GLSL Shaders

Alex Pytel
October 29th, 2010

Basic Shading
A Minimal Shader
Result
Diffuse Lighting
Result
PV Specular 1
PV Specular 2
Result 1
Result 2
PP Specular 1
PP Specular 2
Result
Normalization
2Sided Lighting 1
2Sided Lighting 2
Texturing
Result
Inventive Shading
Conclusion

Basic Shading

A Minimal Shader

Basic Shading
A Minimal Shader
Result
Diffuse Lighting
Result
PV Specular 1
PV Specular 2
Result 1
Result 2
PP Specular 1
PP Specular 2
Result
Normalization
2Sided Lighting 1
2Sided Lighting 2
Texturing
Result
Inventive Shading
Conclusion

Vertex shader

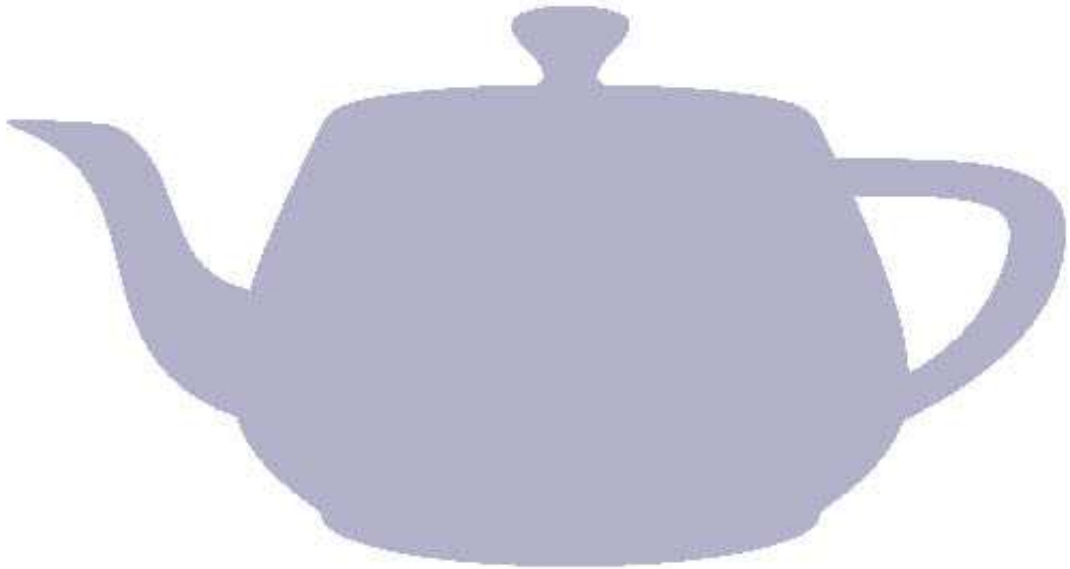
```
1  void main()
2  {
3      gl_Position = ftransform();
4  }
```

Fragment shader

```
1  void main()
2  {
3      gl_FragColor = gl_FrontMaterial.diffuse;
4  }
```

Result

Basic Shading
A Minimal Shader
Result
Diffuse Lighting
Result
PV Specular 1
PV Specular 2
Result 1
Result 2
PP Specular 1
PP Specular 2
Result
Normalization
2Sided Lighting 1
2Sided Lighting 2
Texturing
Result
Inventive Shading
Conclusion



Vertex shader

```
1  varying float albedo;
2
3  void main()
4  {
5      vec3 pos_EC = vec3(gl_ModelViewMatrix * gl_Vertex);
6      vec3 normal_EC = normalize(gl_NormalMatrix * gl_Normal);
7      vec3 light_EC = normalize(gl_LightSource[0].position.xyz -
           pos_EC);
8
9      albedo = dot(normal_EC, light_EC);
10
11     gl_Position = ftransform();
12 }
```

Fragment shader

```
1  varying float albedo;
2
3  void main()
4  {
5      gl_FragColor = vec4(albedo * gl_FrontMaterial.diffuse.rgb,
           1.0);
6  }
```

Result

Basic Shading
A Minimal Shader
Result
Diffuse Lighting
Result
PV Specular 1
PV Specular 2
Result 1
Result 2
PP Specular 1
PP Specular 2
Result
Normalization
2Sided Lighting 1
2Sided Lighting 2
Texturing
Result
Inventive Shading
Conclusion

Vertex shader

```
1  varying float albedo, specular;
2
3  void main()
4  {
5      vec3 pos_EC = vec3(gl_ModelViewMatrix * gl_Vertex);
6      vec3 normal_EC = normalize(gl_NormalMatrix * gl_Normal);
7      vec3 light_EC = normalize(gl_LightSource[0].position.xyz -
           pos_EC);
8      vec3 half_EC = normalize(light_EC - normalize(pos_EC));
9
10     albedo = max(dot(normal_EC, light_EC), 0.0);
11     specular = pow(max(dot(normal_EC, half_EC),0.0),
           gl_FrontMaterial.shininess);
12
13     gl_Position = ftransform();
14 }
```

Fragment shader

```
1  varying float albedo, specular;
2
3  void main()
4  {
5      vec4 ambientp =   gl_LightSource[0].ambient  *
           gl_FrontMaterial.ambient;
6      vec4 diffusep =   gl_LightSource[0].diffuse  *
           gl_FrontMaterial.diffuse;
7      vec4 specularp =  gl_LightSource[0].specular *
           gl_FrontMaterial.specular;
8
9      gl_FragColor = ambientp + albedo * diffusep + specular *
           specularp;
10 }
```

Result 1

Basic Shading
A Minimal Shader
Result
Diffuse Lighting
Result
PV Specular 1
PV Specular 2
Result 1
Result 2
PP Specular 1
PP Specular 2
Result
Normalization
2Sided Lighting 1
2Sided Lighting 2
Texturing
Result
Inventive Shading
Conclusion

Result 2

Basic Shading
A Minimal Shader
Result
Diffuse Lighting
Result
PV Specular 1
PV Specular 2
Result 1
Result 2
PP Specular 1
PP Specular 2
Result
Normalization
2Sided Lighting 1
2Sided Lighting 2
Texturing
Result
Inventive Shading
Conclusion

Vertex shader

```
1  varying vec3 normal_EC, light_EC, pos_EC;
2
3  void main()
4  {
5      pos_EC = vec3(gl_ModelViewMatrix * gl_Vertex);
6      normal_EC = normalize(gl_NormalMatrix * gl_Normal);
7      light_EC = gl_LightSource[0].position.xyz - pos_EC;
8
9      gl_Position = ftransform();
10 }
```

Fragment shader

```
1  varying vec3 normal_EC, light_EC, pos_EC;
2
3  void main()
4  {
5      vec4 ambientp =   gl_LightSource[0].ambient  *
           gl_FrontMaterial.ambient;
6      vec4 diffusep =   gl_LightSource[0].diffuse  *
           gl_FrontMaterial.diffuse;
7      vec4 specularp =  gl_LightSource[0].specular *
           gl_FrontMaterial.specular;
8
9      vec3 n_EC = normalize(normal_EC);
10     vec3 l_EC = normalize(light_EC);
11     vec3 h_EC = normalize(l_EC - normalize(pos_EC));
12
13     float albedo = max(dot(n_EC, l_EC), 0.0);
14     float specular = pow(max(dot(n_EC, h_EC),0.0),
           gl_FrontMaterial.shininess);
15
16     gl_FragColor = ambientp + albedo * diffusep + specular *
           specularp;
17 }
```

Result

Basic Shading
A Minimal Shader
Result
Diffuse Lighting
Result
PV Specular 1
PV Specular 2
Result 1
Result 2
PP Specular 1
PP Specular 2
Result
Normalization
2Sided Lighting 1
2Sided Lighting 2
Texturing
Result
Inventive Shading
Conclusion

Regarding Vector Normalization

Basic Shading
A Minimal Shader
Result
Diffuse Lighting
Result
PV Specular 1
PV Specular 2
Result 1
Result 2
PP Specular 1
PP Specular 2
Result
Normalization
2Sided Lighting 1
2Sided Lighting 2
Texturing
Result
Inventive Shading
Conclusion

- normalize vectors after interpolation
- swizzle away extraneous components when normalizing
- `gl_Normal` is a vec3

Two-Sided Lighting: Method 1

Basic Shading
A Minimal Shader
Result
Diffuse Lighting
Result
PV Specular 1
PV Specular 2
Result 1
Result 2
PP Specular 1
PP Specular 2
Result
Normalization
2Sided Lighting 1
2Sided Lighting 2
Texturing
Result
Inventive Shading
Conclusion

- disable backface culling
- in FS:

```
1  if (!gl_FrontFacing) n = -n;
```

Two-Sided Lighting: Method 2

Basic Shading
A Minimal Shader
Result
Diffuse Lighting
Result
PV Specular 1
PV Specular 2
Result 1
Result 2
PP Specular 1
PP Specular 2
Result
Normalization
2Sided Lighting 1
2Sided Lighting 2
Texturing
Result
Inventive Shading
Conclusion

- disable backface culling, enable `GL_VERTEX_PROGRAM_TWO_SIDE`
- in VS, write to `gl_FrontColor` and `gl_BackColor`
- in FS, read from `gl_Color`

Note: `gl_Color` is an attribute in VS and a varying in FS. This method is for per-vertex lighting only.

Texturing

Basic Shading
A Minimal Shader
Result
Diffuse Lighting
Result
PV Specular 1
PV Specular 2
Result 1
Result 2
PP Specular 1
PP Specular 2
Result
Normalization
2Sided Lighting 1
2Sided Lighting 2
Texturing
Result
Inventive Shading
Conclusion

In VS add:

```
1  gl_TexCoord[0] = gl_MultiTexCoord0;
```

In FS add:

```
1  uniform sampler2D tunit;
```

```
1  vec4 t = texture2D(tunit, gl_TexCoord[0].st);
```

Result

Basic Shading
A Minimal Shader
Result
Diffuse Lighting
Result
PV Specular 1
PV Specular 2
Result 1
Result 2
PP Specular 1
PP Specular 2
Result
Normalization
2Sided Lighting 1
2Sided Lighting 2
Texturing
Result
Inventive Shading
Conclusion

Basic Shading
Inventive Shading
Wireframe Shading
Main Idea
Project
Compute Distance
Fragment Shader
Result
Volume Shading
Vertex Shader
Fragment Shader
Shader Code
3D Texture Example
Noise Example
Sphere Close-up
Cube Close-up
Conclusion

Inventive Shading

Wireframe Shading

Basic Shading
Inventive Shading
Wireframe Shading
Main Idea
Project
Compute Distance
Fragment Shader
Result
Volume Shading
Vertex Shader
Fragment Shader
Shader Code
3D Texture Example
Noise Example
Sphere Close-up
Cube Close-up
Conclusion

OpenGL Red Book describes two methods of rendering a wireframe view with *the hidden surfaces removed*.

A much better method is described in:

- A. Bærentzen, S. L. Nielsen, M. Gjøl, B. D. Larsen, and N. J. Christensen. Single-Pass Wireframe Rendering. In *SIGGRAPH*, page 149, 2006

Main Idea

Basic Shading
Inventive Shading
Wireframe Shading
Main Idea
Project
Compute Distance
Fragment Shader
Result
Volume Shading
Vertex Shader
Fragment Shader
Shader Code
3D Texture Example
Noise Example
Sphere Close-up
Cube Close-up
Conclusion

- provide triangle vertices to the VS
  - In OpenGL 2.0 this can be done by using `gl_MultiTexCoordN.xyz`.
- project the vertices into screen space
- calculate distances to the triangle edges
- interpolate them to determine location of the border in FS

Project

```
1  uniform vec4 viewport;
```

...

```
1  vec4 vp0 = gl_ModelViewProjectionMatrix*vec4(
       gl_MultiTexCoord0.xyz, 1.0);
2  vp0.x /= vp0.w;
3  vp0.y /= vp0.w;
4  vp0.z /= vp0.w;
5  vp0.x = viewport.x + 0.5*viewport.z*(vp0.x + 1.0);
6  vp0.y = viewport.y + 0.5*viewport.w*(vp0.y + 1.0);
7  //vp0.z = gl_DepthRange.near + 0.5*gl_DepthRange.far*(vp0
       .z + gl_DepthRange.diff);
```

code credit T. Hinks

Compute Distance

```
1  float dx, dy;
2  dx = vp1.x - vp0.x;
3  dy = vp1.y - vp0.y;
4  gl_TexCoord[0].x = abs(dx*(vp0.y - vp.y) - (vp0.x - vp.x)
       *dy)/sqrt(dx*dx + dy*dy);
5  gl_TexCoord[0].x *= vp.w; // Perspective pre-
       multiplication.
6  // Varying post-multiplication for fragment shader.
7  gl_TexCoord[0].w = 1.0/vp.w;
```

code credit T. Hinks

Fragment Shader

Basic Shading
Inventive Shading
Wireframe Shading
Main Idea
Project
Compute Distance
Fragment Shader
Result
Volume Shading
Vertex Shader
Fragment Shader
Shader Code
3D Texture Example
Noise Example
Sphere Close-up
Cube Close-up
Conclusion

- find the minimum distance
- compare to line width
- fill in color appropriately

To antialias the lines fade out line color using e^{-kx^2}.
Can use functions exp or exp2.

Result

Basic Shading

Inventive Shading

Wireframe Shading

Main Idea

Project

Compute Distance

Fragment Shader

Result

Volume Shading

Vertex Shader

Fragment Shader

Shader Code

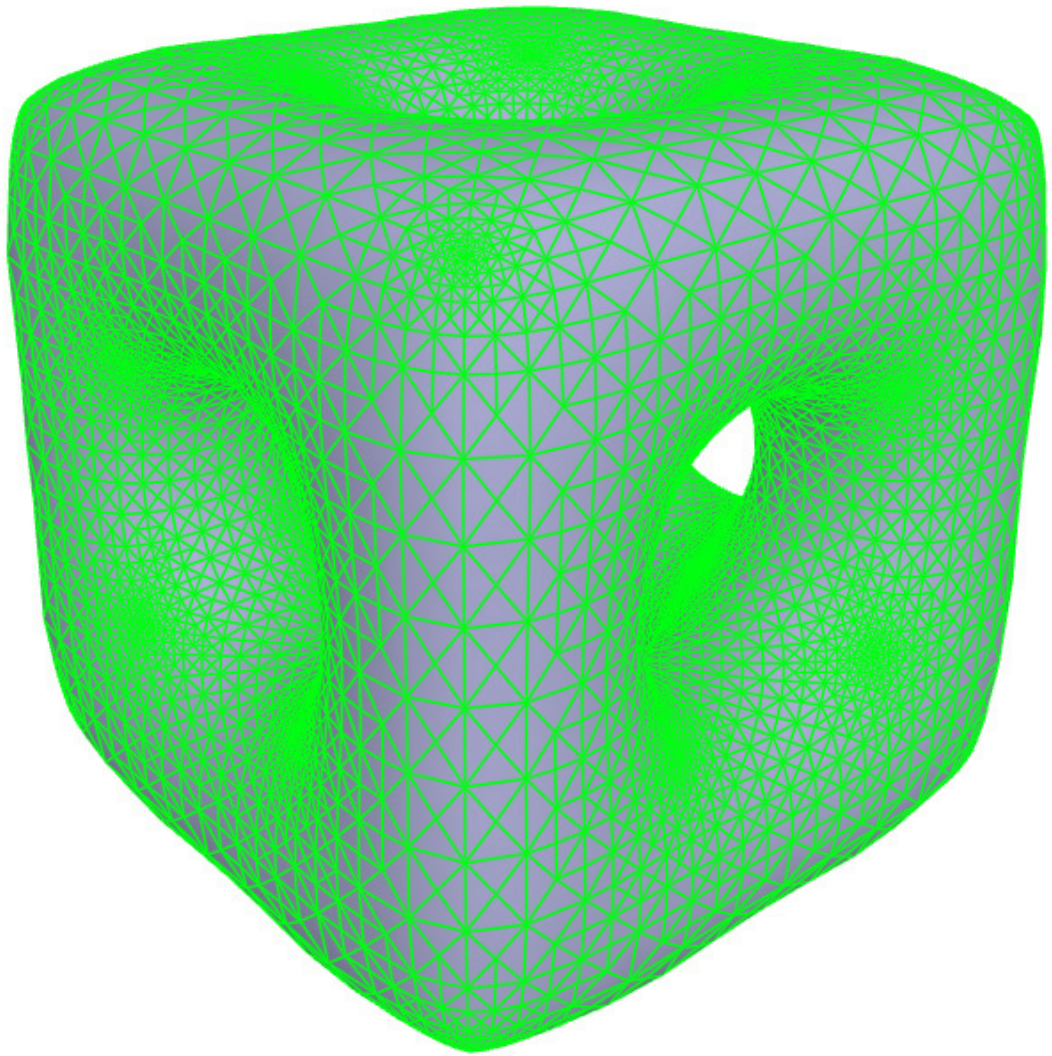
3D Texture Example

Noise Example

Sphere Close-up

Cube Close-up

Conclusion



Volume Shading

Basic Shading
Inventive Shading
Wireframe Shading
Main Idea
Project
Compute Distance
Fragment Shader
Result
Volume Shading
Vertex Shader
Fragment Shader
Shader Code
3D Texture Example
Noise Example
Sphere Close-up
Cube Close-up
Conclusion

A simple method of volume visualization involves tracing voxels in a shader.

Described in:

- W. F. Engel, editor. *ShaderX2*, chapter Voxel Rendering with PS 3.0. pages 161–171.
- maybe also in *Graphics Shaders* by M. Bailey and S. Cunningham

Vertex Shader

Get starting location from (3D) texture coordinates.

Get ray direction from:

```
1  vec4 dir_WCS = gl_ModelViewMatrix*gl_Vertex;
2  dir_WCS.w = 0.0;
3  vec3 DIR = normalize(vec3(gl_ModelViewMatrixInverse *
       dir_WCS));
```

or:

```
1  vec3 DIR = normalize(vec3(gl_Vertex -
       gl_ModelViewMatrixInverse * vec4(0.0, 0.0, 0.0, 1.0)))
       ;
```

Fragment Shader

Basic Shading
Inventive Shading
Wireframe Shading
Main Idea
Project
Compute Distance
Fragment Shader
Result
Volume Shading
Vertex Shader
Fragment Shader
Shader Code
3D Texture Example
Noise Example
Sphere Close-up
Cube Close-up
Conclusion

Composite voxel samples along the ray using α and $1 - \alpha$.
Back to front equation:

$$result_i = \alpha_i \cdot c_i + (1 - \alpha_i) \cdot result_{i-1}$$

Front to back equation (indexed in opp. order):

$$result_N = \sum_{i=0}^{N} \left(\Pi_{k=0}^{i-1}(1 - \alpha_k)\right) \cdot \alpha_i \cdot c_i$$

Note: similar for α of the result.

```
1  float alpha_acc = 1.0;                    // Π_i(1 − α_i)
2  vec3 result = vec3(0.0, 0.0, 0.0); // resulting color
3  float result_alpha = 0.0;
4
5  for(int i=0; i < steps; ++i)
6  {
7      // bounds check: must be inside the texture cube
8      if (all(greaterThan(cur_pos, zero_m_eps)) &&
9          all(lessThan(cur_pos, one_p_eps)))
10       {
11         float alpha_i = ??;
12         vec3 color_i = ??;
13
14         // calculate the contribution:
15         result += alpha_acc * alpha_i * color_i;
16         result_alpha += alpha_acc * alpha_i;
17
18         alpha_acc *= (1.0 - alpha_i);
19       }
20     cur_pos += dir_step;
21 }
22 gl_FragColor = vec4(result + alpha_acc * bg_color,
       result_alpha);
```

3D Texture Example

Basic Shading

Inventive Shading

Wireframe Shading

Main Idea

Project

Compute Distance

Fragment Shader

Result

Volume Shading

Vertex Shader

Fragment Shader

Shader Code

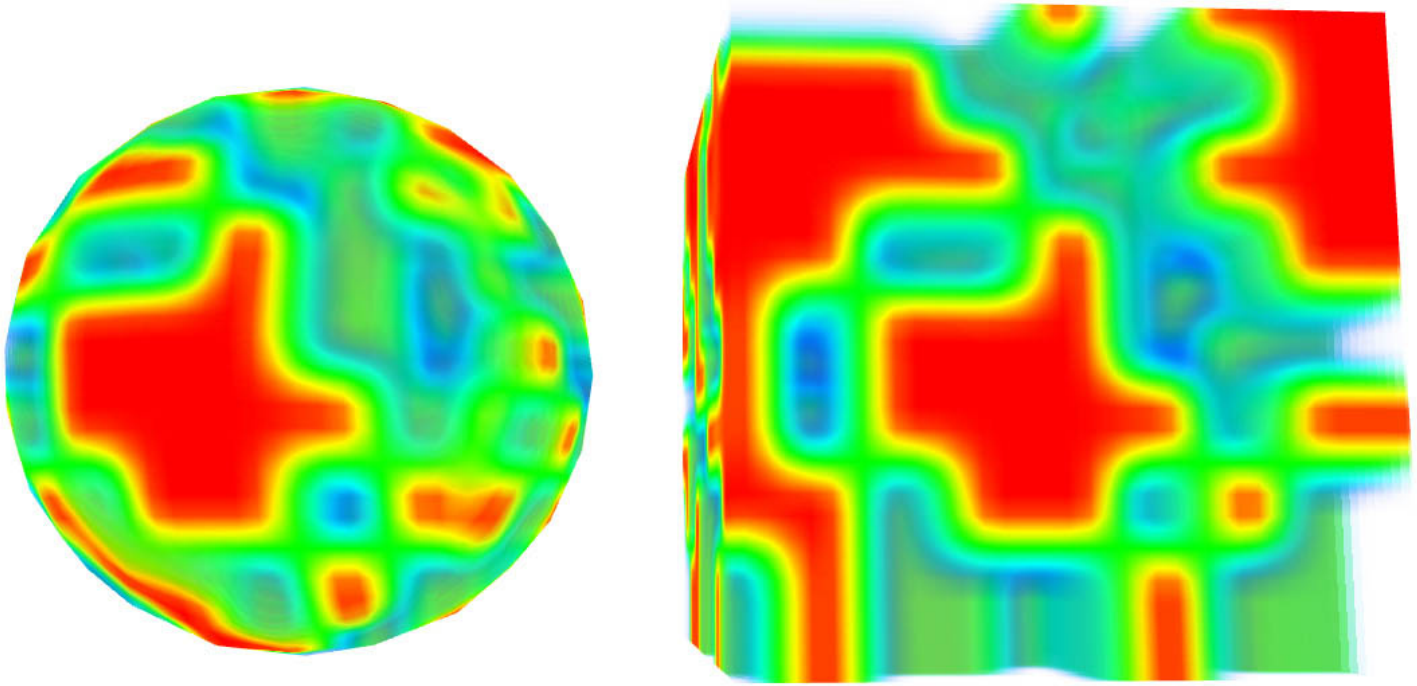
3D Texture Example

Noise Example

Sphere Close-up

Cube Close-up

Conclusion



Noise Example

Basic Shading

Inventive Shading

Wireframe Shading

Main Idea

Project

Compute Distance

Fragment Shader

Result

Volume Shading

Vertex Shader

Fragment Shader

Shader Code

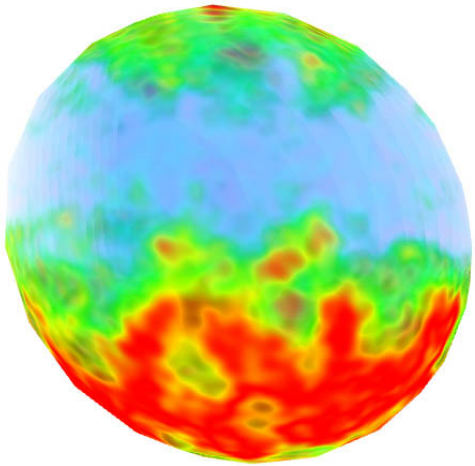
3D Texture Example

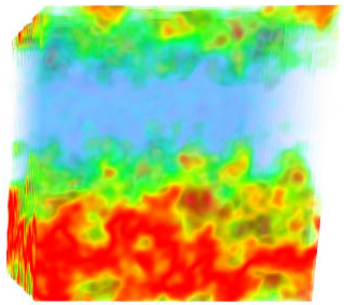
Noise Example

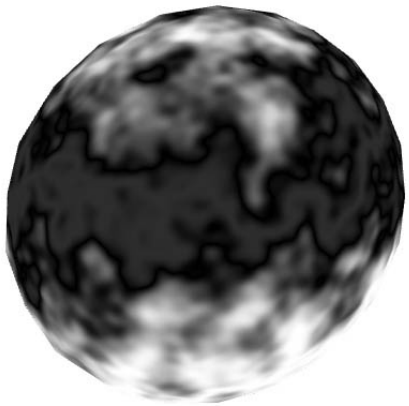
Sphere Close-up

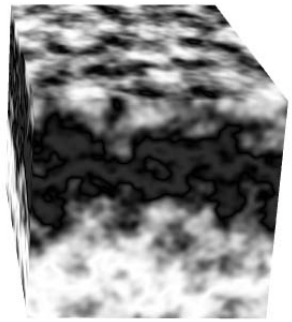
Cube Close-up

Conclusion









Noise Example

Basic Shading

Inventive Shading

Wireframe Shading

Main Idea

Project

Compute Distance

Fragment Shader

Result

Volume Shading

Vertex Shader

Fragment Shader

Shader Code

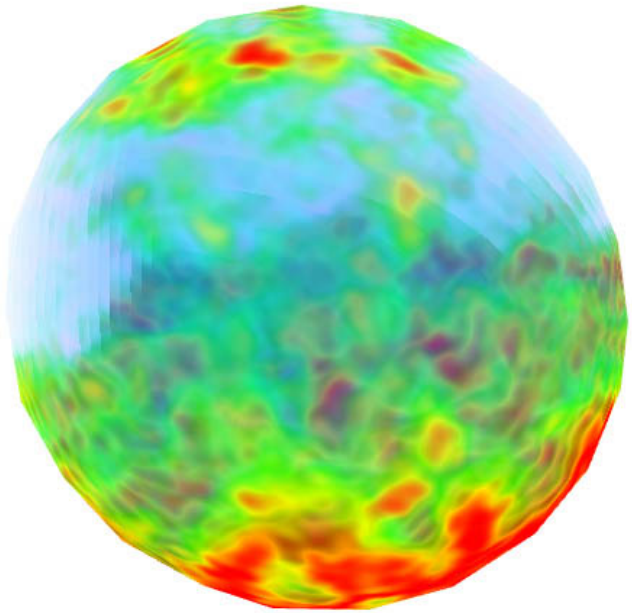
3D Texture Example

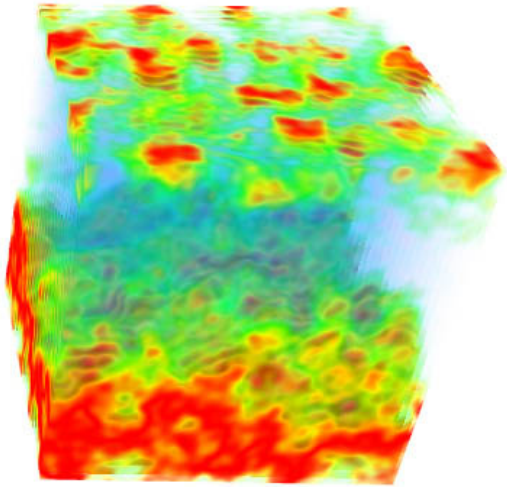
Noise Example

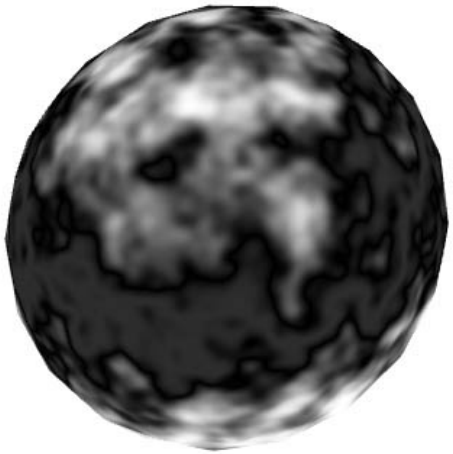
Sphere Close-up

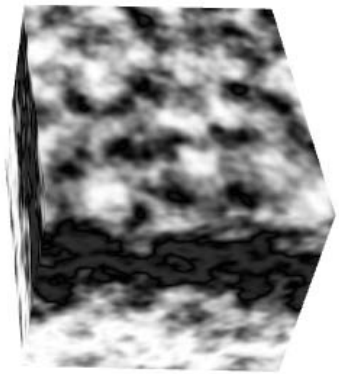
Cube Close-up

Conclusion









Sphere Close-up

Basic Shading

Inventive Shading

Wireframe Shading

Main Idea

Project

Compute Distance

Fragment Shader

Result

Volume Shading

Vertex Shader

Fragment Shader

Shader Code

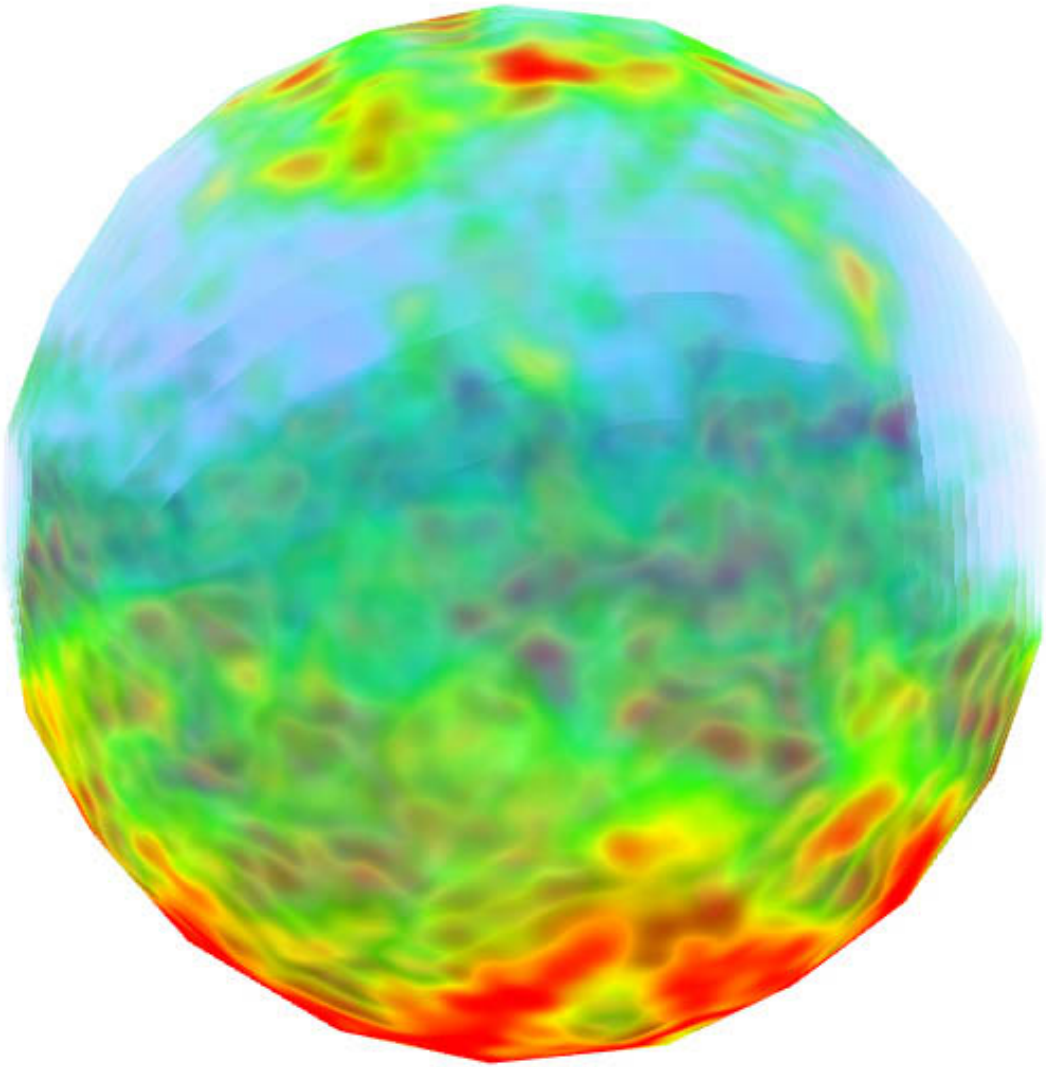
3D Texture Example

Noise Example

Sphere Close-up

Cube Close-up

Conclusion



Cube Close-up

Basic Shading

Inventive Shading

Wireframe Shading

Main Idea

Project

Compute Distance

Fragment Shader

Result

Volume Shading

Vertex Shader

Fragment Shader

Shader Code

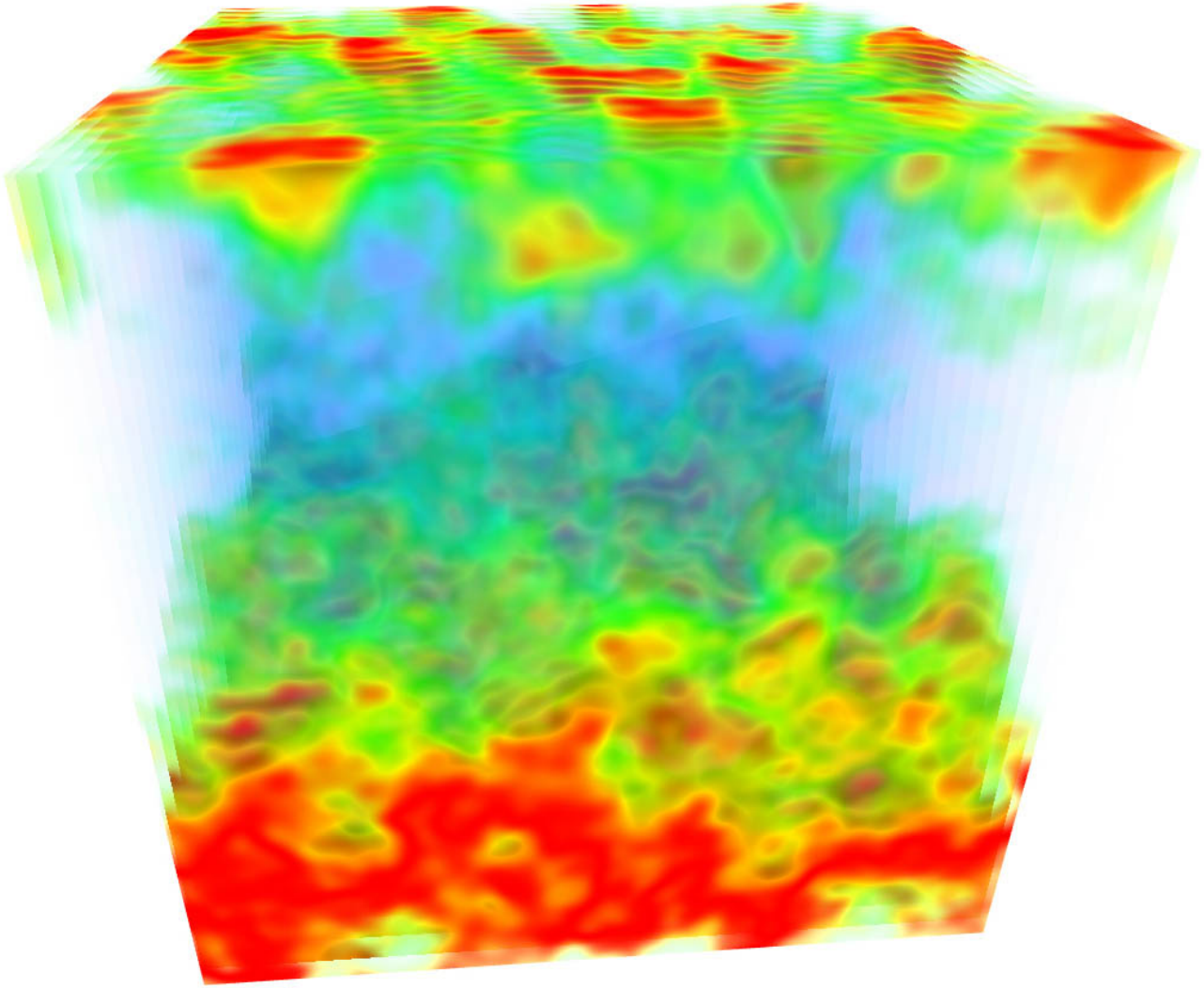
3D Texture Example

Noise Example

Sphere Close-up

Cube Close-up

Conclusion



Version Note

Basic Shading
Inventive Shading
Conclusion
Version Note

OpenGL Version: 2.0.1
Shading language version: 1.10 NVIDIA via Cg 1.3 compiler