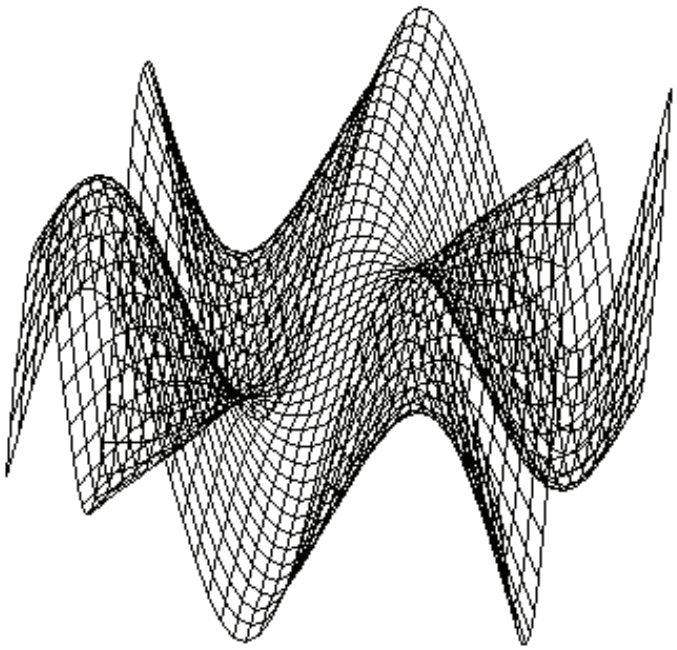


Figura 6: Superficie Mallada.

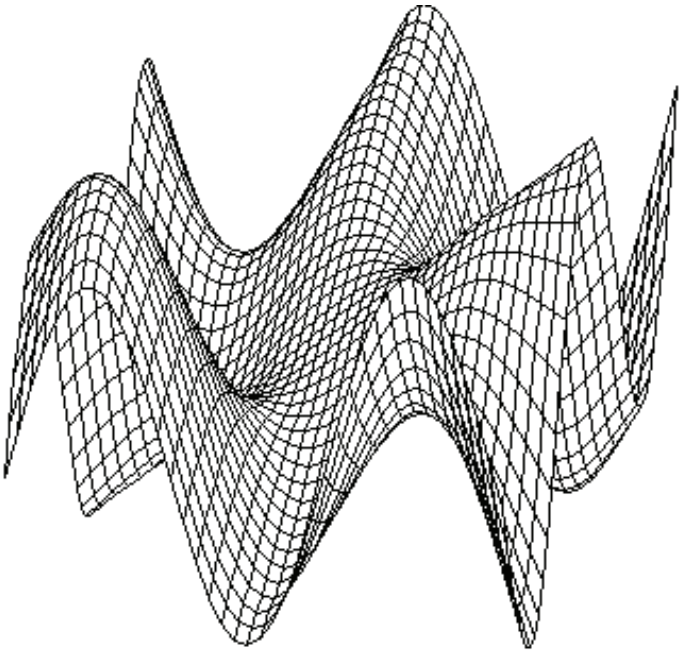


Figura 7: Superficie Eliminando secciones.

5. Concluciones

Los algoritmos clásicos como el *Algoritmo del Pintor* no son tan eficientes al eliminar secciones escondidas en superficies de alta complejidad como por ejemplo las superficies de revolución, a diferencia del algoritmo que calcula el ángulo del observador y la perpendicular a la superficie que lo realiza de mejor forma. Además, el vector normal

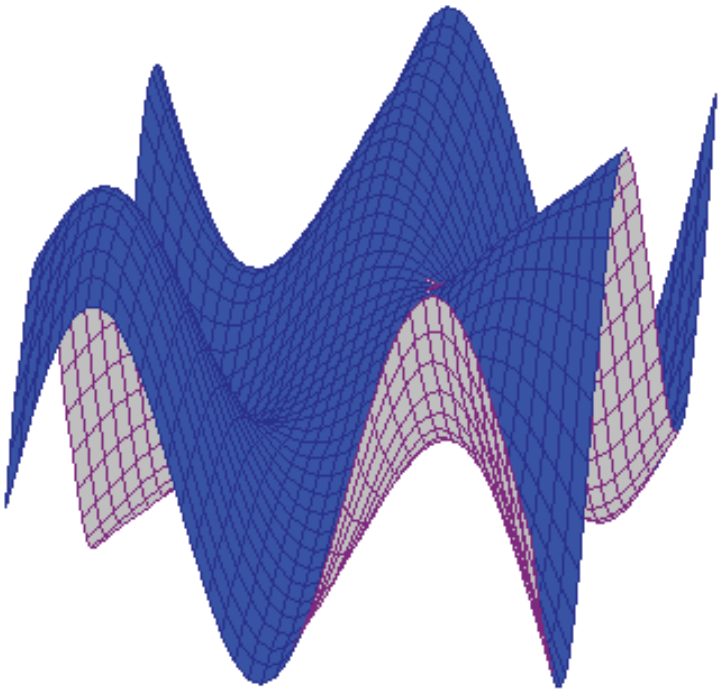


Figura 8: Superficie Texturizada.

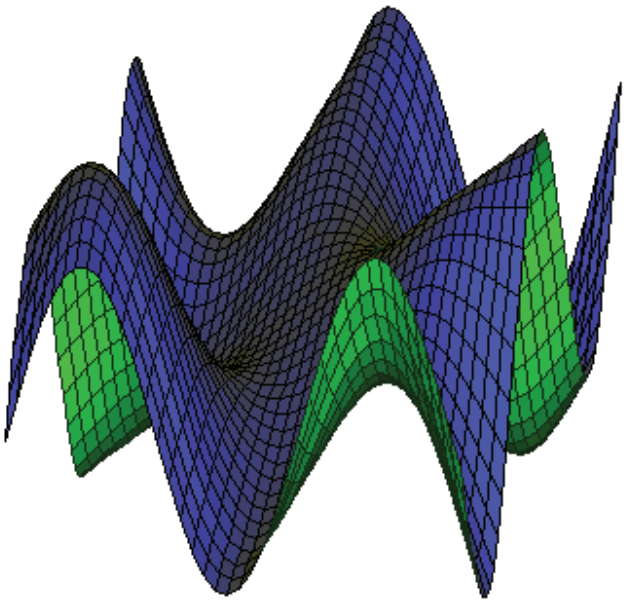


Figura 9: Superficie con Iluminación.

en cada sector de la superficie es un buen indicador para decidir si es un sector externo o interno de la superficie. El gradiente de un operador (Superficie) al medir la dirección de variación de magnitudes fue muy útil para calcular intensidades en la textura de la superficie (iluminación). Finalmente, las herramientas del Calculo Vectorial son más fuertes que las herramientas de la Geometría clásica al generar superficies en un computador.

Las expectativas futuras con este trabajo es poder aplicar las técnicas del Calculo Vectorial para graficar planos tangentes a las superficies o trazar curvas de nivel respecto a cualquiera de los ejes cartesianos.

Referencias

[1] N. Castellanos, «Reconstruccion y sombreado de superficies tridimensionales anatomicas a partir de cortes tomograficos,» 1995. [En línea]. Available: http://tesiuami.izt.uam.mx/uam/aspuam/presentatesis.php?recno=3639&docs=UAM3639.PDF

[2] D. Bini, O. Menchi, "Matematica, mondo reale e calcolatore", Zanichelly, Italia. 2001.

[3] J. Glyn, " Matemáticas Avanzadas para Ingeniería", Pearson, Madrid. 2012.

[4] A. Álvarez, "Matemática de la Computación Grafica para la simulación de fenómenos ondulatorios y dinámicos", Tesis de Maestría en Informática Aplicada, ESPOCH, 2010

[5] L. Meza, «Gradiente, divergencia y rotacional,» 2017. [En línea]. Available: http://www.ifuap.buap.mx/$\sim$lilia/mating/Grad-div-rot.pdf.

[6] M. Garcia, "Creación de un software con programación concurrente para la iluminación y sombreado de superficies vectoriales utilizando gradientes vectoriales", Tesis de Pregrado, ESPOCH, 2016.

[7] Servicio de Informatica, «Curso.Net con C#,» [En línea]. Available: https://si.ua.es/es/documentacion/c-sharp/documentos/masterpages/modulo1.pdf.

[8] J. González, «El Lenguaje de Programación C#,» 2015. [En línea]. Available: http://users.dsic.upv.es/$\sim$jlinares/csharp/lenguajeCsharp.pdf.