

KAIST

CS 380

Introduction to Computer Graphics

LAB (9)

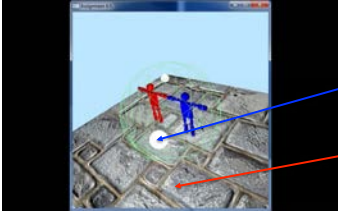
2015.05.06

VISUAL
COMPUTING Lab

KAIST

Overview

- Material Infrastructure
 - Multiple shaders per one frame
- Bump mapping
 - Normal map



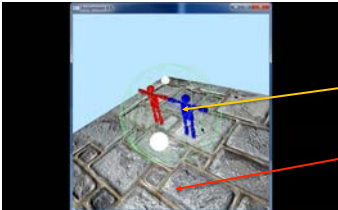
2

VISUAL
COMPUTING Lab

KAIST

Multiple Shaders

- Each shader has own **uniform** variables
- Different GLSL shader do not know about the values of each other's uniform variables



3

VISUAL
COMPUTING Lab

KAIST

Transferring Uniform Value

Uniform.h

- Uniform: dictionary mapping from name to value

*Uniforms.put(the name of the variable in the shader, **the actual value**)*

Types: float, int, matrix4, shared_ptr <Texture>, ...

Drawer
Picker
SgShapeNode

E.g.)
 // Suppose uniforms is of type Uniforms, and m is of type Matrix4
 uniforms.put("uProjection", m);
 // Suppose light is of type Cvec3
 uniforms.put("uLight", light);
 // Set uColor variable to red
 uniforms.put("uColor", Cvec3(1, 0, 0));
 // You can even chain the put, since put returns the object itself
 uniforms.put("a", 1)
 .put("b", 10)
 .put("c", Cvec2(1, 2));

4

VISUAL
COMPUTING Lab

RenderStates

KAIST

- *RenderStates*: A subset of OpenGL state
- State does not immediately take effect in OpenGL
- The state will be applied when you call the member function: ***apply()***
- Useful for multi-shader case

E.g.)
RenderStates r1;
r1.enable(GL_BLEND);
r1.apply();

5



Geometry & Texture

KAIST

- Complex types of geometry and texture
- Geometry
 - GeometryPN: position and normal
 - GeometryPNTBX: position, normal, tangent, binormal, and texture coordinate
- Texture
 - ImageTexture

6



Material

KAIST

- **Material** contains three parts
 - Shared pointer
 - GLSL shader program used
 - Uniforms
 - accessible through *getUniforms()*
 - RenderStates
 - accessible through *getRenderStates ()*
- Member function
 - *.draw(geometry, extraUniforms)*

E.g.)
sendModelViewNormalMatrix(uniforms, MVM, normalMatrix(MVM));
g_arcballMat->draw("g_sphere, uniforms);

7



Scene Graph & Material

KAIST

- Each *SgGeometryShapeNode* has own "Material"
- The robots: diffuse color
- The arcball: wireframe and solid color
- The ground: texture

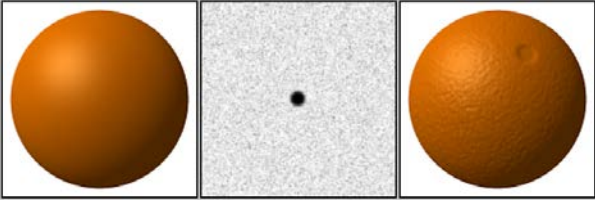
8




Bump Mapping

KAIST

- Simulating the bumps on the surface




- Instead of changing the geometry itself,
- Modify the surface normal to simulate bumps

9 

Bump Mapping

KAIST



Normal map
 $\mathbf{n} = [n_r, n_g, n_b, 0]^t$

- Normal map defined w.r.t. the tangent frame
- Object frame: ${}^i b' = {}^i e' M$
- Tangent frame: $T(1:3, 1:3) = [\text{tangent}, \text{binormal}, \text{normal}] \quad {}^i e' = {}^i b' T$
- New normal: $M^{-1} T \mathbf{n}$

10 