DOCUMENTACIÓN DE LA IMPLEMENTACIÓN:

Multi-textura

informatika fakultatea

facultad de informática

eman ta zabal zazu

Universidad del País Vasco

Euskal Herriko Unibertsitatea

Iker Pintado García

Índice

1. Introducción

En el mundo real, los objetos pueden tener varias texturas simultáneamente. Para simular esta característica se ha implementado una técnica de multi-textura. Esta consiste en obtener dos texturas para un objeto y fusionarlas. Esta fusión se hace mediante la ponderación del valor de las dos texturas; es decir, a cada una de las texturas se le multiplicará un número menor que 1 y, la suma de estos dos números deberá ser igual a 1.

Esta técnica se ha aplicado sobre una esfera, haciéndola parecer el planeta tierra y su atmósfera. Una de las texturas es el mapamundi y, la otra, es una serie de nubes que representan la atmósfera. A esta última textura se le añadirá una pequeña animación sobre el eje x para simular el movimiento que la atmósfera real tendría sobre la tierra.

2. Implementación

Para implementar esta característica, se ha seguido el esquema dirigido de egela. Lo primero que se indica en este documento es que se debe asignar la segunda textura si esta tiene la competencia **multitex**. Todo esto ha sido implementado en el fichero ***shader.cc***, en la función **beforedraw**. Al final de la función hay una cláusula condicional en la que se entra si el puntero al material no es nulo. Dentro de esta cláusula, al final del todo, se ha escrito el código que implementa la especificación dada. Estas líneas son las siguientes:

```
if (this->has_capability("multitex")) {
      Texture *tex2 = mat->getTexture(1);
      if (tex2 != 0) {
            tex2->bindGLUnit(Constants::gl_texunits::rest);
            this->send_uniform("texture1",
                              Constants::gl_texunits::rest);
      }
}
```

Como podemos observar, si se tiene la capacidad **multitex**, se obtendrá la segunda textura y si esta no es nula, se asigna en la unidad dada por **Constants::g1_texunits::rest**. El siguiente paso ha sido la fusión ponderada de las texturas en el fragment shader correspondiente:

```
vec4 texColor = 0.5 * texture2D(texture0, f_texCoord) + 0.5 *
                texture2D(texture1, f_texCoord);

gl_FragColor = f_color * texColor;
```

Con todas estas modificaciones, el efecto de multi-textura ya estaría correctamente implementado. Pero, en nuestro caso, faltaría añadir la animación de las nubes. Lo primero que se debe hacer para esto es, incluir en el **RenderState** el atributo **m_cloudsOffset**, junto a sus correspondientes funciones **set** y **get**. Esto se debe implementar tanto en **renderState.h** como **renderState.cc**:

```
//.h en la sección public
      void setCloudsOffset(float v);
      float getCloudsOffset();
.
.
.
//en la sección private
      float m_cloudsOffset;

//En el .c
void RenderState::setCloudsOffset(float v){m_cloudsOffset = v;}
float RenderState::getCloudsOffset(){return m_cloudsOffset;}
```

Una vez hecho esto, necesitamos implementar la animación modificando el valor de **m_cloudsOffset** progresivamente en la función **animate** de **browser.cc**:

```
void animate(int value) {
      static float c = 0.0;
      .
      .
      .
      if (runAnimation) {
            if(c > 1.0){
                  c = 0.0;
            }else{
                  c += 0.001;
            }
            RenderState::instance()->setCloudsOffset(c);
            glutPostRedisplay();
      }
      .
      .
      .
}
```

Para finalizar, debemos volver al fragment shader y a la función **beforedraw** para plasmar esta animación de manera definitiva:

```
//beforedraw
if (this->has_capability("multitex")) {
      Texture *tex2 = mat->getTexture(1);
      if (tex2 != 0) {
            .
            .
            .
            this->send_uniform("uCloudOffset",
                              rs->getCloudsOffset());
      }
}

//fragment shader
.
.
.
vec2 desfase = vec2(uCloudOffset, 0.0);
      texColor = 0.5 * texture2D(texture0, f_texCoord) + 0.5 *
                  texture2D(texture1, f_texCoord + desfase);
.
.
.
```

3. Resultado en ejecución

Ahora se mostrarán varias capturas de la ejecución del programa jugando con distintas ponderaciones:

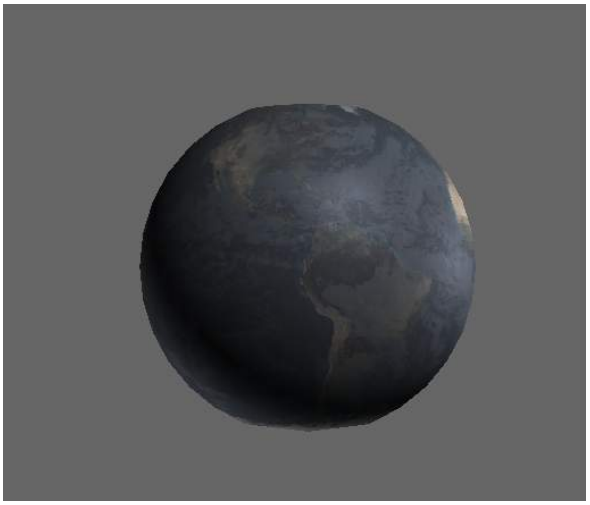

Fig. 1. 0.5 * nubes + 0.5 * mapamundi

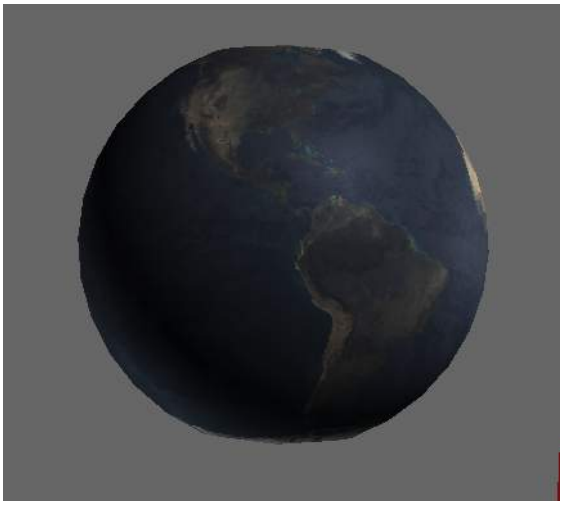

Fig. 2. 0.25* nubes + 0.75 * mapamundi

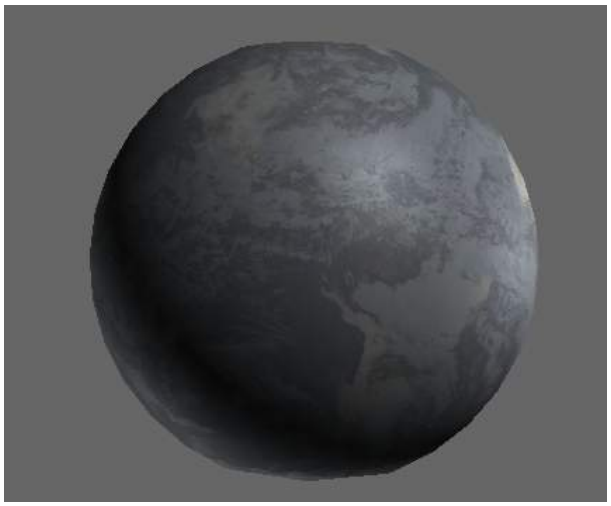

Fig. 3. 0.75 * nubes + 0.25 * mapamundi

Fig. 4. 0.0 * nubes + 1.0 * mapamundi

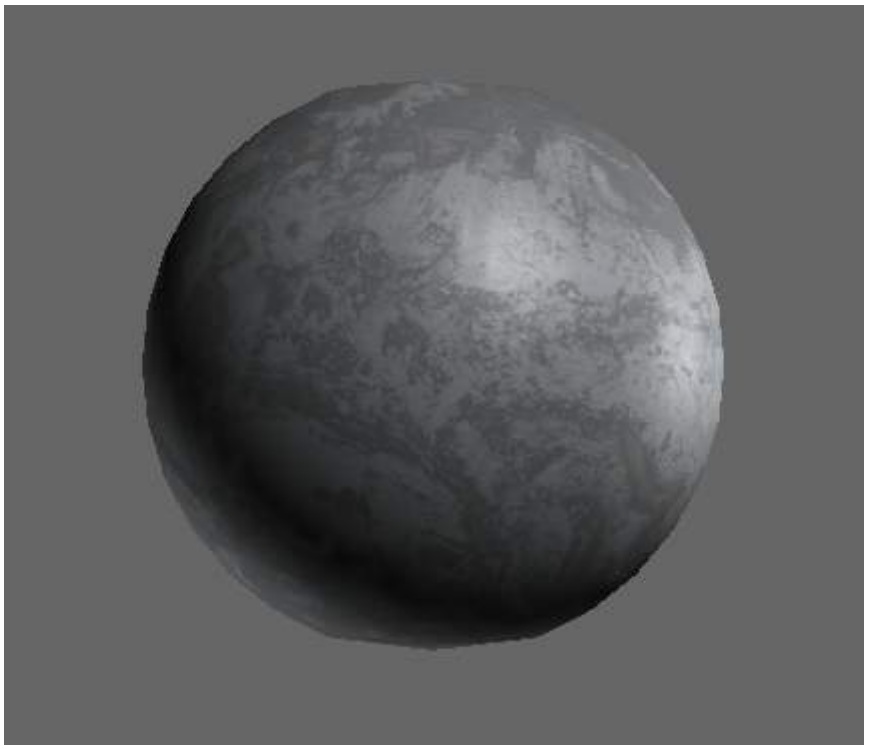

Fig. 5. 1.0 * nubes + 0.0 * mapamundi