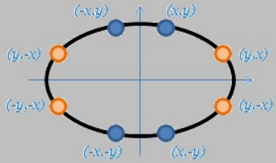
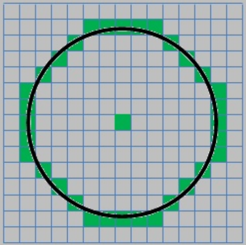
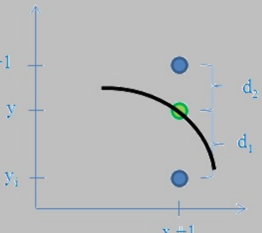






Universidad Autónoma del Estado de México
Centro Unviersitario UAEM Zumpango

Graficación
Unidad de Competencia III
Generación de objetos poligonales y poliédricos. Temas 3 al 8.

M. en C. Rafael Rojas Hernández

Septiembre 2017

Índice de la presentación

- Información general de la Unidad
- Estructura de la Unidad de Aprendizaje
- Unidad de Competencia III

Unidad de Aprendizaje

Graficación

Propósito de la Unidad de Aprendizaje

El alumno establecerá los fundamentos y algoritmos básicos dentro del campo de la infografía y graficación, y elaborará herramientas o aplicaciones específicas como resolución a problemas del área de trabajo.

Estructura de la Unidad de Aprendizaje

1. Introducción a la Graficación e Infografía.
2. **Generación de objetos poligonales y poliédricos.**
3. Transformaciones elementales en 2D y 3D.
4. Modelado de objetos en 2D y 3D.
5. Visualización de objetos controlando las condiciones lumínicas.
6. Generación de imágenes escenarios virtuales y Animación.

- Dibujo de circunferencias
  - Simetría de ocho lados
  - Método de Bresenham
- Dibujo de elipses
- Representación de polígonos y poliedros
  - Polígonos convexos y cóncavos
  - Triangulación de polígonos
- Métodos de rellenado de áreas
  - Relleno por barrido (raster)
  - Relleno por inundación
- Otras formas
- Aproximaciones poligonales para el dibujo de curvas
  - Problemas de inclusión

Convexidad

Operadores

- Geométricos
- Topológicos
- Boléanos

Objetivo de la Unidad de Competencia

Describir y manipular los métodos básicos para construir objetos poligonales y poliédricos en la práctica.

Conocimientos

- Ilustra el modelado poligonal y poliédrico.
- Dibuja líneas a través de diversos métodos.
- Dibuja circunferencias a través de diversos métodos.
- Dibuja polígonos y poliedros.
- Emplea métodos de rellenado de áreas.
- Utiliza aproximaciones poligonales para el dibujo de curvas.
- Aplica operadores geométricos, topológicos y boléanos.

Habilidades

Diseña objetos elementales en perspectiva en la pantalla, soluciona problemas prácticos de dibujo de formas, y bosqueja objetos elementales.

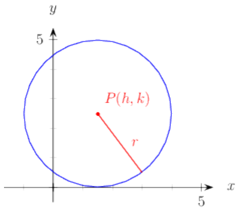
Actitudes y valores

Autonomía, responsabilidad, respeto, tolerancia, puntualidad y trabajo. Selecciona métodos apropiados, explica conceptos, propone soluciones, resuelve problemas, pone en práctica los conocimientos adquiridos, actúa conforme a un plan.

Circunferencia

Lugar geométrico de un punto que se mueve en un plano de tal manera que se conserva siempre a una distancia (*radio*) constante de un punto fijo (*centro*) en ese plano.

Analíticamente es una ecuación de segundo grado con dos variables.



Dibujo de circunferencias

Ecuación cartesiana

La circunferencia con centro en el punto $P(x_c, y_c)$ y con radio constante r, tiene la ecuación:

$$(x - x_c)^2 + (y - y_c)^2 = r^2$$

Si el centro es el origen la ecuación es:

$$x^2 + y^2 = r^2$$

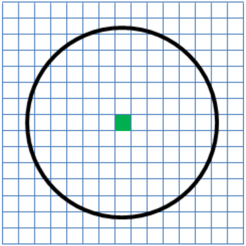
Ecuación general

$$x^2 + Dx + y^2 + Ey + F = 0$$

con centro en el punto $\left(-\frac{D}{2}, -\frac{E}{2}\right)$ y radio $r = \frac{1}{2}\sqrt{D^2 + E^2 - 4F}$.

Dibujo de circunferencias

Las ecuaciones anteriores funcionan cuando se tiene el caso de un plano continuo, sin embargo, para la gráficar una circunferencia en el plano discreto surge el siguiente problema:



Dibujo de circunferencias

La ecuación cartesiana podría usarse para trazar una circunferencia recorriendo el eje x en pasos unitarios de $x_c - r$ a $x_c + r$ y calcular los valores de y de la siguiente forma:

$$y = y_c \pm \sqrt{r^2 - (x - x_c)^2}$$

Con en el centro en el origen:

$$y = \pm\sqrt{r^2 - x^2}$$

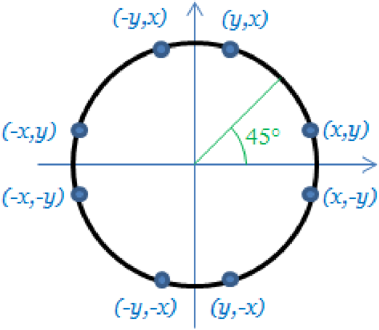
Realizar este procedimiento implica una tarea de cálculo considerable, además de que el espaciamiento entre las posiciones de los pixeles trazados no es uniforme.

Una forma de resolver el problema de espaciamiento consiste en calcular puntos situados en la frontera circular mediante el uso de coordenadas polares:

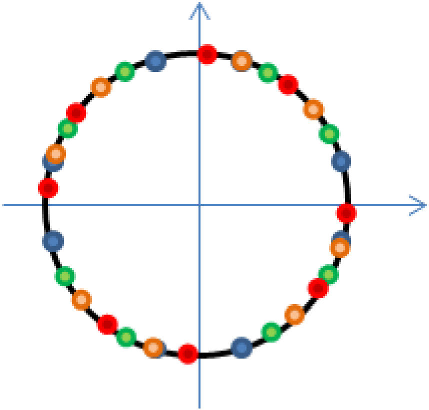
$$x = x_c + r \cdot \cos\theta$$

$$y = y_c + r \cdot \operatorname{sen}\theta$$

Es posible mejorar la manera de trazar el circulo considerando la simetría de la circunferencia, considerando que un punto puede ser trazado en otros puntos de la circunferencia intercambiando coordenadas y alterando el signo de los valores coordenados; de esta manera es posible trazar 8 puntos, uno por cada sección de 45°.



Utilizando los 8 puntos de simetría de la circunferencia, recorriendo el eje x en incrementos de 1 , y caluclando el valor de y con la ecuación cartesiana.



Algorithm 1.1: Ocho Puntos

Data: Centro $P_c(x_c, y_c)$ y Punto simetría $P(x, y)$
Result: Puntos simetricos

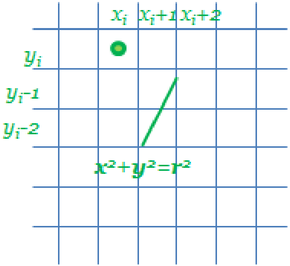
putPixel($x_c + x, y_c + y$);
putPixel($x_c - x, y_c + y$);
putPixel($x_c + x, y_c - y$);
putPixel($x_c - x, y_c - y$);
putPixel($x_c + y, y_c + x$);
putPixel($x_c - y, y_c + x$);
putPixel($x_c + y, y_c - x$);
putPixel($x_c - y, y_c - x$);

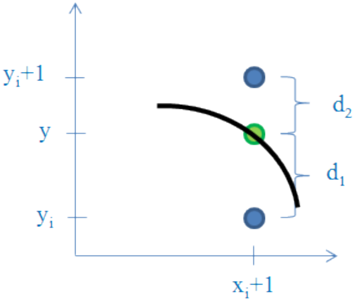
Algorithm 1.2: Circulo Básico

Data: Centro $P_c(x_c, y_c)$ y radio r
Result: Circunferencia

$x = 0$;
$y = r$;
$f = r$;
while $x \leq f$ **do**
    OchoPuntos(x_c, y_c, x, f);
    $x = x + 1$;
    $y = \sqrt{r^2 - x^2}$;
    $f = \lfloor y + 0.5 \rfloor$;

Al igual que para el trazado de líneas las posiciones enteras de una trayectoria circular puede determinarse considerando cuál de los dos pixeles esta más próximo a la circunferencia.





La posición de y considerando en primer lugar el centro de la circunferencia en cero, y las dos distancias d_1 y d_2 pueden serán exactamente igual a la mitad de la distancia entre el punto $(x_i + 1, x_i)$ y $(x_i + 1, y_i - 1)$, es decir el punto $(x_i + 1, y_i - \frac{1}{2})$.
De esta manera si la distancia $d = (x_i + 1)^2 + (y_i - \frac{1}{2})^2 - r^2$, teniendo que:

$$p_i = \begin{cases} (x_i + 1, y_i - 1) & si \ d > 0 \\ (x_i + 1, y_i) & si \ d < 0 \end{cases}$$

Considerando para el siguiente valor de x se tiene $\left(x_i + 2, y_i - \frac{1}{2}\right)$.

Para el primer caso $d > 0$se tiene que:

$$\begin{aligned} d_i &= (x_i+2)^2 + \left(y_i - \frac{1}{2}\right)^2 - r^2 \\ &= (x_i^2 + 4x_i + 4) + \left(y_i - \frac{1}{2}\right)^2 - r^2 \\ &= (x_i^2 + 2x_i + 1) + 2x_i + 3 + \left(y_i - \frac{1}{2}\right)^2 - r^2 \\ &= (x_i+1)^2 + \left(y_i - \frac{1}{2}\right)^2 - r^2 + 2x_i + 3 \end{aligned}$$

Los primeros tres términos corresponde al valor del punto central, por lo que la distancia dependerá de la distancia del punto anterior, quedando:

$$d_i = d_i + 2x_i + 3$$

Para el primer caso $d < 0$se evalua el punto ():

$$\begin{aligned} d_i &= (x_i+2)^2 + \left(y_i - \frac{3}{2}\right)^2 - r^2 \\ &= (x_i^2 + 4x_i + 4) + \left(y_i^2 - 3y_i + \frac{9}{4}\right) - r^2 \\ &= (x_i^2 + 2x_i + 1) + 2x_i + 3 + \left(y_i^2 - y_i + \frac{1}{4}\right) - 2y_i + 2 - r^2 \\ &= (x_i+1)^2 + (y_i - \left(\frac{1}{2}\right)^2 - r^2 + 2x_i - 2y_i + 5 \end{aligned}$$

Los primeros tres términos corresponde al valor del punto central, por lo que la distancia dependerá de la distancia del punto anterior, quedando:

$$d_i = d_i + 2x_i - 2y_i + 5$$

Utilizando las dos distancia anteriores podemos considerar que se generaliza para todo valor de i, excepto para el valor inicial x_0, ya que se evalua el punto $\left(1, r - \frac{1}{2}\right)$ quedando:

$$\begin{aligned} d_0 &= (1)^2 + \left(r - \frac{1}{2}\right)^2 - r^2 \\ &= 1 + \left(r^2 - r + \frac{1}{4}\right) - r^4 \\ &= 1 - r + \frac{1}{4} \\ &= \frac{5}{4} - r \end{aligned}$$

Algorithm 1.3: Circulo Bresenham

```
Data: Centro P_c(x_c, y_c) y radio r
Result: Circunferencia
x = 0;
y = r;
d = 5/4 − r;
while x ≤ y do
    OchoPuntos(x_c, y_c, x, y);
    if d < 0 then
        d = d + 2x + 3;
    else
        d = d + 2(x − y) + 5;
        y=y-1;
    x = x + 1;
```

En el algoritmo anterior el valor inicial de d resulta un valor no entero, sin embargo, difiere del espacio discreto que estamos graficando, ya que los incrementos son siempre cantidades enteras, por lo que es posible hacer un cambio de variable de la siguiente manera:

$$\begin{aligned} h &= d - \frac{1}{4} \\ &= \frac{5}{4} - r - \frac{1}{4} \\ &= 1 - r \end{aligned}$$

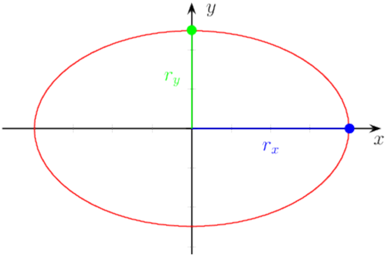
Algorithm 1.4: Circulo Bresenham modificado

Data: Centro $P_c(x_c, y_c)$ y radio r
Result: Circunferencia

$x = 0$;
$y = r$;
$h = 1 - r$;
while $x \leq y$ **do**
  OchoPuntos(x_c, y_c, x, y);
  if $d < 0$ **then**
    $h = h + 2x + 3$;
  else
    $d = d + 2(x - y) + 5$;
    y=y-1;
  $x = x + 1$;

Elipse

Es el lugar geométrico de los puntos cuya suma de distancias a dos puntos fijos es constante. Los puntos fijos se llaman *focos*.



Ecuación cartesiana

La elipse con centro en el punto $P(x_c, y_c)$ y con radios r_x y r_y tiene la ecuación:

$$\frac{(x - x_c)^2}{r_x^2} + \frac{(y - y_c)^2}{r_y^2} = 1$$

Si el centro es el origen la ecuación es:

$$\frac{x^2}{r_x^2} + \frac{y^2}{r_y^2} = 1$$

La ecuación cartesiana podría usarse para trazar una elipse recorriendo el eje x en pasos unitarios de $x_c - r_x$ a $x_c + r_x$ y calcular los valores de y de la siguiente manera:

$$y = y_c \pm \frac{r_y}{r_x}\sqrt{r_x^2 - (x - x_c)^2}$$

Con el centro en el origen:

$$y = \pm \frac{r_y}{r_x}\sqrt{r_x^2 - x^2}$$

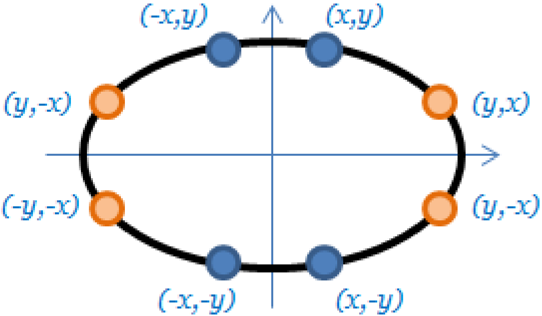
De igual manera es posible recorrer el eje y en pasos unitarios de $y_c - r_y$ a $y_x + r_y$ y calcular los valores de x de la siguiente forma:

$$x = x_c \pm \frac{r_x}{r_y}\sqrt{r_y^2 - (y - y_c)^2}$$

y también:

$$x = \pm \frac{r_x}{r_y}\sqrt{r_y^2 - y^2}$$

Es posible mejorar la manera de trazar la elipse considerando su simetría en x e y, considerando que un punto puede ser trazado intercambiando de signo los valores coordenados, (x, y) para el eje x y (y, x) para el eje y.



La manera en como se trazarán los puntos dependerá de la sección, para ello es necesario obtener la pendiente de la elipse:

$$\frac{f(x,y)}{dxdy} = \frac{-r_y^2 x}{r_x^2 y}$$

Para el caso del desplazamiento sobre x se considera que el numerador sea menor al denominador y para y el caso contrario.

Algorithm 2.1: Cuatro Puntos

Data: Centro $P_c(x_c, y_c)$, radio en x (r_x) y radio en y (r_y)
Result: Puntos simétricos

```
putPixel(x_c + x, y_c + y);
putPixel(x_c + x, y_c - y);
putPixel(x_c - x, y_c + y);
putPixel(x_c - x, y_c - y);
```

Algorithm 2.2: Elipse Básico

Data: Centro $P_c(x_c, y_c)$, radio en x (r_x) y radio en y (r_y)
Result: Elipse

$x = 0;$
$y = r_y;$
$f = r_y;$
while $r_y^2 x < r_x^2 f$ **do**
    cuatroPuntos$(x_c, y_c x, f)$;
    $x = x + 1;$
    $y = \frac{r_y}{r_x}\sqrt{r_x^2 - x^2};$
    $f = \lfloor y + 0.5 \rfloor;$
$y = 0;$
$x = r_x;$
$f = r_x;$
while $r_y^2 f >= r_x^2 y$ **do**
    cuatroPuntos$(x_c, y_c x, f)$;
    $y = y + 1;$
    $x = \frac{r_x}{r_y}\sqrt{r_y^2 - y^2};$
    $f = \lfloor x + 0.5 \rfloor;$

Similar al caso de la circunferencia, existe la duda de cuál de los dos pixeles es el más cercano. es posible reescribir la ecuación de la elipse de la siguiente manera:

$$r_y^2(x - x_c)^2 + r_x^2(y - y_c)^2 - r_x^2 r_y^2 = 0$$

O con centro en el origen:

$$r_y^2 x^2 + r_x^2 y^2 - r_x^2 r_y^2 = 0$$

Teniendo ahora que evaluar cuatro casos, dos para x y dos para y, cada uno con la condición de la distancia obtenida de manera similar que en la circunferencia.

Algorithm 2.3: Elipse Bresenham

Data: Centro $P_c(x_c, y_c)$, radio en x (r_x) y radio en y (r_y)

Result: Elipse

$x = 0;$
$y = r_y;$
$d = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2;$
while $r_y^2 x < r_x^2 y$ **do**
    cuatroPuntos(x, f);
    if $d < 0$ **then**
        $d = d + r_y^2(2x + 3);$
    else
        $d = d + r_y^2(2x + 3) + 2r_x^2(1 - y);$
        $y = y - 1;$
    $x = x + 1;$
$x = r_x;$
$y = 0;$
$d = r_x^2 - r_y^2 r_x + \frac{1}{4} r_y^2;$
while $r_y^2 x \geq r_x^2 y$ **do**
    cuatroPuntos(f, y);
    if $d < 0$ **then**
        $d = d + r_x^2(2y + 3);$
    else
        $d = d + r_x^2(2y + 3) + 2r_y^2(1 - x);$
        $x = x - 1;$
    $y = y + 1;$

Algorithm 2.4: Elipse Bresenham (cont.)

$x = r_x$;
$y = 0$;
$d = r_x^2 - r_y^2 r_x + \frac{1}{4} r_y^2$;
while $r_y^2 x \geq r_x^2 y$ **do**
    cuatroPuntos(f, y);
    if $d < 0$ **then**
        $d = d + r_x^2(2y + 3)$;
    else
        $d = d + r_x^2(2y + 3) + 2r_y^2(1 - x)$;
        $x = x - 1$;
    $y = y + 1$;

Algorithm 2.5: Elipse Bresenham modificado

Data: Centro $P_c(x_c, y_c)$, radio en x (r_x) y radio en y (r_y)
Result: Elipse

$x = 0;$
$y = r_y;$
$h = \lfloor r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2 + 0.5 \rfloor;$
while $r_y^2 x < r_x^2 y$ **do**
  cuatroPuntos(x, y);
  if $h < 0$ **then**
    $h = h + r_y^2(2x + 3);$
  else
    $h = h + r_y^2(2x + 3) + 2r_x^2(1 - y);$
    $y = y - 1;$
  $x = x + 1;$

Algorithm 2.6: Elipse Bresenham modificado

$x = r_x$;
$y = 0$;
$h = \lfloor r_x^2 - r_y^2 r_x + \frac{1}{4} r_y^2 \rfloor$;
while $r_y^2 x \geq r_x^2 y$ **do**
    cuatroPuntos(x, y);
    if $h < 0$ **then**
        $h = h + r_x^2(2y + 3)$;
    else
        $h = h + r_x^2(2y + 3) + 2r_y^2(1 - x)$;
        $x = x - 1$;
    $y = y + 1$;

Algorithm 3.1: Polígono

Data: Centro $P_c(x_c, y_c)$, radio r y número de lados l
Result: Polígono

$\alpha = \frac{360^\circ}{l}$;
$a = 0$;
$x_0 = x_c + \lfloor r \cdot \cos(90^\circ - a) + 0.5 \rfloor$;
$y_0 = y_c + \lfloor r \cdot \sin(-90^\circ - a) + 0.5 \rfloor$;
for $i = 1, 2, \cdots, l - 1$ **do**
    $a = a + \alpha$;
    $x_i = x_c + \lfloor r \cdot \cos(90^\circ - a) + 0.5 \rfloor$;
    $y_i = y_c + \lfloor r \cdot \sin(-90^\circ - a) + 0.5 \rfloor$;
    $\text{linea}(x_{i-1}, y_{i-1}, x_i, y_i)$;
$\text{linea}(x_0, y_0, x_{l-1}, y_{l-1})$;

Algorithm 3.2: Polígono

Data: Centro $P_c(x_c, y_c)$, radio r y número de lados l

Result: Polígono

$\alpha = \frac{360^\circ}{l}$;

$a = 0$;

$x_0 = x_c + \lfloor r \cdot \cos(90^\circ - a) + 0.5 \rfloor$;

$y_0 = y_c + \lfloor r \cdot \sin(-90^\circ - a) + 0.5 \rfloor$;

for $i = 1, 2, \cdots, l-1$ **do**

- $a = a + \alpha$;
- $x_i = x_c + \lfloor r \cdot \cos(90^\circ - a) + 0.5 \rfloor$;
- $y_i = y_c + \lfloor r \cdot \sin(-90^\circ - a) + 0.5 \rfloor$;
- linea$(x_c, y_c, x_{i-1}, y_{i-1})$;
- linea$(x_{i-1}, y_{i-1}, x_i, y_i)$;
- linea(x_c, y_c, x_i, y_i);

linea$(x_0, y_0, x_{l-1}, y_{l-1})$;

Algorithm 4.1: Relleno por barrido

Data: Puntos $P_1(x_1, y_1)$ y $P_2(x_2, y_2)$

Result: Relleno

```
n = 0;
maxPuntos = |x2 − x1|;
inter[maxPuntos] = {−1, −1, ··· , −1};
inter2[maxPuntos] = {−1, −1, ··· , −1};
for y = y1, ··· , y2 do
    for x = x1, ··· , x2 do
        if getPixel(x, y)!=0 then
            inter_i = x;
            i = i + 1;
    if n <> 0 then
        aux = inter_0;
        a = 0;
        ban = FALSO;
        for i = 1, ··· , n do
            if |aux − inter_i| = 1 then
                aux = inter_i;
                ban=VERDADERO;
            else
                inter2_a = inter_i;
                a = a + 1;
                ban=FALSO;
            if ban = FALSO then
                inter2_a = aux;
                a = a + 1;
        for i = 0, 2, 4, ··· , n − 1 do
            linea(inter2_i, y, inter2_{i+1}, y);
```

Algorithm 4.2: 4-conexo

Data: Centro $P(x, y)$
Result: Relleno

```
if getPixel(x, y) == fondo then
    putPixel(x, y);
    4-conexo(x, y + 1);
    4-conexo(x + 1, y);
    4-conexo(x − 1, y);
    4-conexo(x, y − 1);
```

Otras formas
Línea con un sólo punto

Algorithm 5.1: Línea con un sólo punto

Data: Coordenadas punto $P_0(x_0, y_0)$
Result: Línea

```
if existe primer punto then
    linea(x_aux, y_aux, x_0, y_0);
else
    Ya existe un primer punto;
x_aux = x_0;
y_aux = y_0;
```

if *existe primer punto* **then**
  linea($x_{aux}, y_{aux}, x_0, y_0$);
else
  Ya existe un primer punto;
$x_{aux} = x_0$;
$y_{aux} = y_0$;

Bibliografía

- D. Marsh, Applied Geometry for Computer Graphics and CAD, Springer 2005.
- D. Salomon, Curves and Surfaces for Computer Graphics, Springer 2005.
- P. Comninos, Mathematical and Computer Programming Techniques for Computer Graphics, Springer, 2004.
- F. S. Hill, Computer Graphics, Prentice-Hall, 1990.
- E. Angel, Interactive Computer Graphics, Addison Wesley, 2000.
- D. Hearn & M. P. Baker, Gráficas por Computadora, prentice-Hall, 1988.