Advanced Shaders

Advanced Rendering

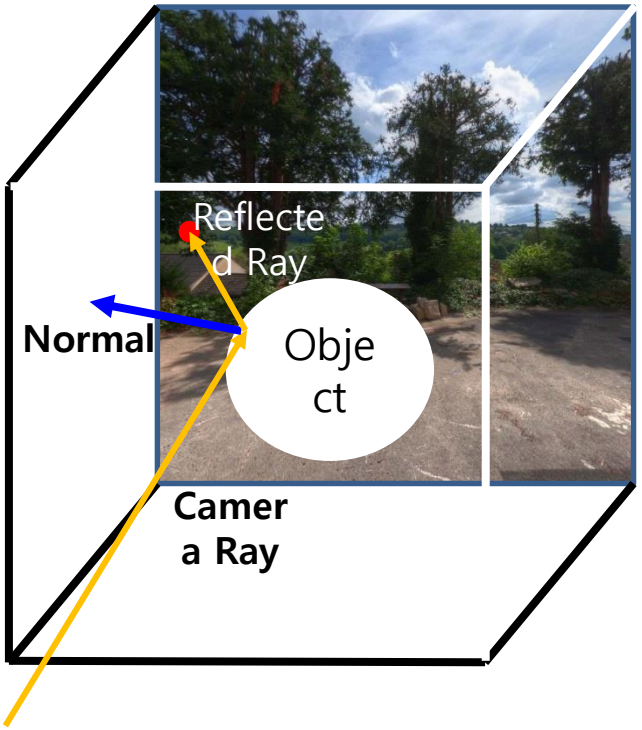
- **Environment Mapping**
  - Reflective environment mapping
  - Refractive environment mapping
- **Normal Mapping**

Reflective Environment mapping

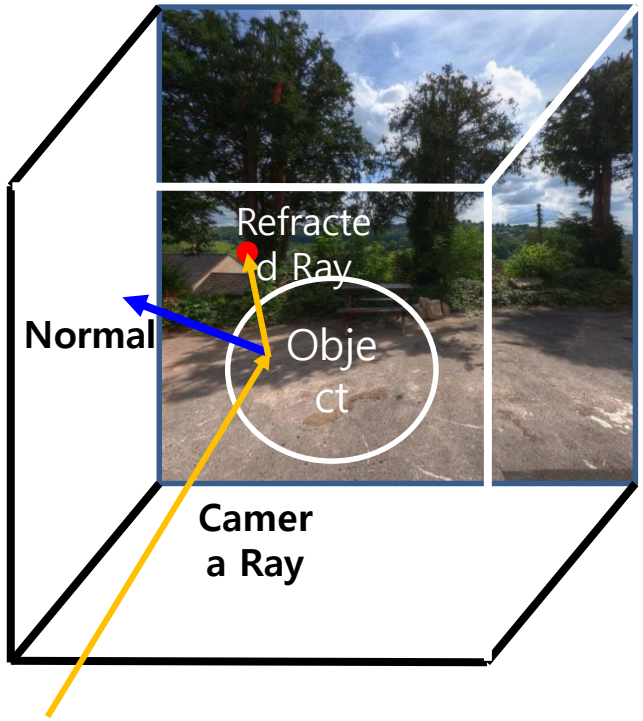
Refractive Environment mapping

Normal mapping

Environment mapping



<Reflective mapping>

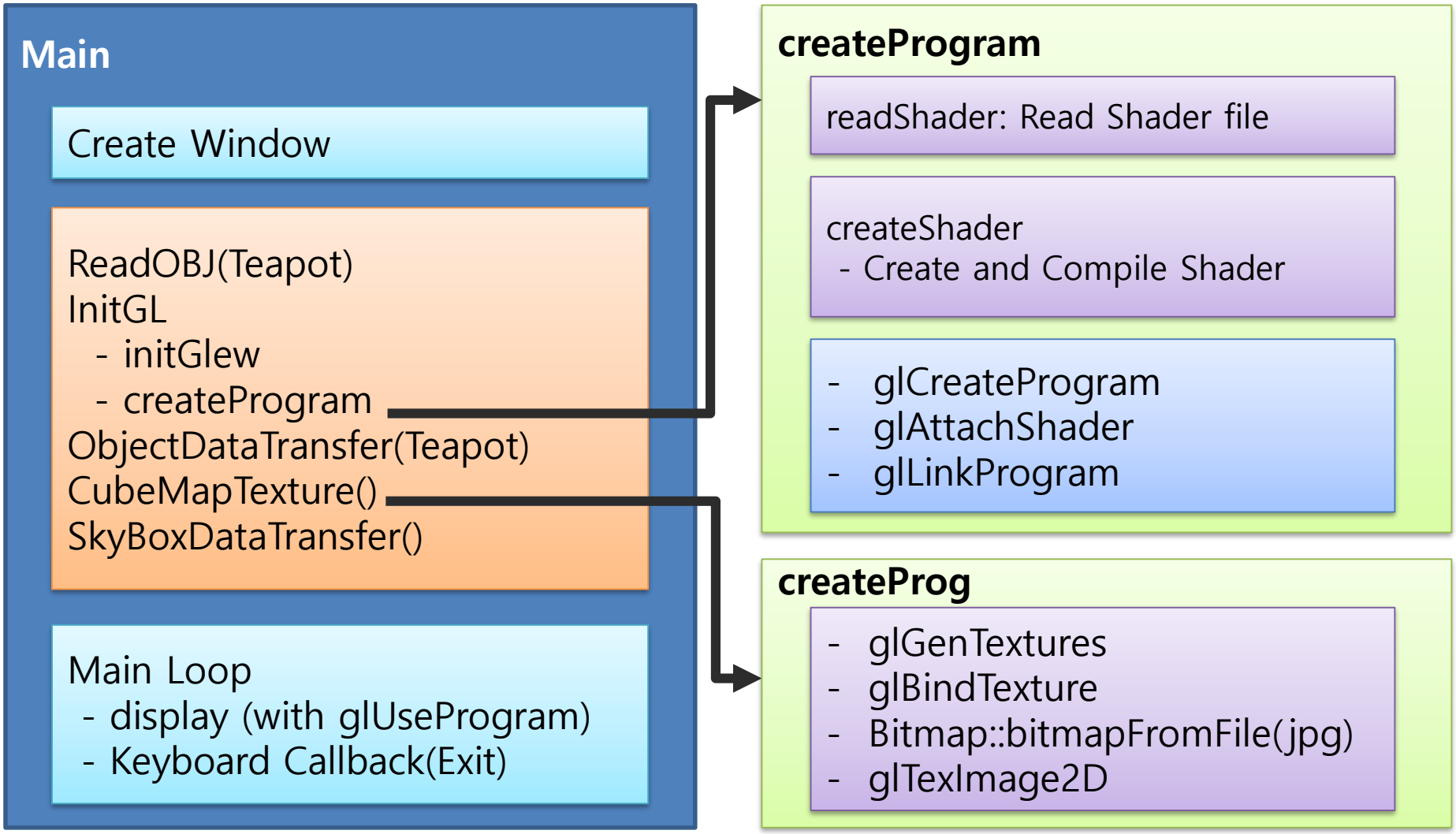


<Refractive mapping>

Environment Coding Exercise

- **Copy Sample Skeleton Code**
  - vglconnect ID@163.152.20.246
  - cp –r /home/share/Environment ./
  - cd Environment
- **Notepad: Shader code 수정**
- **Compile program**
  - make
  - vglrun ./EXE

Program Flow



Program structure

```
#include "XWindow.h"
#include <stdio.h>
#include <stdlib.h>
#include <string>
//function declaration//
//Global variables//

int main(int argc, char *argv[]) {
    //Window Initialization//
    ReadOBJ("Teapot.obj");
    initGL();
    ObjectDataTransfer(); //Transferring Teapot.obj file to GPU
    CubeMapTexture(); //Binding Texture
    SkyboxDataTransfer(); // DataTranfer for SkyBox

    while(1) {
        Display();
        KeyboardCallback();
        }
    }
}
```

initGL() @Main

```
void initGL(){
        glewInit();
        createProgram();
        glEnable(GL_DEPTH_TEST);
}
void createProgram(){
          //Shader for Teapot drawing
        char *vertexShaderSource = ReadFile("Object.vs");
        char *fragmentShaderSource = ReadFile("Object.fs");

        ObjVertShader = createShader(vertexShaderSource, GL_VERTEX_SHADER);
        ObjFragShader = createShader(fragmentShaderSource, GL_FRAGMENT_SHADER);
        ObjShaderProgram = glCreateProgram();
        glAttachShader(ObjShaderProgram, ObjVertShader);
        glAttachShader(ObjShaderProgram, ObjFragShader);
        glLinkProgram(ObjShaderProgram);
```

initGL() –Cont.

```
    //Shader for Skybox
    vertexShaderSource = ReadFile("Skybox.vs");
    fragmentShaderSource = ReadFile("Skybox.fs");

    SkyVertShader = createShader(vertexShaderSource, GL_VERTEX_SHADER);
    SkyFragShader = createShader(fragmentShaderSource, GL_FRAGMENT_SHADER);
    SkyShaderProgram = glCreateProgram();
    glAttachShader(SkyShaderProgram, SkyVertShader);
    glAttachShader(SkyShaderProgram, SkyFragShader);
    glLinkProgram(SkyShaderProgram);
}
```

ObjectDataTransfer Code@Main

```
void ObjectDataTransfer(){
        //Generating Buffer Objects
        glGenVertexArrays(1, VAO);
        glGenBuffers(2, VBO);
        glGenBuffers(1, EBO);
        //Transferring Vertex data to Device
        glBindVertexArray(VAO[0]);
        glBindBuffer(GL_ARRAY_BUFFER, VBO[0]);
        glBufferData(GL_ARRAY_BUFFER, 4 * sizeofVert, vertices, GL_STATIC_DRAW);
        glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(float), (void*)0);
        glEnableVertexAttribArray(0);
         //Transferring Normal(Color) data to Device
        glBindBuffer(GL_ARRAY_BUFFER, VBO[1]);
        glBufferData(GL_ARRAY_BUFFER, 4 * sizeofNorm, normals, GL_STATIC_DRAW);
        glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(float), (void*)0);
        glEnableVertexAttribArray(1);
        //Transferring Index data to Device
        glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EBO[0]);
        glBufferData(GL_ELEMENT_ARRAY_BUFFER, 4 * sizeofInd, indices, GL_STATIC_DRAW);
        glBindBuffer(GL_ARRAY_BUFFER, 0);
        glBindVertexArray(0);
}
```

CubeMapTexture Code@Main

```
void CreateCubeMap(){
        glActiveTexture(GL_TEXTURE0); //Activating Texture 0
        glGenTextures(1, &cubeMapId); //Generating Texture
        glBindTexture(GL_TEXTURE_CUBE_MAP, cubeMapId); //Binding Texture
        //Add texture to Back side
        Bitmap bmp = Bitmap::bitmapFromFile("textures/cubemap1.jpg");
        glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X + 0, 0, GL_RGB, bmp.width(),
        bmp.height(), 0, GL_RGB, GL_UNSIGNED_BYTE, bmp.pixelBuffer());
        bmp = Bitmap::bitmapFromFile("textures/cubemap2.jpg");
         //Add texture to Front side
        glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X + 1, 0, GL_RGB, bmp.width(),
        bmp.height(), 0, GL_RGB, GL_UNSIGNED_BYTE, bmp.pixelBuffer());
        bmp = Bitmap::bitmapFromFile("textures/cubemap3.jpg");
         //Add texture to Left side
        glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X + 2, 0, GL_RGB, bmp.width(),
        bmp.height(), 0, GL_RGB, GL_UNSIGNED_BYTE, bmp.pixelBuffer());
        bmp = Bitmap::bitmapFromFile("textures/cubemap4.jpg");
         //Add texture to Right side
        glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X + 3, 0, GL_RGB, bmp.width(),
        bmp.height(), 0, GL_RGB, GL_UNSIGNED_BYTE, bmp.pixelBuffer());
        bmp = Bitmap::bitmapFromFile("textures/cubemap5.jpg");
```

CubeMapTexture –Cont.

```
        //Add texture to Top side
        glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X + 4, 0, GL_RGB, bmp.width(),
        bmp.height(), 0, GL_RGB, GL_UNSIGNED_BYTE, bmp.pixelBuffer());
        bmp = Bitmap::bitmapFromFile("textures/cubemap6.jpg");
        //Add texture to Bottom side
        glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X + 5, 0, GL_RGB, bmp.width(),
        bmp.height(), 0, GL_RGB, GL_UNSIGNED_BYTE, bmp.pixelBuffer());

        glTexParameterf(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
        glTexParameterf(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
        glTexParameterf(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
        glTexParameterf(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
        glTexParameterf(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_R, GL_CLAMP_TO_EDGE);
        glBindTexture(GL_TEXTURE_CUBE_MAP, 0);
}
```

SkyboxDataTransfer Code@Main

```
void SkyboxDataTransfer(){
        float skyboxVertices[] ={ ... };
        glGenVertexArrays(1, &skyVAO);
        glGenBuffers(1, &skyVBO);
        glBindVertexArray(skyVAO);
        glBindBuffer(GL_ARRAY_BUFFER, skyVBO);
        glBufferData(GL_ARRAY_BUFFER, sizeof(skyboxVertices),
        &skyboxVertices, GL_STATIC_DRAW);
        glEnableVertexAttribArray(0);
        glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 *
        sizeof(float), (void*)0);
}
```

```
{ // positions
    -1.0f,  1.0f, -1.0f,-1.0f, -1.0f, -1.0f,
     1.0f, -1.0f, -1.0f, 1.0f, -1.0f, -1.0f,
     1.0f,  1.0f, -1.0f,-1.0f,  1.0f, -1.0f,

    -1.0f, -1.0f,  1.0f,-1.0f, -1.0f, -1.0f,
    -1.0f,  1.0f, -1.0f,-1.0f,  1.0f, -1.0f,
    -1.0f,  1.0f,  1.0f,-1.0f, -1.0f,  1.0f,

    -1.0f, -1.0f, -1.0f, 1.0f, -1.0f,  1.0f,
     1.0f,  1.0f,  1.0f, 1.0f,  1.0f,  1.0f,
     1.0f,  1.0f, -1.0f, 1.0f, -1.0f, -1.0f,

    -1.0f, -1.0f,  1.0f,-1.0f,  1.0f,  1.0f,
     1.0f,  1.0f,  1.0f, 1.0f,  1.0f,  1.0f,
     1.0f, -1.0f,  1.0f,-1.0f, -1.0f,  1.0f,

    -1.0f,  1.0f, -1.0f, 1.0f,  1.0f, -1.0f,
     1.0f,  1.0f,  1.0f, 1.0f,  1.0f,  1.0f,
    -1.0f,  1.0f,  1.0f,-1.0f,  1.0f, -1.0f,

    -1.0f, -1.0f, -1.0f,-1.0f, -1.0f,  1.0f,
     1.0f, -1.0f, -1.0f, 1.0f, -1.0f, -1.0f,
    -1.0f, -1.0f,  1.0f, 1.0f, -1.0f,  1.0f
};
```

Display function@Main

```
void display(){
        glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
        glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
        // Drawing Object
        glUseProgram(ObjShaderProgram);
        float* model = (float*)malloc(16*sizeof(float));
        float* view = (float*)malloc(16*sizeof(float));
        float* projection = (float*)malloc(16*sizeof(float));

        //model Initialization;
        //view Initialization;
        //projectiopn Initialization;

        glUniformMatrix4fv(glGetUniformLocation(ObjShaderProgram, "model"), 1, GL_FALSE, &model[0]);
        glUniformMatrix4fv(glGetUniformLocation(ObjShaderProgram, "view"), 1, GL_FALSE, &view[0]);
        glUniformMatrix4fv(glGetUniformLocation(ObjShaderProgram, "projection"), 1, GL_FALSE, &projection[0]);
        glUniform3f(glGetUniformLocation(ObjShaderProgram, "cameraPos"), 0.0f, 0.0f, 10.0f);

        glBindVertexArray(VAO[0]);
        glActiveTexture(GL_TEXTURE0);
        glBindTexture(GL_TEXTURE_CUBE_MAP, cubeMapId);
        glDrawElements(GL_TRIANGLES, sizeofInd, GL_UNSIGNED_INT, 0);
```

Display function –Cont.

```
        //Drawing Skybox
        view[14] = 0.0;
        glDepthFunc(GL_LEQUAL);
        glUseProgram(SkyShaderProgram);

        glUniformMatrix4fv(glGetUniformLocation(SkyShaderProgram, "view"), 1, GL_FALSE, &view[0]);
        glUniformMatrix4fv(glGetUniformLocation(SkyShaderProgram, "projection"), 1, GL_FALSE, &projection[0]);

        glBindVertexArray(skyVAO);
        glActiveTexture(GL_TEXTURE0);
        glBindTexture(GL_TEXTURE_CUBE_MAP, cubeMapId);
        glDrawArrays(GL_TRIANGLES, 0, 36);
        glBindVertexArray(0);
        glDepthFunc(GL_LESS);

        glXSwapBuffers(dpy, win);
}
```

Shader code: Skybox

```
//Skybox.vs
#version 130
layout (location = 0) in vec3 aPos;
out vec3 TexCoords;
uniform mat4 projection;
uniform mat4 view;
void main(){
    TexCoords = aPos;
    vec4 pos = projection * view * vec4(aPos, 1.0);
    gl_Position = pos.xyww;
}
//Skybox.fs
#version 130
vec4 FragColor;
in vec3 TexCoords;
uniform samplerCube skybox;
void main(){
        FragColor = texture(skybox, TexCoords);
}
```

Shader code: Object Vertex

```
//Object.vs

#version 130
layout(location = 0) in vec3 aPos;
layout(location = 1) in vec3 aNormal;

out vec3 Normal;
out vec3 Position;
uniform mat4 model;
uniform mat4 view;
uniform mat4 projection;

void main(){
    Normal = mat3(transpose(inverse(model))) * aNormal;
    Position = vec3(model * vec4(aPos, 1.0));
    gl_Position = projection * view * model * vec4(aPos, 1.0);
}
```

Shader code: Object Fragment

```
//Object.fs

#version 130

out vec4 FragColor;
in vec3 Normal;
in vec3 Position;

uniform vec3 cameraPos;
uniform samplerCube skybox;

void main(){
    vec3 I = normalize(Position - cameraPos);
    vec3 R = reflect(I, normalize(Normal));
    vec3 reflectRay = reflect(I, normalize(Normal));
    vec3 refractRay = refract(I, normalize(Normal), 0.6f);
    FragColor = vec4(texture(skybox, reflectRay).rgb, 1.0);
}
```

Result of Reflective Environment Mapping

Change the mode to Refraction

```
//Object.fs

#version 130
.....
void main(){
    .....
    vec3 reflectRay = reflect(I, normalize(Normal));
    vec3 refractRay = refract(I, normalize(Normal), 0.6f);
    FragColor = vec4(texture(skybox, reflectRay).rgb, 1.0);
}
```

Change reflectRay to refractRay.

Result of Refractive Environment Mapping