Informática Gráfica
Ingeniería en Informática
Curso 13-14. Práctica 3.3

Carácter: obligatorio.

Fecha de entrega: martes 7 de enero a las 23:55 (la revisión será el miércoles 8 en el laboratorio).

Objetivo: transformaciones afines 2D.

Descripción: se trata de modificar tu implementación de la práctica 2 en dos aspectos. Por una parte, incorporarás un nuevo tipo de obstáculo, las elipses. Por otro, modificarás el modo en que se renderiza la pelota.

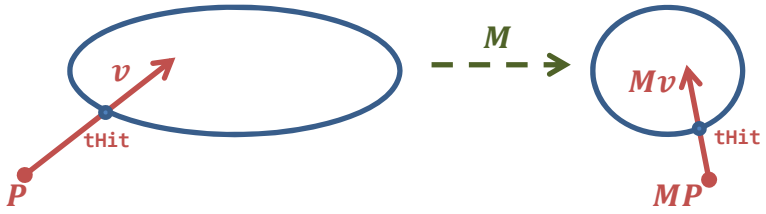
1. Una elipse está especificada por su centro $C = (C_x, C_x)$ y el tamaño de sus ejes W y H. Aunque formalmente los puntos de una elipse satisfacen la ecuación

$$\frac{(x-C_x)^2}{W^2} + \frac{(y-C_y)^2}{H^2} = 1,$$

usaremos transformaciones afines para entender que una elipse es una circunferencia transformada adecuadamente. Lo más sencillo es usar una circunferencia de radio 1 centrada en el origen. En lo que sigue, supongamos que M es la matriz que transforma la elipse en la circunferencia (M^{-1} transforma la circunferencia en elipse). Pista: quizá interese obtener M como una secuencia de transformaciones.

Nuestro enfoque tiene dos consecuencias:

- Para renderizar la elipse dibujarás una circunferencia (aproximada mediante un polígono regular), después de aplicar las transformaciones que correspondan sobre la matriz `GL_MODELVIEW`.
- El algoritmo de intersección pelota contra elipse se basará en el de intersección pelota contra círculo. Para ello debes aplicar M a la pelota, concretamente a la posición de su centro y a su velocidad lineal. El `tHit` que resulte al resolver la colisión entre la pelota transformada y el círculo será el instante de colisión entre la pelota original y la elipse.



Además, si n es la normal que resulta en tal intersección, ${M_R}^T n$ será la normal correspondiente a la colisión entre la pelota original y la elipse. Se entiende que M_R es la parte rotacional de M (se anula la componente traslacional), y T representa la trasposición de matrices:

$$M = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ 0 & 0 & 1 \end{pmatrix} \Rightarrow M_R = \begin{pmatrix} m_{11} & m_{12} & 0 \\ m_{21} & m_{22} & 0 \\ 0 & 0 & 1 \end{pmatrix} \Rightarrow {M_R}^T = \begin{pmatrix} m_{11} & m_{21} & 0 \\ m_{12} & m_{22} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

2. La pelota guardará su centro y su velocidad como hasta ahora, pero para renderizarla usaremos ahora una circunferencia de radio 1 centrada en el origen, aproximada como es habitual por un polígono regular. El método `draw` en la clase pelota comenzará por ello por una serie de transformaciones que coloquen la circunferencia con el tamaño y en la posición adecuada.
Además simularemos la rotación de la pelota a la hora de dibujarla. Dicha rotación no se considerará no al resolver las colisiones. Para ello, el polígono regular que representa la pelota debe completarse con un radio. Cuando dibujes la pelota aplicarás localmente la rotación que corresponda a la orientación actual de la pelota, que implementarás mediante un nuevo atributo θ. En cada ejecución del método `step`, incrementarás θ adecuadamente. Si se produce una colisión invertirás el sentido de la rotación.

$\boldsymbol{\theta = 0}$ $\boldsymbol{\theta = \pi/4}$ $\boldsymbol{\theta = \pi/2}$