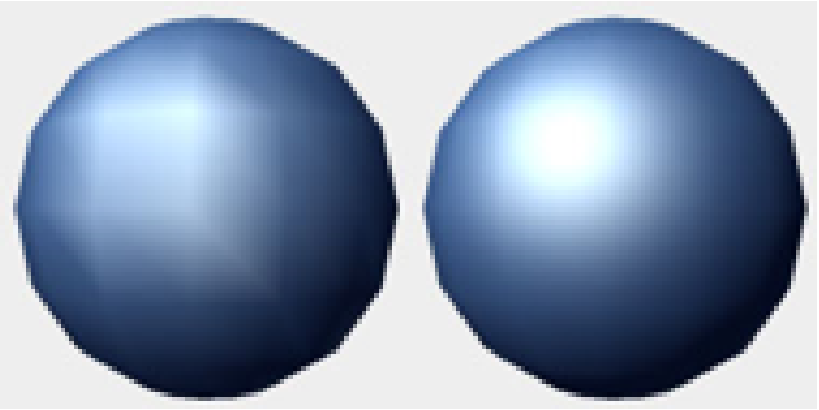
DOCUMENTACIÓN DE LA IMPLEMENTACIÓN:

Perfragment



Per-vertex lighting

Per-fragment lighting

informatika fakultatea

facultad de informática

eman ta zabal zazu

Universidad del País Vasco

Euskal Herriko Unibertsitatea

Iker Pintado García

Índice

1. Introducción

La técnica de *perfragment* se basa en interpolar para cada fragmento las normales de los vértices que lo componen y calcular el color en cada píxel, en vez de en cada vértice y luego interpolarlo.

Esta técnica consigue un efecto bastante más realista, que la de *pervertex*, aunque tiene el inconveniente de que su coste es mayor.

2. Implementación

Se han tenido que implementar los aportes de los 3 tipos distintos de luces otra vez, pero en esta ocasión, a nivel de fragmento. En realidad la situación no cambia mucho con respecto a la de *pervertex*. El único cambio es que en el vertex shader, las variables varying serán distintas y en el fragment shader debemos implementar algo parecido a lo implementado en el vertex shader de *pervertex*:

```
.
.
.
varying vec3 f_position;       // camera space
varying vec3 f_viewDirection; // camera space
varying vec3 f_normal;         // camera space, se pasa la normal
                            //para interpolar
varying vec2 f_texCoord;

void main() {

      vec4 tmp_pos = modelToCameraMatrix * vec4(v_position, 1.0);
      f_position = tmp_pos.xyz;


      f_viewDirection = -1.0 * f_position;

      vec4 tmp_normal = modelToCameraMatrix * vec4(v_normal, 0.0);
      f_normal = tmp_normal.xyz;

      f_texCoord = v_texCoord;
      gl_Position = modelToClipMatrix * vec4(v_position, 1.0);
}
```

Una vez hecho esto, solo quedaría copiar las funciones y el método **main** del vertex shader de *pervertex* a nuestro fragment shader. El único cambio que habría que aplicar es el siguiente: En vez de escribir el valor de **f_color** en la variable varying correspondiende (como se hace en el **main** del vertex shader de *pervertex*) para que se interpole para el fragment shader (al estar en un fragment shader, las variables varying son únicamente de lectura), calculamos directamente el valor del color y lo pasamos a **gl_FragColor** directamente (multiplicando componente a componente con el color de la textura). Para facilitar la copia, podemos crear una variable temporal en el método **main**, con el nombre **f_color:**

```
.
.
.
void main() {
      vec4 f_color;
      .
      .
```

```
        .
        f_color = vec4(acum_dif + scene_ambient + acum_esp, 1.0);
        vec4 texColor;
        texColor = texture2D(texture0, f_texCoord);
        gl_FragColor = f_color * texColor;
}
```

Aplicando este único cambio, hemos conseguido implementar *perfragment* exitosamente basándonos en lo implementado para *pervertex*.

3. Resultado en ejecución

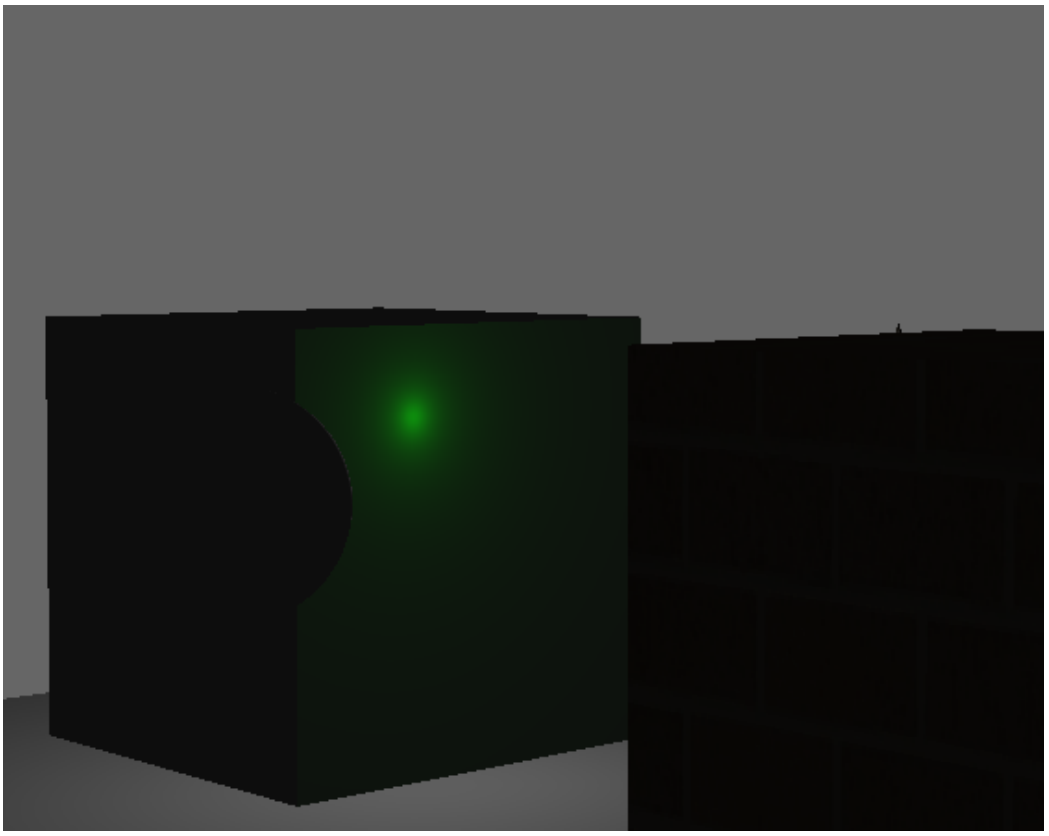


Fig. 1. Luz posicional atenuada sin quemazón

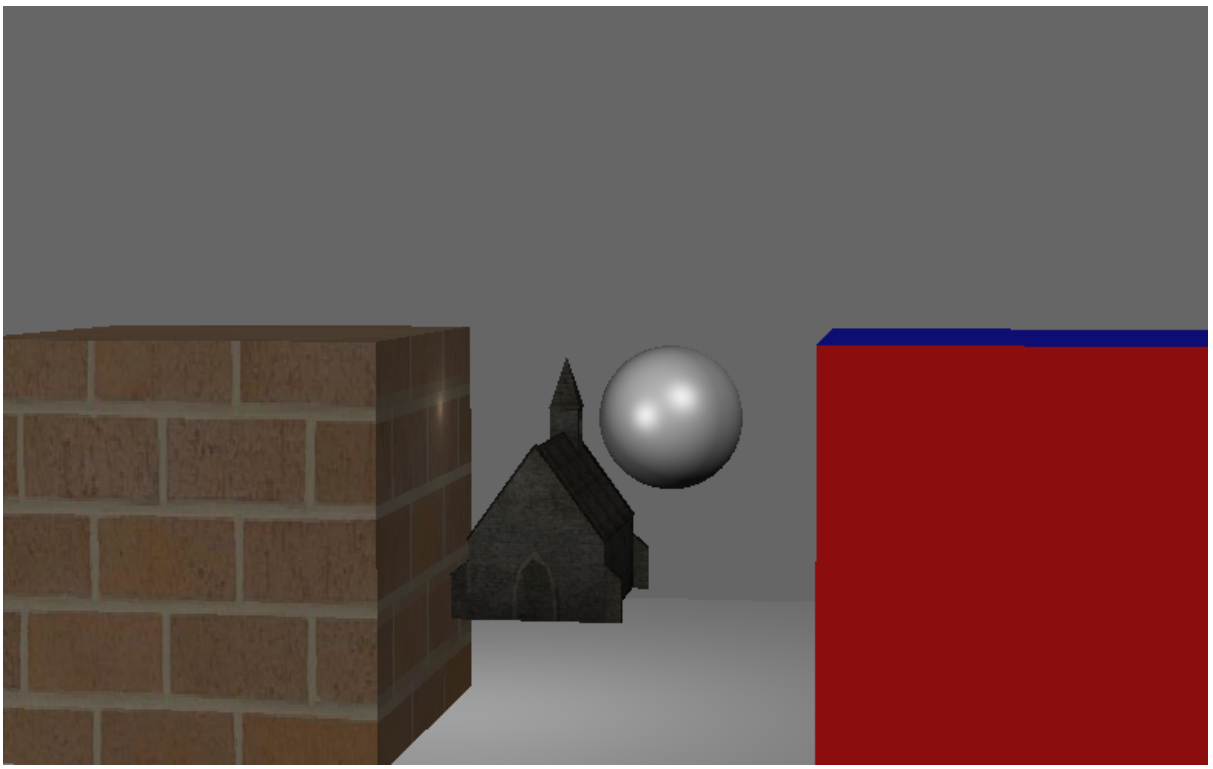


Fig. 2. Escena completa