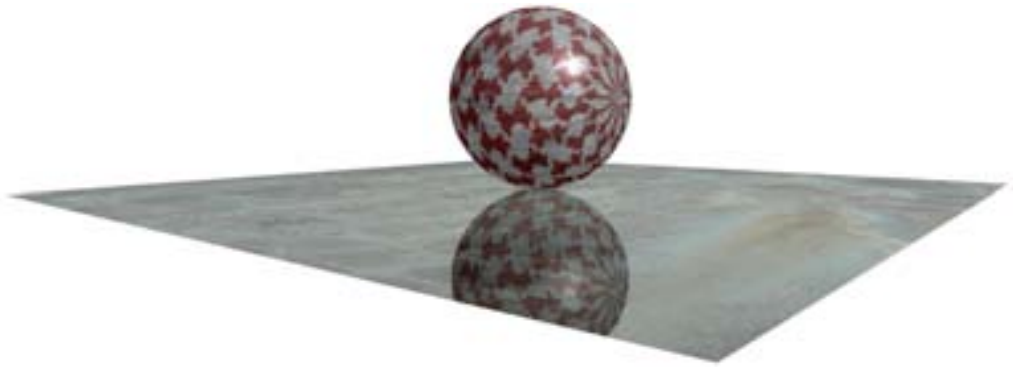


Tema 8: Realismo Visual

J. Ribelles

SIE020: Síntesis de Imagen y Animación
Institute of New Imaging Technologies, Universitat Jaume I

Contenido

Introducción

Introducción

- Este capítulo presenta tres tareas básicas en la búsqueda del realismo visual en imágenes sintéticas: trasparencia, reflejos y sombras.
- En la literatura se han presentado numerosos métodos para cada una de ellas.
- Aquí, se muestra una manera simple de realizarlas, consiguiendo una mejora importante en la calidad visual con poco esfuerzo de programación.

Trasparencia

Descripción

- Si hay algún objeto trasparente, el color de los píxeles cubiertos por dicho objeto depende de sus propiedades y las de los objetos que hayan detrás de él.
- Un método simple para incluir objetos trasparentes consiste en dibujar en primer lugar los objetos que sean opacos para después dibujar los trasparentes.
- El grado de trasparencia se suministra a la GPU como cuarta componente en las propiedades de material, conocida como componente *alfa*.
- Si *alfa* es 1 el objeto es totalmente opaco, y 0 significa que es trasparente:

$$C_{final} = alfa \cdot C_{fragmento} + (1 - alfa) \cdot C_{framebuffer} \quad (1)$$

Ejemplos

Figura: Cilindro trasparente con valores, de izquierda a derecha, $alfa = 0'1$, $alfa = 0'4$ y $alfa = 0'7$.

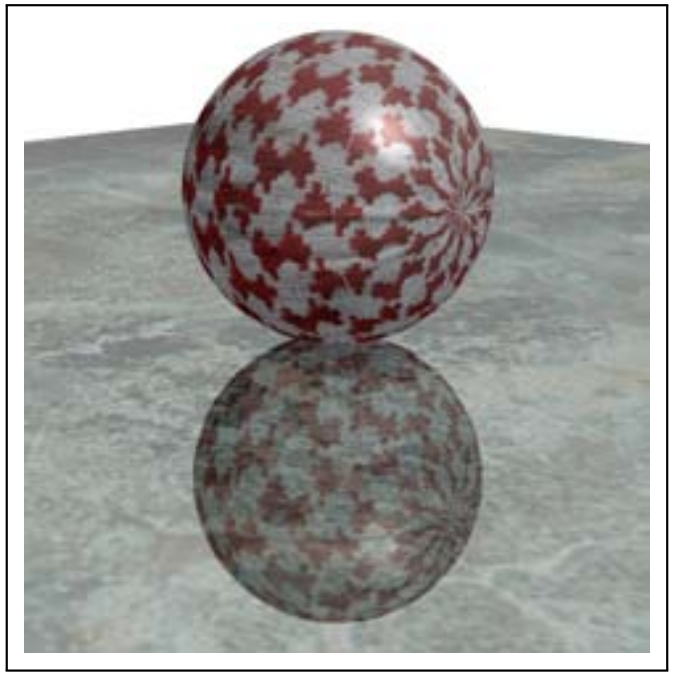
En OpenGL

```
// dibuja en primer lugar los objetos opacos
...

// activa el calculo de la trasparencia
glEnable (GL_BLEND);

// especifica la funcion de calculo
glBlendFunc (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

// impide la actualizacion del buffer de profundidad, por que?
glDepthMask (GL_FALSE);

// dibuja los objetos trasparentes
...

// inhabilita la trasparencia y
// permite actualizar el buffer de profundidad
glDisable   (GL_BLEND);
glDepthMask (GL_TRUE);
```

Reflejos

Descripción

- El cálculo del reflejo es realmente complejo, se han desarrollado métodos alternativos con muy buenos resultados visuales.
- Nos centramos en superficies planas reflejantes.
- El método consiste en dibujar la escena de forma simétrica respecto al plano que contiene al objeto reflejante. Dos tareas:
  - Obtener la transformación de simetría.
  - Evitar que la escena simétrica se observe fuera de los límites del objeto reflejante.



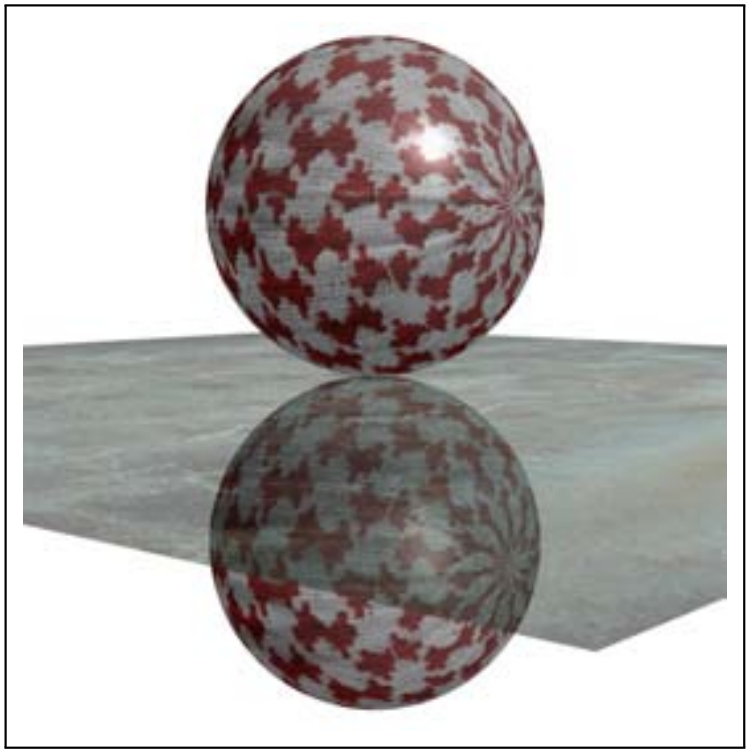
Simetría

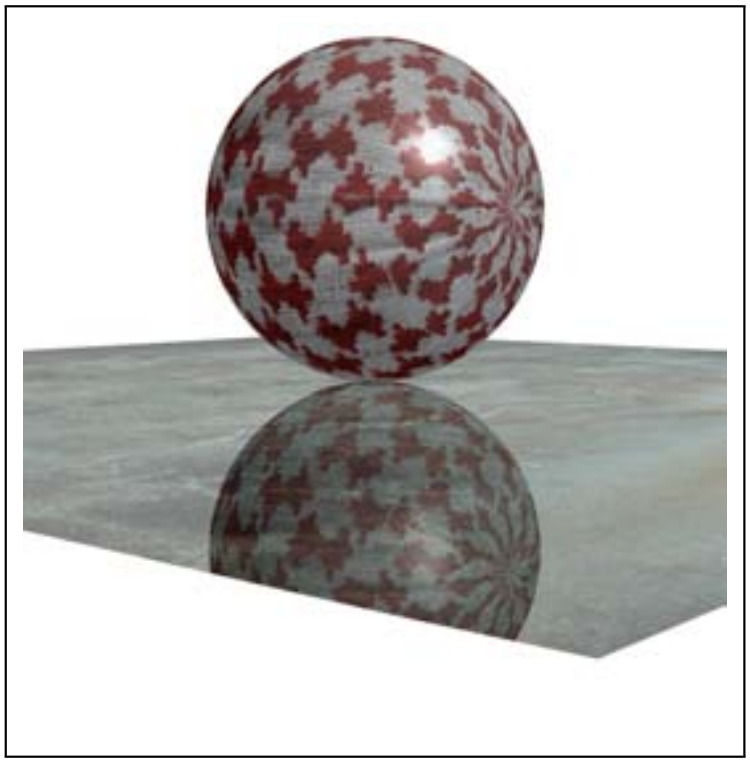
Operaciones

1. Trasladar el plano al origen.
2. Girarlo para hacerlo coincidir con, por ejemplo, el plano $Z = 0$.
3. Escalar con un factor de -1 en la dirección Z.
4. Deshacer el giro y la traslación.

$$M = \begin{pmatrix} 1-2V_x^2 & -2V_xV_y & -2V_xV_z & 2(P\cdot V)V_x \\ -2V_xV_y & 1-2V_y^2 & -2V_yV_z & 2(P\cdot V)V_y \\ -2V_xV_z & -2V_yV_z & 1-2V_z^2 & 2(P\cdot V)V_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (2)$$

Límites





Utilizar el buffer de plantilla

```
glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT | GL_STENCIL_BUFFER_BIT);

// Desactiva los buffers de color y profundidad
glDisable   (GL_DEPTH_TEST);
glColorMask (GL_FALSE, GL_FALSE, GL_FALSE, GL_FALSE);

// Establece como valor de referencia el 1
glEnable      (GL_STENCIL_TEST);
glStencilOp   (GL_REPLACE, GL_REPLACE, GL_REPLACE);
glStencilFunc (GL_ALWAYS,1,0xffffffff);

// Dibuja el objeto reflejante

// Activa de nuevo los buffers de profundidad y de color
glColorMask (GL_TRUE, GL_TRUE, GL_TRUE, GL_TRUE);
glEnable    (GL_DEPTH_TEST);

// Configura el buffer de plantilla
glStencilFunc (GL_EQUAL,1,0xffffffff);
glStencilOp   (GL_KEEP, GL_KEEP, GL_KEEP);

// Dibuja la escena reflejada

// Desactiva el test de plantilla
glDisable(GL_STENCIL_TEST);

// Dibuja la escena normal

// Dibuja el objeto reflejante con trasparencia
```

Sombras

Descripción

- La ausencia de sombras en la escena es algo que, además de incidir negativamente en el realismo visual de la imagen sintética, nos dificulta de manera importante su comprensión.
- Prácticamente cualquier método que nos permita añadir sombras, por sencillo que sea, puede ser más que suficiente para aumentar el realismo y que el usuario se sienta cómodo al observar el mundo 3D.
- Un método muy simple es el de sombras proyectivas:
  - Consiste en obtener la proyección del objeto situando la cámara en el punto de luz y estableciendo como plano de proyección aquel en el que queramos que aparezca su sombra.
  - El resultado de la proyección, al que llamamos objeto sombra, se dibuja como un objeto más de la escena pero sin propiedades de material ni iluminación, simplemente de color oscuro.

Matriz

Dada una fuente de luz L y un plano de proyección $N \cdot x + d = 0$ la matriz de proyección M es la siguiente.

$$M = \begin{pmatrix} N \cdot L + d - L_x N_x & -L_x N_y & -L_x N_z & -L_x d \\ -L_y N_x & N \cdot L + d - L_y N_y & -L_y N_z & -L_y d \\ -L_z N_x & -L_z N_y & N \cdot L + d - L_z N_z & -L_z d \\ -N_x & -N_y & -N_z & N \cdot L \end{pmatrix} \quad (3)$$

Problemas

Este método presenta una serie de problemas

- Como el objeto sombra es coplanar con el plano que se ha utilizado para el cálculo de la proyección, habría que añadir un pequeño desplazamiento a uno de ellos para evitar el efecto conocido como *stitching*. OpenGL proporciona la orden *glPolygonOffset* para especificar el desplazamiento que se sumará a la profundidad de cada fagmento siempre y cuando se haya habilitado con *glEnable(GL_POLYGON_OFFSET_FILL)*.
- Hay que controlar que el objeto sombra no vaya más allá de la superficie sobre la que recae, por ejemplo, usando el buffer de plantilla.
- Las sombras son muy oscuras, pero utilizando trasparencia se puede conseguir un resultado mucho más agradable al dejar entrever el plano sobre el que se asientan.

Ejemplo