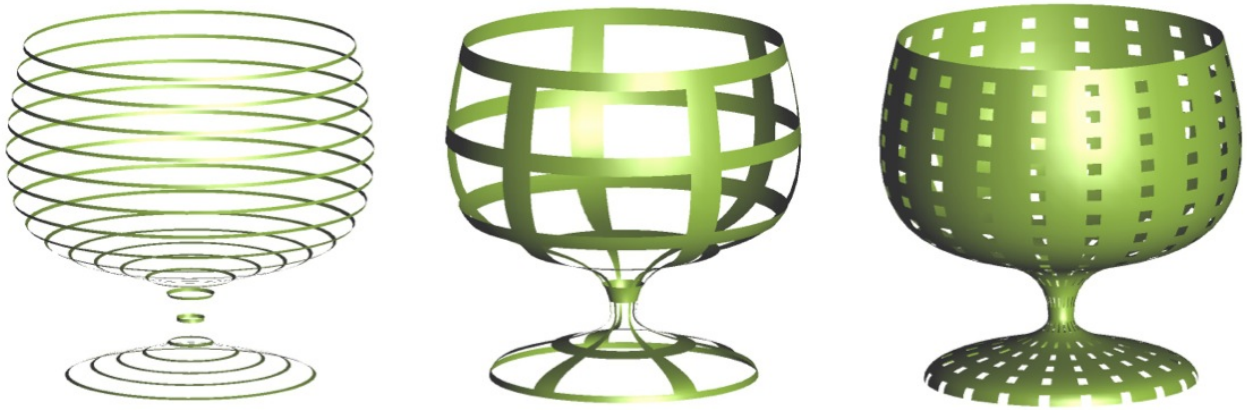


Tema 6: Texturas Procedurales

José Ribelles

Departamento de Lenguajes y Sistemas Informáticos, Universitat Jaume I

VJ1221 - Informática Gráfica

Contenido

Hoy veremos...

Introducción

Texturas procedurales: definición y ventajas

- Sin duda una de las grandes ventajas de las GPU programables.
- El Shader determina el aspecto de la superficie de un objeto principalmente mediante un algoritmo.
- Principales ventajas comparado con utilizar imágenes 2D:
  - Consumo de memoria menor. Y muchísimo menor en el caso de texturas 3D.
  - No hay resolución. Por mucho que te acerques a un objeto nunca verás los problemas habituales en texturas 2D.
  - Los aspectos clave se implementan mediante parámetros pudiendo obtener multitud de efectos distintos. Por ejemplo las copas de la primera traspa.

Introducción

¿Desventajas?

- Bueno, quizá el algoritmo no sea sencillo de escribir o no esté a nuestro alcance.
- Es un proceso más lento que acceder a una imagen.
- Problemas de aliasing, más difíciles de solucionar que utilizando imágenes donde el hardware implementa métodos de filtrado, mipmapping, etc.
- Los problemas de precisión numérica puede hacer que el resultado sea diferente para cada GPU.

Introducción

Entonces, ¿qué?

- Normalmente deberás utiliza ambos métodos de texturas.
- Observa la imagen de ejemplo.

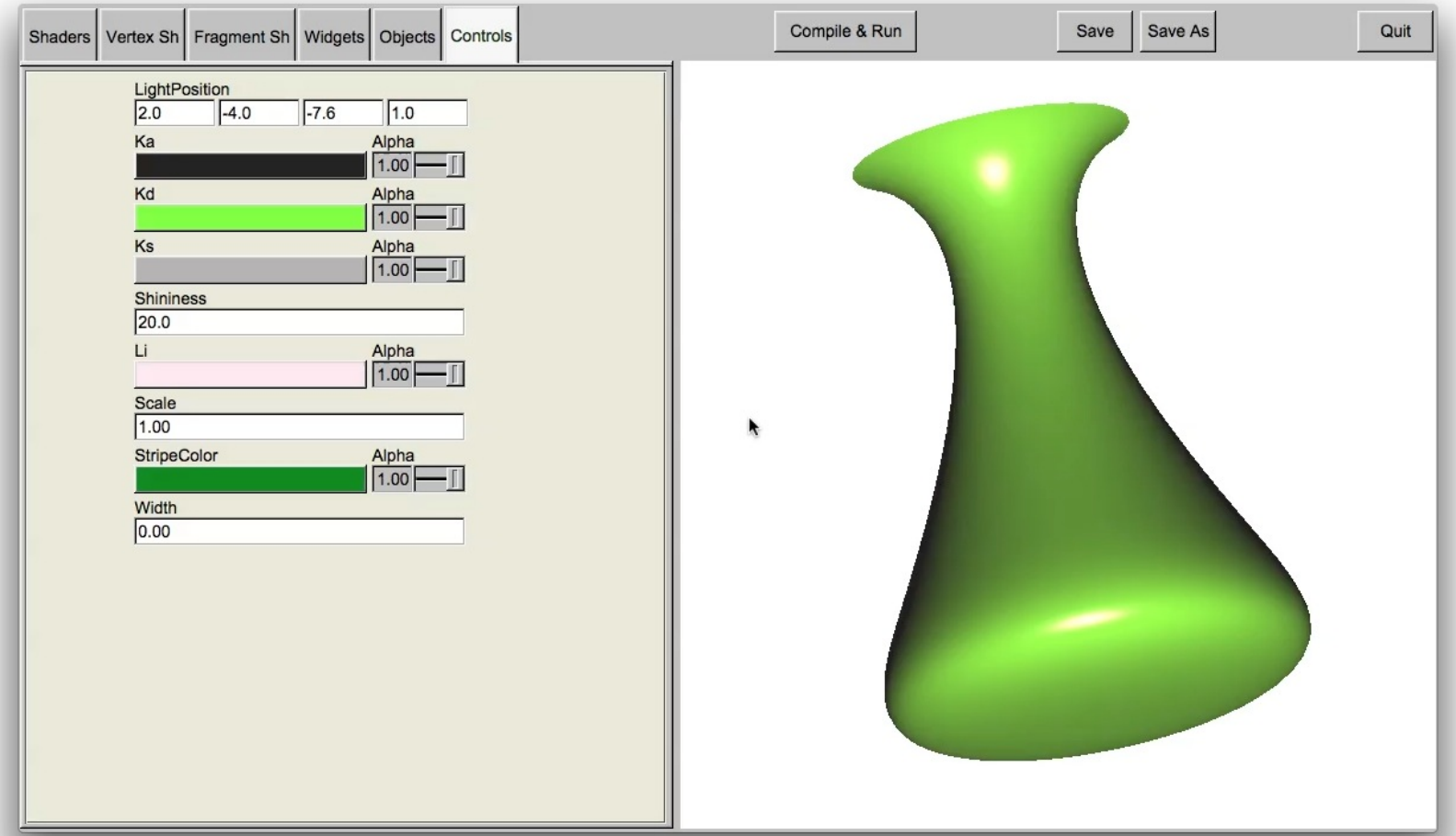
Ejercicio

Señala los aspectos que consideres que se corresponden con ventajas de las texturas procedurales frente a texturas de imagen 2D:

- La resolución.
- La parametrización de aspectos clave.
- El consumo de memoria.
- La velocidad de proceso.

Hoy veremos...

Rayado


Shaders
Vertex Sh
Fragment Sh
Widgets
Objects
Controls
Compile & Run
Save
Save As
Quit
LightPosition
2.0
-4.0
-7.6
1.0
Ka
Alpha
1.00
Kd
Alpha
1.00
Ks
Alpha
1.00
Shininess
20.0
Li
Alpha
1.00
Scale
1.00
StripeColor
Alpha
1.00
Width
0.00

El Shader para Rayado

Listado 1: Shader para Rayado

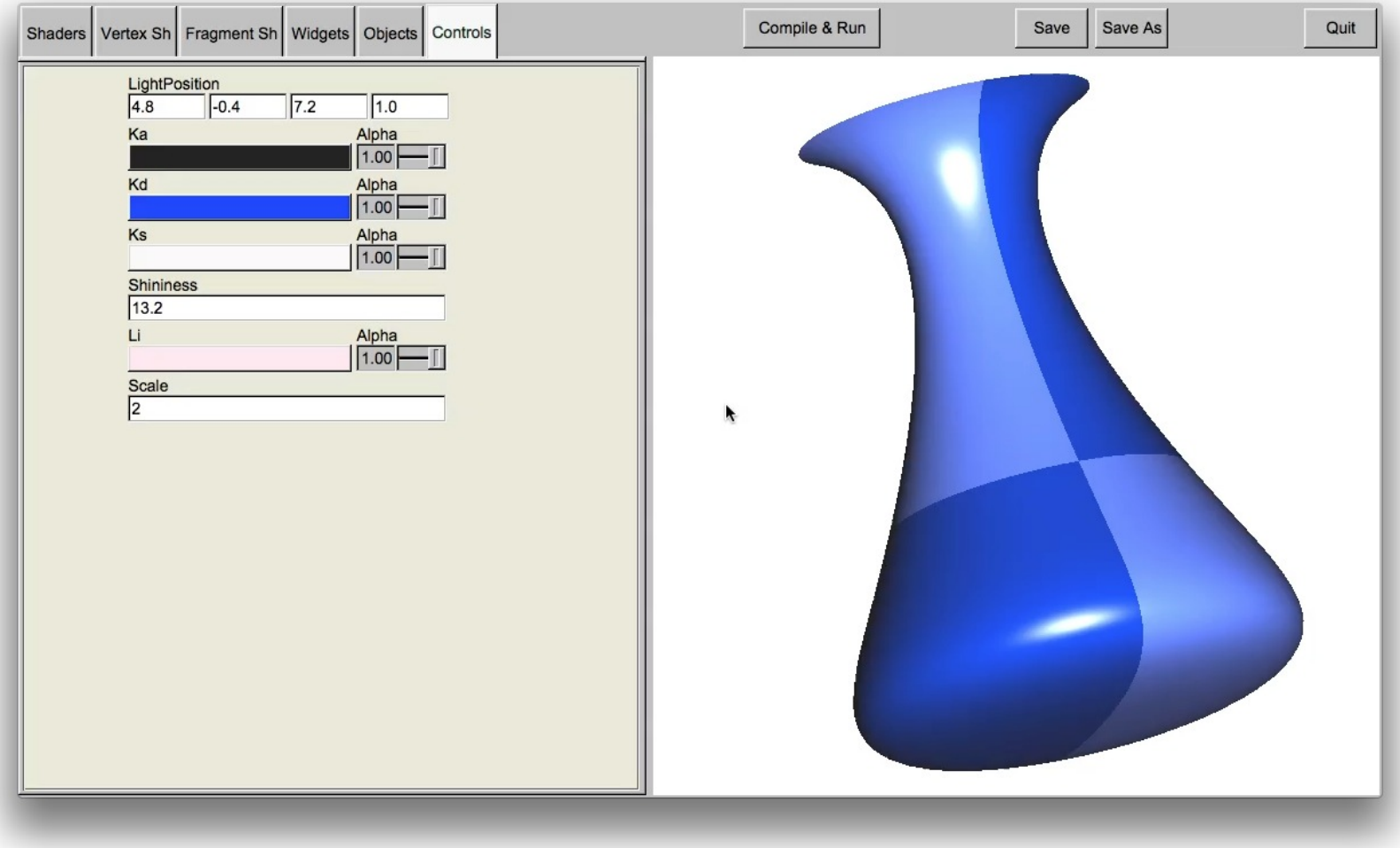
```
// Vertex shader
...
out float texCoord;

void main()  {
  ...
  texCoord = VertexTexcoords.t;
}

// Fragment shader
...
uniform vec3  StripeColor;     // color de la raya
uniform float Scale;           // numero de rayas
uniform float Width;           // ancho de la raya

in  float texCoord;
out vec4  fragmentColor;

void main() {

  float scaledT = fract(texCoord * Scale);
  float       s = step (Width, scaledT);
  vec3    newKd = mix (StripeColor, Kd, s);

  fragmentColor = vec4(phong(newKd), 1.0);

}
```

Ejercicio

En el Shader de rayado se realiza la siguiente operación: float s = step(Width, scaledT);. ¿Qué valores puede tomar la variable 's' como resultado de la ejecución de esa operación?

1. Un valor en el rango [0..1].
2. Cero o uno.
3. Un valor en el rango [Width..scaledT].
4. Width o scaledT.

Checkers


Shaders
Vertex Sh
Fragment Sh
Widgets
Objects
Controls
Compile & Run
Save
Save As
Quit
LightPosition
4.8
-0.4
7.2
1.0
Ka
Alpha
1.00
Kd
Alpha
1.00
Ks
Alpha
1.00
Shininess
13.2
Li
Alpha
1.00
Scale
2

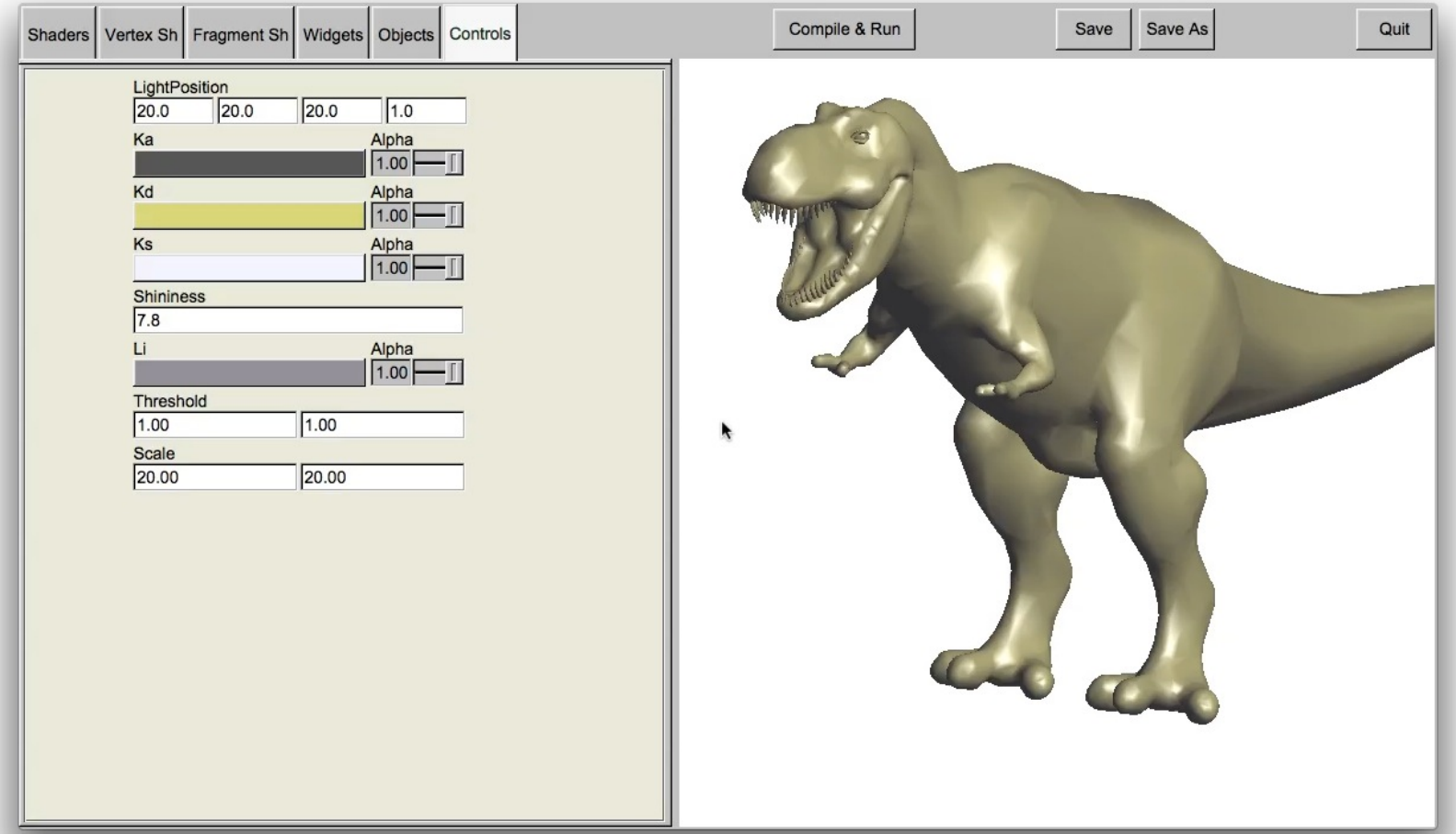
El Shader para Checkers

Listado 2: Shader para Checkers

```
// Vertex shader
...
out vec2 texCoord;

void main()  {
  ...
  texCoord = VertexTexcoords;
}

// Fragment shader
...
uniform float Scale;          // numero de rayas

in  vec2 texCoord;
out vec4 fragmentColor;

void main() {

  float row = floor( texCoord.s * Scale );
  float col = floor( texCoord.t * Scale );

  float res   = mod( row + col , 2.0);
  vec3  newKd = Kd + (res * 0.4);

  fragmentColor = vec4(phong(newKd), 1.0);

}
```

Hoy veremos...

Enrejado


Shaders
Vertex Sh
Fragment Sh
Widgets
Objects
Controls
Compile & Run
Save
Save As
Quit
LightPosition
20.0
20.0
20.0
1.0
Ka
Alpha
1.00
Kd
Alpha
1.00
Ks
Alpha
1.00
Shininess
7.8
Li
Alpha
1.00
Threshold
1.00
1.00
Scale
20.00
20.00

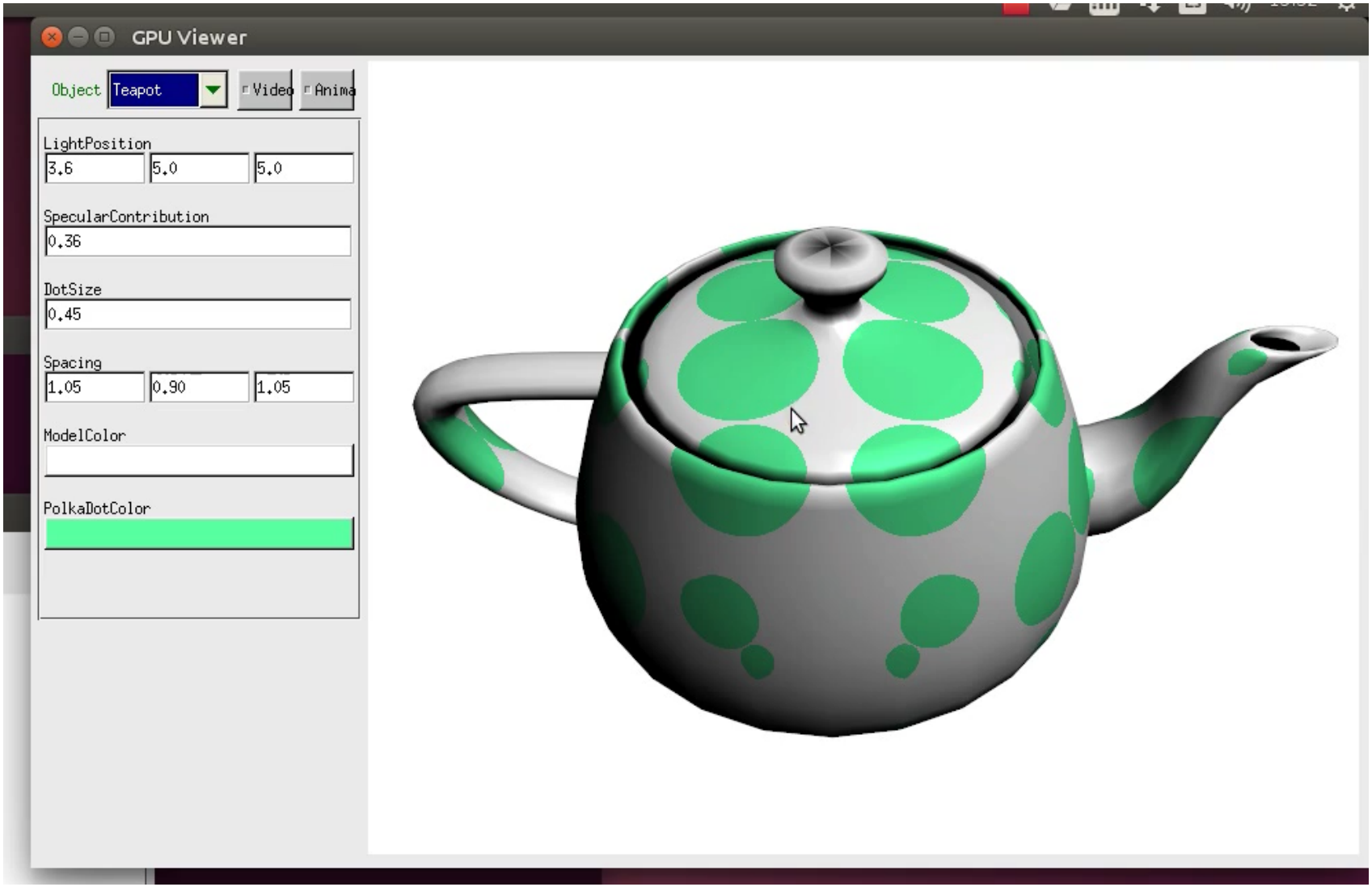
El Shader para Enrejado

Listado 3: Shader para Enrejado

```
// Vertex shader
...
out vec2 texCoord;

void main()  {
  ...
  texCoord = VertexTexcoords;
}

// Fragment shader
...
uniform vec2 Scale;
uniform vec2 Threshold;

in  vec2 texCoord;
out vec4 fragmentColor;

void main() {

  float ss = fract(texCoord.s * Scale.s);
  float tt = fract(texCoord.t * Scale.t);

  if ((ss > Threshold.s) && (tt > Threshold.t))
    discard;

  fragmentColor = vec4(phong(), 1.0);

}
```

Hoy veremos...

Otros ejemplos



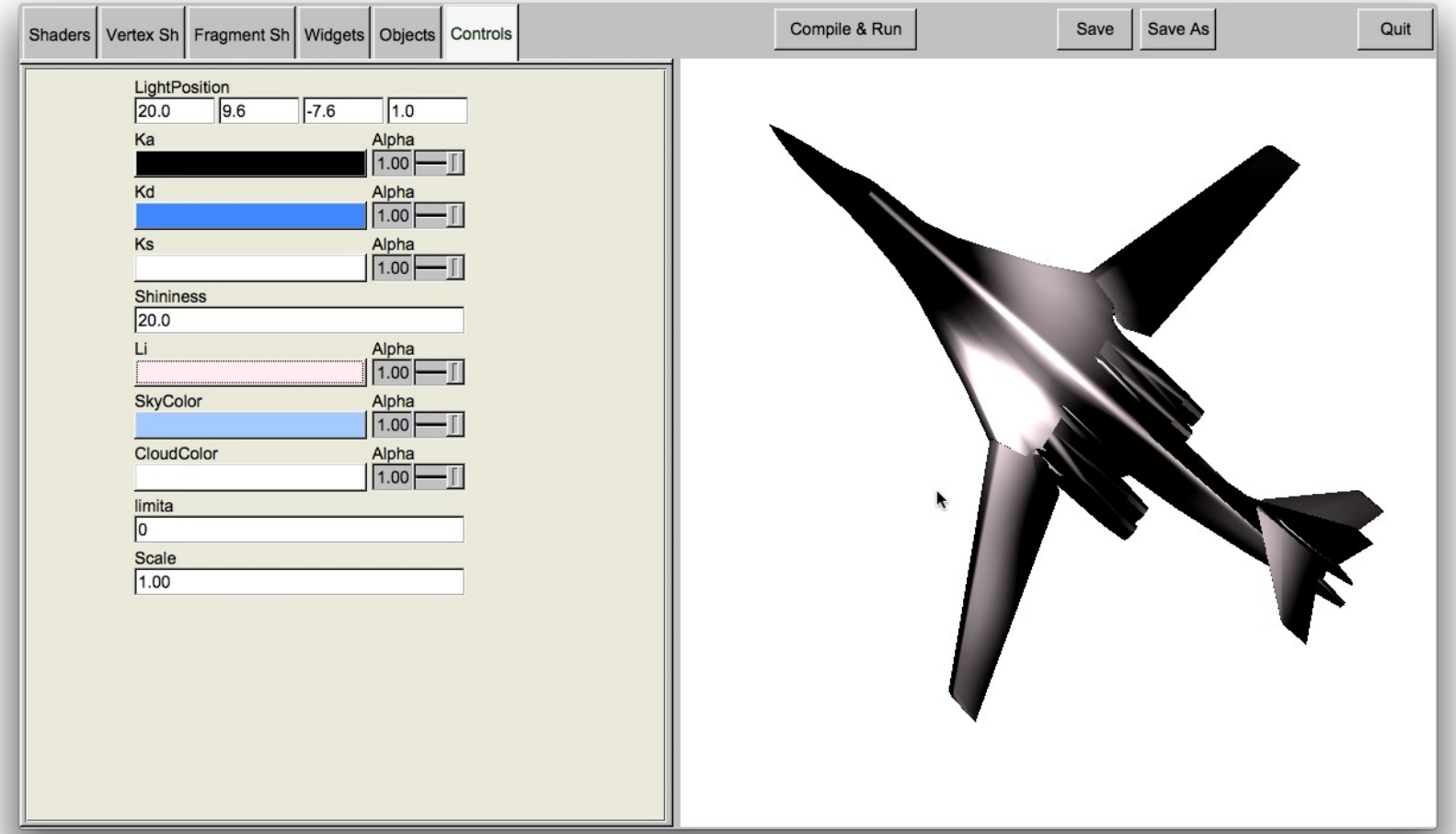
Hoy veremos...

Ruido

Definición

- Ken Perlin fue el precursor, 1985.
- Perlin reconoció la imperfección de las perfectas imágenes generadas por computador.
- La función ideal de ruido:
  - Contínua y aparentar aleatoriedad.
  - Repetible, para la misma entrada genera siempre la misma salida.
  - Rango bien definido (por ejemplo [-1..1] ó [0..1]).
  - Debe poder ser definida para distintas dimensiones: 1, 2, 3, 4 o más.

Ejemplo de la función de ruido de Perlin



Aplicaciones

¿De qué me sirve? La lista es interminable

- Visualizar fenómenos naturales (nubes, fuego, humo, efectos del viento, etc.)
- Visualizar materiales naturales (mármol, granito, madera, montañas, etc.)
- Visualizar materiales hechos por el hombre (asfalto, cemento, etc.)
- Añadir imperfecciones a modelos perfectos (suciedad, polvo, tierra, manchas, etc.)
- Añadir imperfecciones a patrones perfectos (bultos, rotos, variaciones de color)
- Añadir imperfecciones a periodos de tiempo, movimiento ,etc.

Ruido en un Shader

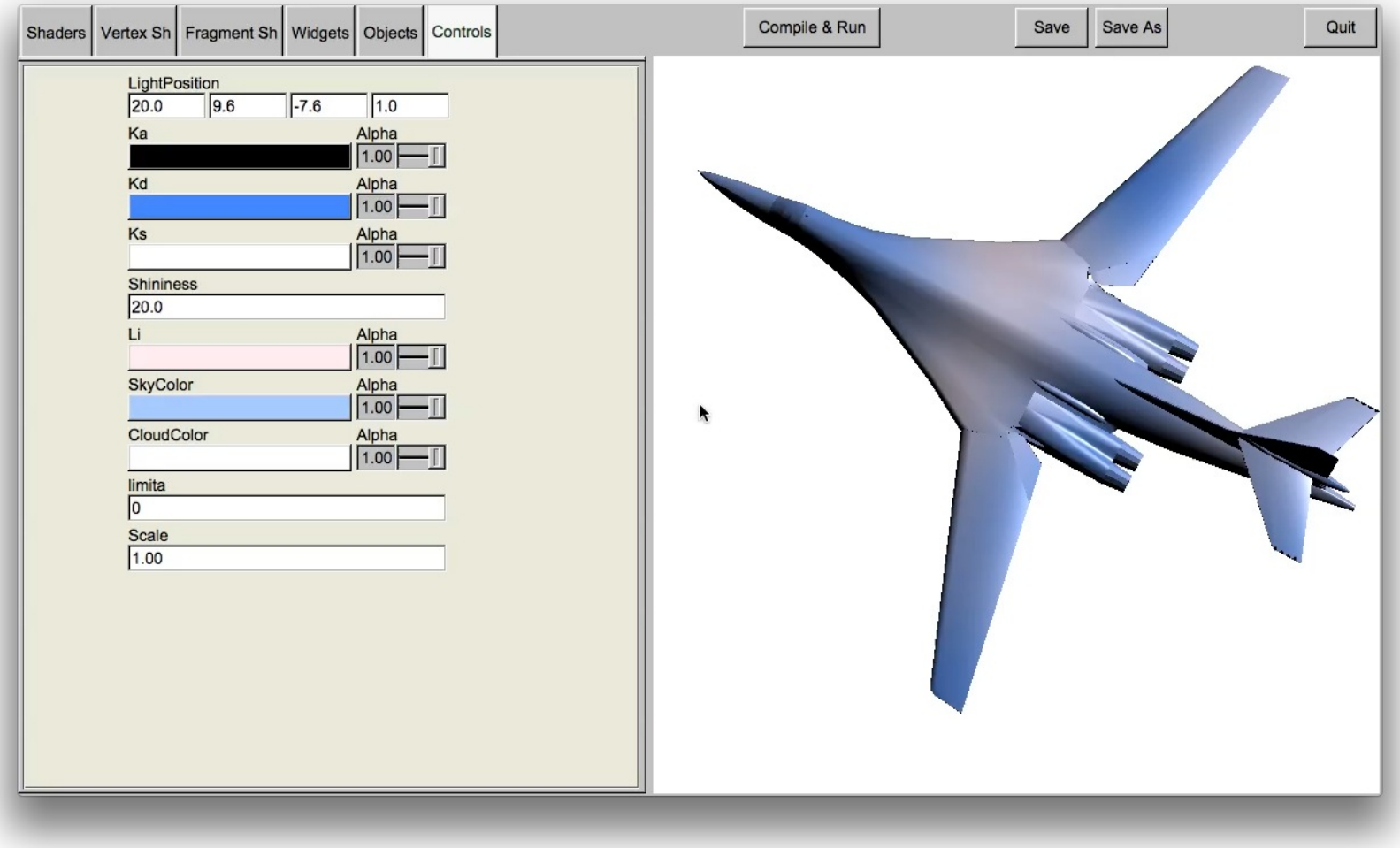
Posibilidades

- Escribir tu propia función de ruido en el Shader.
- Almacenar el resultado de la función de ruido en una textura.

¿Cuál es la mejor opción?

- Las texturas de ruido aseguran un comportamiento independiente del hardware y es mucho más rápido.
- Si utilizas tu propia función, es más caro computacionalmente pero no tienes los inconvenientes de las texturas discretas.

Nubes

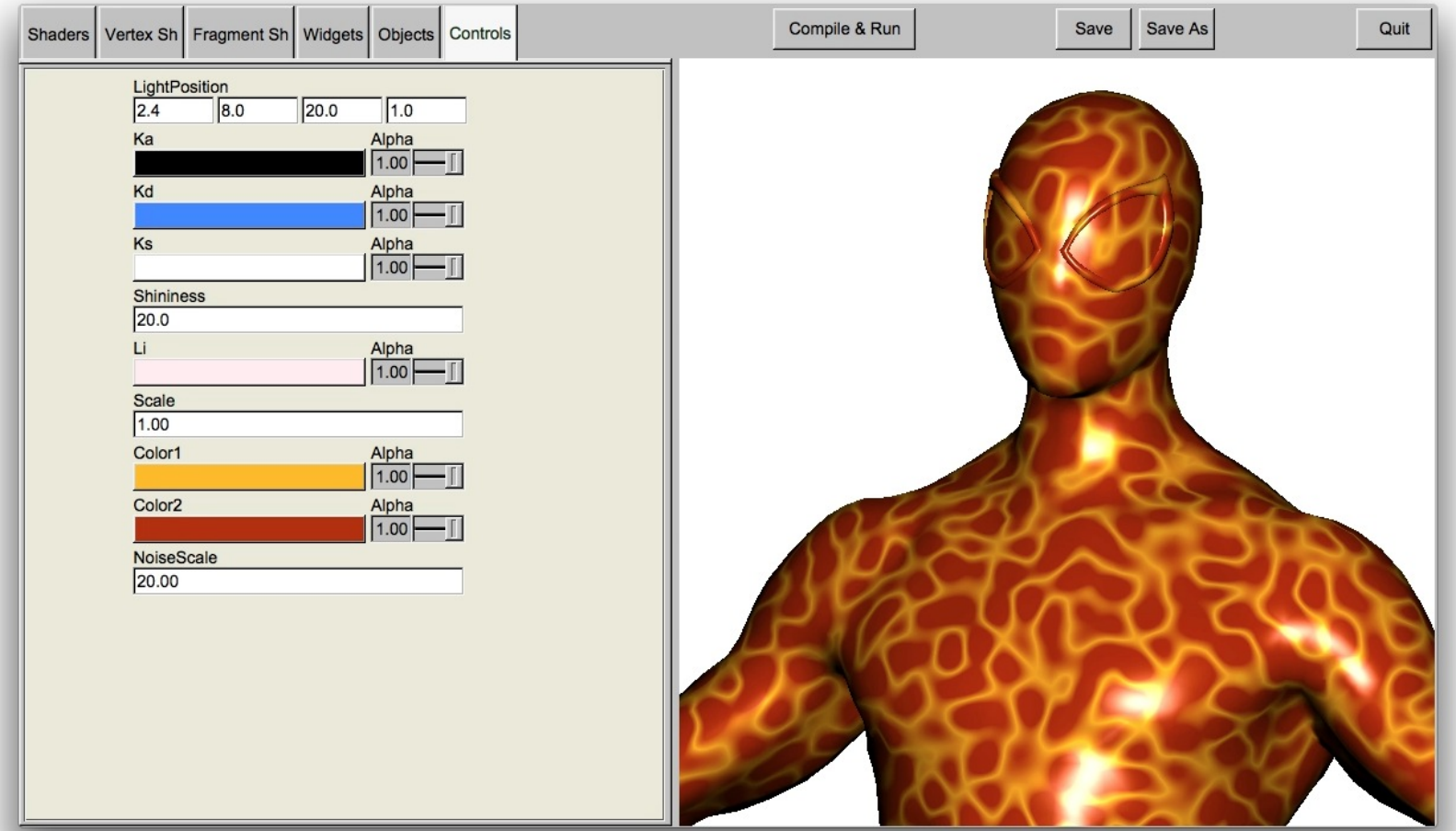

Shaders
Vertex Sh
Fragment Sh
Widgets
Objects
Controls
Compile & Run
Save
Save As
Quit
LightPosition
20.0
9.6
-7.6
1.0
Ka
Alpha
1.00
Kd
Alpha
1.00
Ks
Alpha
1.00
Shininess
20.0
Li
Alpha
1.00
SkyColor
Alpha
1.00
CloudColor
Alpha
1.00
limita
0
Scale
1.00

El Shader para Nubes

Listado 4: Shader para Nubes

```
// Vertex shader
...
uniform float Scale;
out     vec3  MCposition;

void main()  {
  ...
  MCposition  = VertexPosition * Scale;
}

// Fragment shader
...
uniform vec3 SkyColor;
uniform vec3 CloudColor;

in  vec3 MCposition;
out vec4 fragmentColor;

void main() {

  vec4 noisevec   = noise(MCposition);
  float intensity = (noisevec[0] + noisevec[1] + noisevec[2] + noisevec[3] + 0.03125) * 1.5;
  vec3 newKd      = mix (SkyColor, CloudColor, intensity);

  fragmentColor   = vec4(phong(newKd), 1.0);

}
```

Superficie del Sol



El Shader para superficie solar

Listado 5: Shader para superficie solar

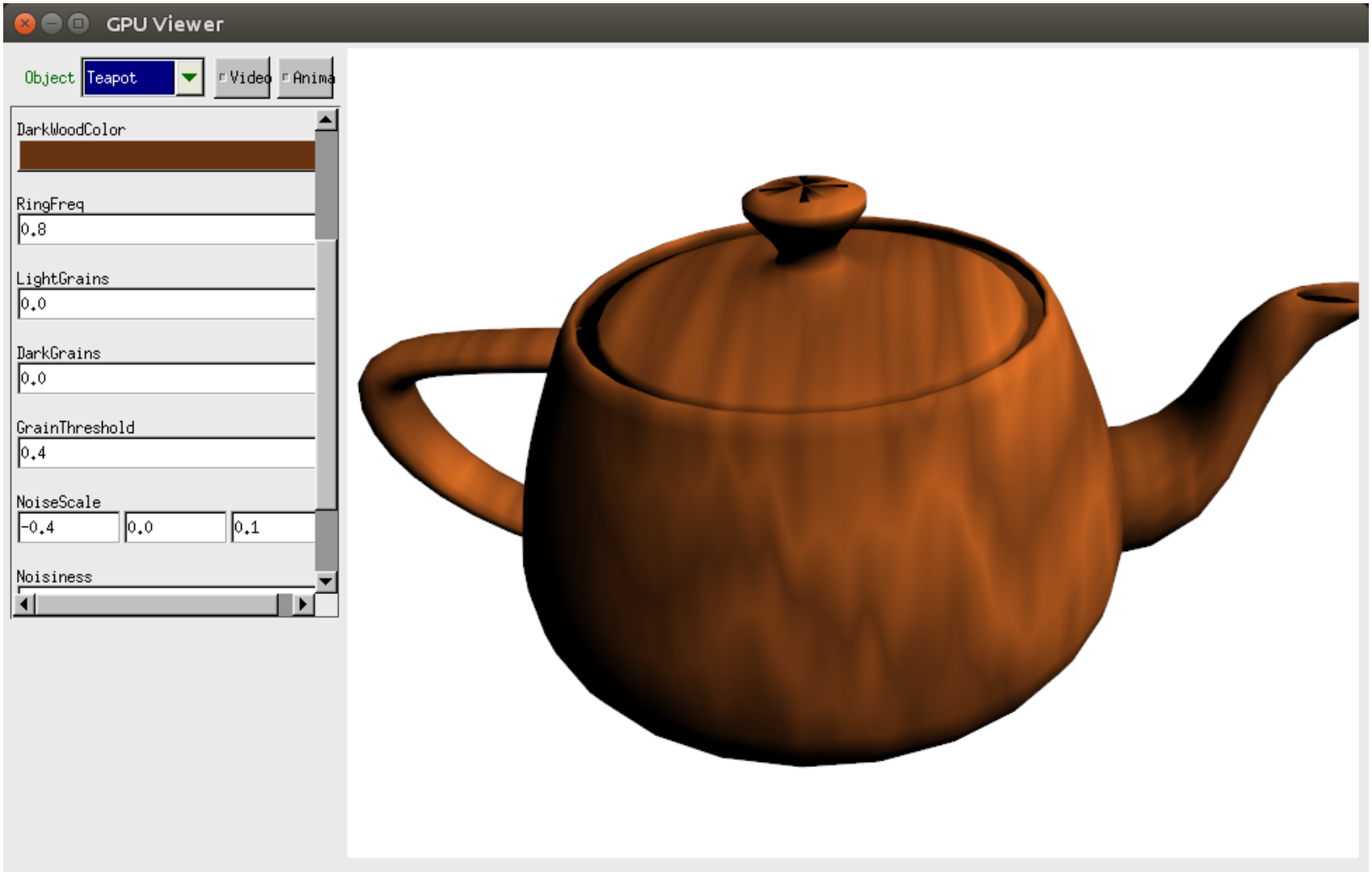
```
// Fragment shader
...
uniform vec3 Color1;
uniform vec3 Color2;

in  vec3 MCposition;
out vec4 fragmentColor;

void main() {

  vec4 noisevec   = noise(MCposition*NoiseScale);
  float intensity = abs(noisevec[0] - 0.25) +
                    abs(noisevec[1] - 0.125) +
                    abs(noisevec[2] - 0.0625) +
                    abs(noisevec[3] - 0.03125);
  intensity       = clamp(intensity * 6.0, 0.0, 1.0);
  vec3 newKd      = mix (Color1, Color2, intensity);

  fragmentColor   = vec4(phong(newKd), 1.0);

}
```

Otros ejemplos

Listado 6: Shader para mármol

```
// Fragment shader
...
uniform vec3 VeinColor;
uniform vec3 MarbleColor;

in  vec3 MCposition;
out vec4 fragmentColor;

void main() {

  vec4  noisevec  = noise(MCposition);
  float intensity = abs(noisevec[0] - 0.25) +
                    abs(noisevec[1] - 0.125) +
                    abs(noisevec[2] - 0.0625) +
                    abs(noisevec[3] - 0.03125);
  float sineval   = sin(MCposition.y * 6.0 + intensity * 12.0) * 0.5 + 0.5;
  vec3  newKd     = mix(VeinColor, MarbleColor, sineval);

  fragmentColor   = vec4(phong(newKd), 1.0);

}
```

Otros ejemplos



Ejercicio

Si deseas utilizar texturas procedurales, ¿estás de acuerdo en que nunca es necesario proporcionar coordenas de textura como atributo por cada vértice?

1. No, depende de la propia función de textura, como por ejemplo el caso del Enrejado en el que sí que hace falta.
2. Si, es precisamente una de las prinicipales ventajas frente a utilizar texturas de imagen 2D.
3. No, en absoluto, siempre voy a necesitar proporcionar las coordenadas de textura como atributo para cada vértice.
4. Si, nunca necesito que se proporcionen como atributo pero aún así debo obtenerlas en el vertex shader.