INFORMÁTICA GRÁFICA:

ESTRELLAS

Gallardo González, Pablo Miguel
70892413-L

Departamento de Informática y Automática

Universidad de Salamanca

1. Diseño e implementación

Para la creación de los triángulos, se tuvo que realizar un exhaustivo estudio previo para descubrir cómo se generaban de manera matemática de una forma generalizable para cualquier número de lados.

El primer paso era conseguir generar figuras básicas de n lados, como lo son el pentágono, el hexágono, el heptágono, etc. El problema se definió como la creación de una circunferencia y la división de la misma en n puntos. La unión de estos puntos generaría las figuras deseadas.

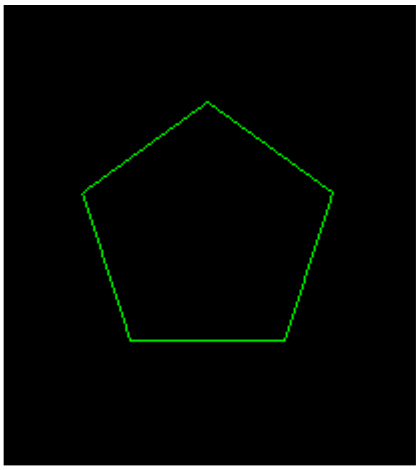

Figura 1: el pentágono, una figura básica

Supongamos siempre que la figura está en el centro de coordenadas. Siendo *x* e *y* las coordenadas x e y de un punto, y θ el ángulo de la circunferencia, la relación entre el ángulo y los puntos de una circunferencia es:

$$tg\,\theta = \frac{x}{y}$$

La única información que tenemos nosotros de la figura es el radio. Siendo *p* el número de partes en que queremos dividir la circunferencia (número de puntos del polígono), y *n* el número de punto del que queremos obtener las coordenadas, relacionamos el radio y el ángulo de la siguiente manera:

$$\theta \cdot r = n\frac{2 \cdot \pi \cdot r}{p}$$

Poniendo en común ambas fórmulas, desaparecerá la r, y al final tendremos relacionada la x y la y con el número de punto que queremos obtener.

$$y = \frac{x}{tg^2(n \cdot 2 \cdot \pi / p)}$$

La ecuación de la circunferencia es la siguiente:

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

Poniendo estas dos fórmulas en común y despejando, obtenemos:

$$x = \frac{r \cdot tg(n \cdot 2 \cdot \pi / p)}{\sqrt{1 + tg^2(n \cdot 2 \cdot \pi / p)}}$$

De esta manera, si queremos obtener las coordenadas *x* e *y* del punto *n* (donde *n = 0, 1, …, p-1*) de un polígono de *p* lados con centro en el eje de coordenadas cartesianas, sabiendo sólo el radio *r* de la circunferencia, las ecuaciones a utilizar serán las siguientes:

$$x = \frac{r \cdot tg(n \cdot 2 \cdot \pi / p)}{\sqrt{1 + tg^2(n \cdot 2 \cdot \pi / p)}}$$

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

Ya tenemos los puntos del polígono. Pero una estrella las estrellas que queremos no son las geométricas que se forman mediante la unión de estos puntos, lo que queremos es una más artística. Por ello tendremos que utilizar el doble de puntos, y marcar en los puntos pares el radio a la mitad. De esta forma obtendremos el aspecto de estrella que deseamos.

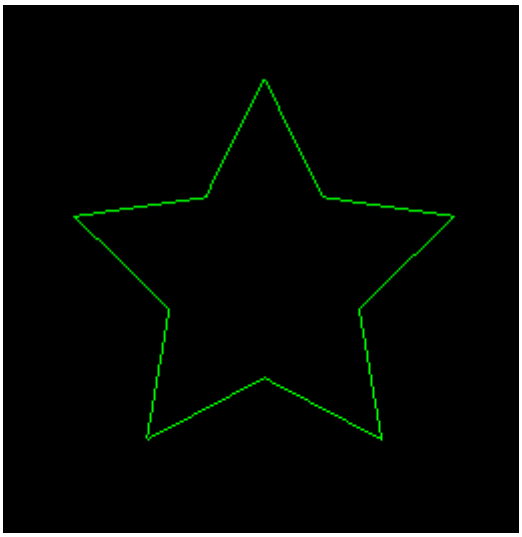

Figura 2: estrella estándar

Sólo tendremos que unir cada uno de estos puntos con otro que esté situado en el centro, pero en un plano distinto al resto de puntos, rellenarlo todo como si fuera un polígono y no líneas, y ya tendremos un resultado tridimensional de la estrella.

2. Volumen de trabajo empleado

La materia de la que trata la asignatura (el tratamiento de funciones de biblioteca OpenGL) era nuevo para el alumno que realizó la práctica, por lo que la parte de aprendizaje se unió a la de implementación. Muchos aspectos técnicos fueron eliminados, añadidos o modificados a lo largo de la realización de la práctica atendiendo a la adquisición de nuevos conocimientos, y al entendimiento de cuales serían los efectos más fáciles y alcanzables con OpenGL.

Por ello, el tiempo utilizado en la creación de la práctica ha sido mucho mayor del que se ha tardado en escribir el código o poner a prueba cada cambio realizado en el código. Se ha buscado consejo y se han seguido tutoriales de páginas webs, y se ha consultado la bibliografía en los momentos en los que ha hecho falta.

La investigación sobre la descripción matemática de la estrella ha sido un trabajo propio, lo cual ha consumido mucho tiempo, pues era información que no se encontraba en cualquier lugar y que suponía un repaso y una deducción de los conocimientos matemáticos.

3. Figuras modeladas en la escena e utilización de primitivas

El salva pantallas se compone de varias figuras de estrellas, con distintas características cada una, las cuales rotan de forma distinta. Cada estrella tiene un número de puntas distintos, por lo que pueden considerarse como figuras distintas. Se han utilizado las fórmulas matemáticas para describir de forma general cada una de las estrellas, de forma que este proceso se ha convertido en totalmente automático.

Para la creación de la estrella, se han hallado los puntos de todos los vértices de la estrella en dos dimensiones. Después, se ha cogido el punto que está justo en el centro de la estrella pero ligeramente desplazado hacia delante, y se han unido utilizando la función glBegin(GL_TRIANGLE_FAN). Esto hace que todos los puntos definidos después se unan a éste primero que se ha declarado.

De esta forma habremos generado media estrella. Si dibujábamos una estrella similar dada la vuelta, obteníamos problemas de iluminación, ya que normalmente se iluminaba la cara interna (y no visible) de la estrella. Esto se solucionó mediante llamadas a glRotate, de forma que sólo tenemos que duplicar la estrella recién creada y solaparla correctamente con su otra mitad.

Para la rotación y traslación de la figura, se han utilizado llamadas glRotate y glTranslate con los valores adecuados. El resto de llamadas a funciones de OpenGL responden al establecimiento inicial de opciones de OpenGL y a la limpieza de búffers.

4. Iluminación

Para que la iluminación tenga el efecto deseado sobre cada figura, se debe definir el vector normal a cada superficie. Con esta información, las rutinas de iluminación de OpenGL se encargarán de aplicar la intensidad de luz correcta para cada cara.

Para encontrar la normal de una superficie poligonal, basta con tomar tres vértices de la superficie (los cuales se representan como conjuntos de tres coordenadas v_1, v_2 y v_3) y hacer el siguiente producto vectorial:

$$[v_1 - v_2] \times [v_2 - v_3]$$

El resultado de esta operación será un vector normal al plano representado por los tres puntos. Es importante que estos tres puntos no estén situados en línea recta. El producto vectorial, desarrollado, quedaría de la siguiente manera, de forma que sólo tendríamos que calcular restas y productos:

$$[v_x\, v_y\, v_z] \times [w_x\, w_y\, w_z] = [(v_y\, w_z - w_y\, v_z)\, (w_x\, v_z - v_x\, w_z)\, (v_x\, w_y - w_x\, v_y)]$$

En nuestro caso, al tratarse de un triángulo, los tres vértices que se han tomado para calcular la normal son los mismos que los que se usa para crear cada triángulo. Como se puede predecir viendo las dos fórmulas anteriores, los cálculos para hallar el vector normal de cada superficie son numerosos, y suponen muchas restas y productos. Los resultados de estos cálculos para cada triángulo se introducen en función glNormal*(), justo antes de dibujarse en pantalla.

Ahora ya podemos utilizar las funciones que provee OpenGL para crear efectos de iluminación.

Cuando se inicia el programa, se indica en la función InitGL las características de la luz que queremos obtener, mediante llamadas de tipo glLightModel o glLightfv. Una vez hecho esto, podremos activar los efectos de luz con glEnable.

También hemos tenido en cuenta la iluminación a la hora de dibujar las figuras. Además de realizar el cálculo de las normales, como bien hemos visto antes, hemos dotado de propiedades reflectantes al material del que están constituidas las estrellas mediante llamadas de tipo glMaterial.

5. Animación final conseguida

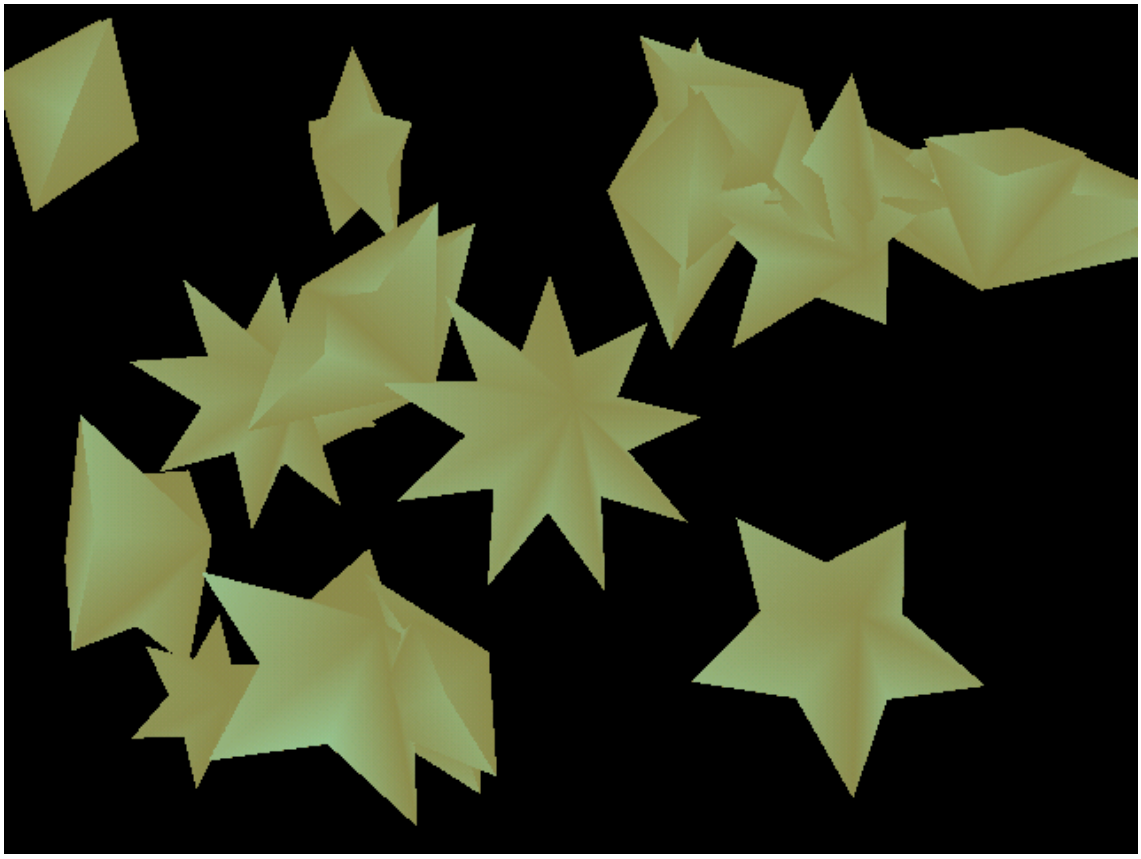

Figura x: resultado final de la práctica

El resultado final es el siguiente: obtenemos en pantalla un número variado de estrellas en pantalla, las cuales se colocan en disposición aleatoria dentro del marco de la pantalla. El número de puntas de la estrella es completamente aleatorio, y el rango se encuentra entre las cuatro y las nueve puntas, aunque éste puede ser aumentado.

Las estrellas realizan un giro, cada una distinto y en un sentido u orientación diferente para cada caso. El radio de las estrellas también es diferente en cada estrella. Existe un efecto de iluminación que dota a las estrellas con una sensación de volumen.

Las estrellas girarán en pantalla hasta que ocurra algún evento que les despierte, como el movimiento del ratón, la pulsación de alguna tecla del teclado, o la recepción de un mensaje del sistema operativo.

6. Bibliografía

· Neider, J., Davis, T. y Woo, M., *OpenGL programming guide: the official guide to learning OpenGL, release 1,* Addison-Wesley, Reading, MA, EE UU, 1993.

· NeHe. *NeHe Productions.* <http://nehe.gamedev.net/>.

· Jouvie, J., *OpenGL lessons.* <http://jerome.jouvie.free.fr/OpenGl/index.php>