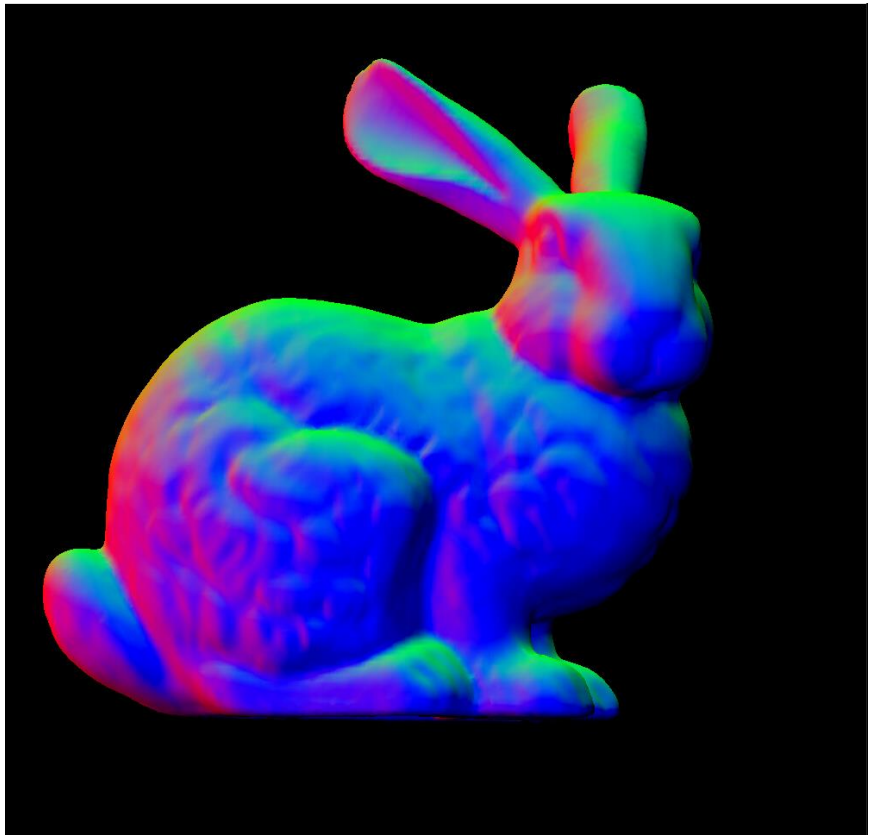
Mesh Render with Buffer Object

Practice : Drawing Bunny 3D Model.

- **Draw Stanford bunny from OBJ File using buffer objects and Shading language.**



Bunny Coding Exercise

- **Copy Sample Skeleton Code**
  - vglconnect ID@163.152.20.246
  - cp –r /home/share/12_OBJLoader ./[Folder name]
  - cd [Folder_name]
- **Notepad: DataTransfer function 작성**
- **Compile program**
  - make
  - vglrun ./EXE

Program Flow

Main

Create Window

ReadOBJ
InitGL
- initGlew
- createProgram
DataTransfer(with buffers)

Main Loop
- display (with glUseProgram)
- Keyboard Callback(Exit)

createProgram

readShader: Read Shader file

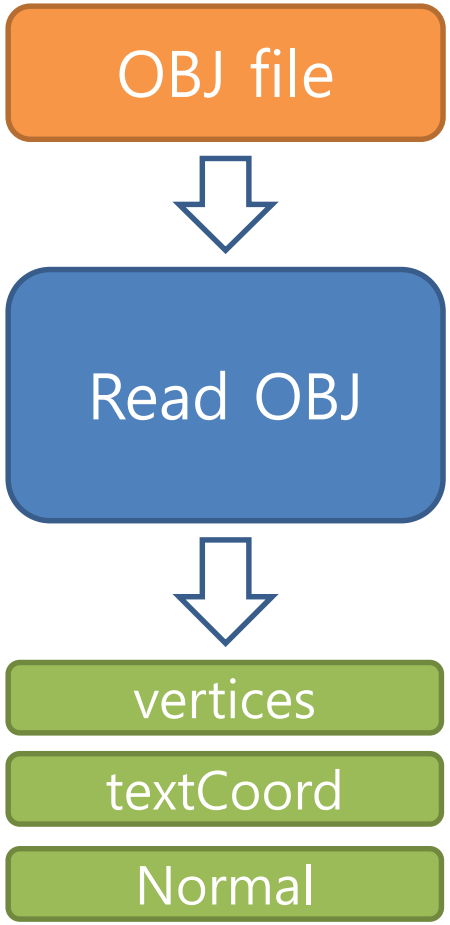
createShader
- Create and Compile Shader

- glCreateProgram
- glAttachShader
- glLinkProgram

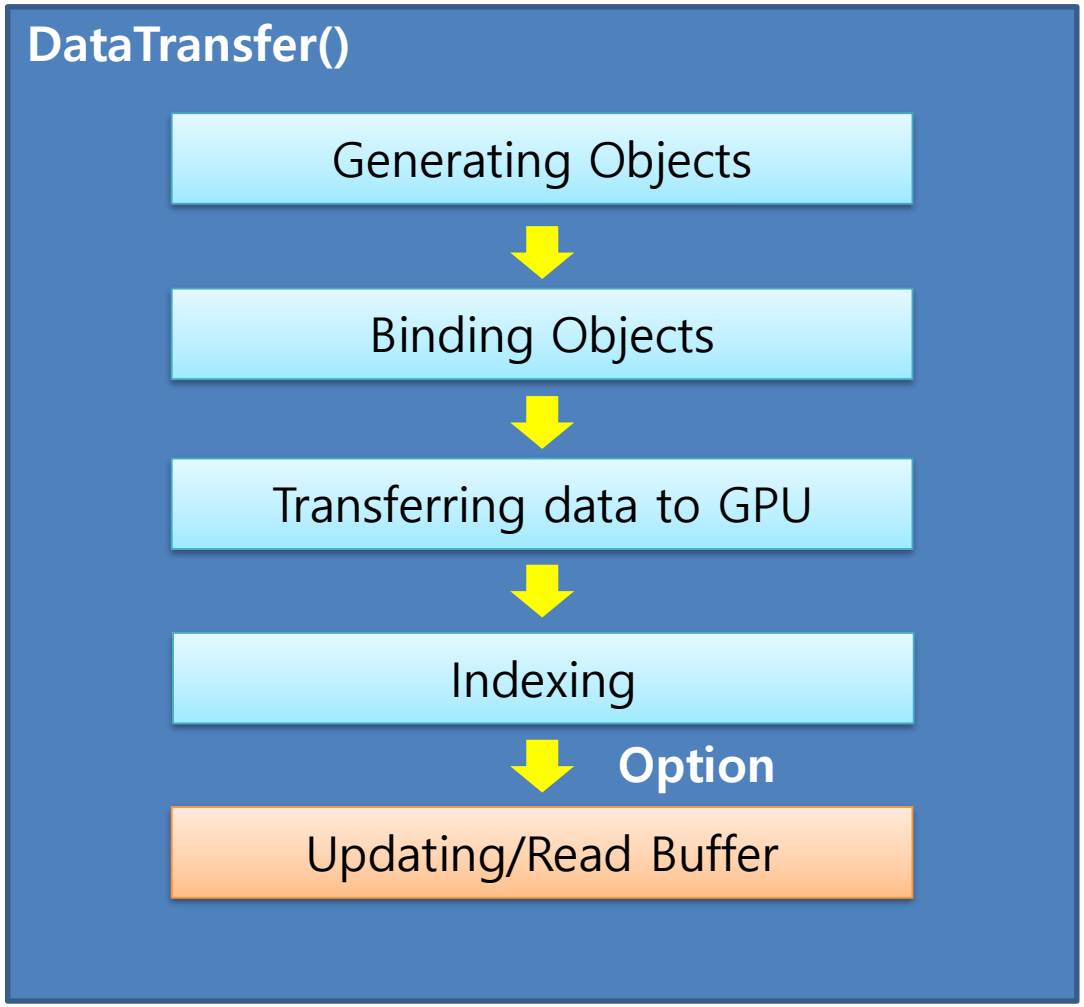
Program structure

```
#include headers
using namespace std;
//function declaration//
//Global variables//
float *vertices;
float *textCoord;
Float *normals;
int main(int argc, char *argv[]) {

    //Window Initialization//
    ReadOBJ("Bunny.obj");
    initGL();
    DataTransfer();
    while(1) {
        Display();
        KeyboardCallback();
        }
    }
}
```



Data Transfer function structure



DataTransfer Code@Main

```
void DataTransfer(){
        //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);
        glBindBuffer(GL_ARRAY_BUFFER, 0);
        glBindVertexArray(0);
}
```

Vertex Attribute pointer

VAO

VBO 1

VBO 2

initGL()

```
void initGL(){
        glewInit();
        createProgram();
}

void createProgram(){
        char *vertexShaderSource = ReadFile("Vertex.glsl");
        char *fragmentShaderSource = ReadFile("Fragment.glsl");

        vertShader = createShader(vertexShaderSource, GL_VERTEX_SHADER);
        fragShader = createShader(fragmentShaderSource, GL_FRAGMENT_SHADER);
        Program = glCreateProgram();
        glAttachShader(Program, vertShader);
        glAttachShader(Program, fragShader);
        glLinkProgram(Program);
}
```

Display function@Main

```
int sizeofVert; // Number of elements in vertices

void display(){
        glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
        glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
        glUseProgram(Program);
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        glTranslatef(-0.3,-0.5,0.0);
        glRotatef(180.0, 0.0, 1.0, 0.0);
        glBindVertexArray(VAO[0]);
        glDrawArrays(GL_TRIANGLES, 0, 3*sizeofInd);
        glXSwapBuffers(dpy, win);
}
```

Shader code

Vertex Attribute pointer

```
//Vertex Shader code
#version 130
layout(location = 0) vec3 aPos; //Alternative to Attribute variable
layout(location = 1) vec3 aColor; //Alternative to Attribute variable
uniform mat4 modelView;
out vec3 ourColor; //Alternative to varying variable
void main()
{
        gl_Position = modelView*vec4(aPos.x, aPos.y, aPos.z, 1.0);
        ourColor = aColor;
}

//Fragment Shader code
#version 130
in vec3 ourColor; //Alternative to varying variable
void main()
{
        gl_FragColor = vec4(ourColor.x, ourColor.y, ourColor.z, 1.0f);
}
```

Chage the OBJ Files

- **Change the filename**

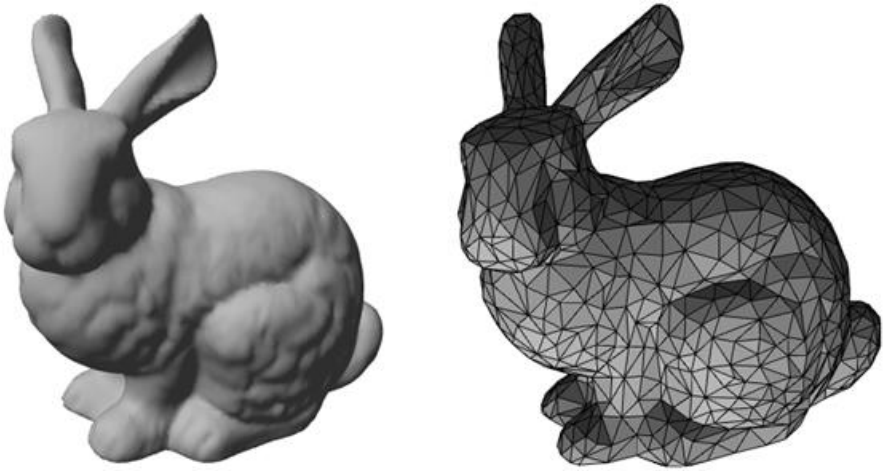
  ReadOBJ(“filename.obj");

- **OBJ File List**
  - Bunny1.obj
  - Bunny2.obj
  - F16.obj
  - sphere.obj

3D Mesh Model: Wavefront File

3D Data File Formats

- **Popular 3D data file formats**
  - Maya binary (.mb)
  - Maya ASCII (.ma)
  - 3DStudio/max file (.3ds or .max)
  - Autodesk binary file (.fbx)
  - Wavefront file (.obj)



cp /home/share/DataTransfer/Bunny.obj ./

Wavefront File

- OBJ (or .OBJ) is a geometry definition file format first developed by Wavefront Technologies for its Advanced Visualizer animation package. The file format is open and has been adopted by other 3D graphics application vendors. For the most part it is a universally accepted format.

- The OBJ file format is a simple data-format that represents 3D geometry alone — namely, the position of each vertex, the UV position of each texture coordinate vertex, vertex normals, and the faces that make each polygon defined as a list of vertices, and texture vertices. Vertices are stored in a counter-clockwise order by default, making explicit declaration of face normals unnecessary. OBJ coordinates have no units, but OBJ files can contain scale information in a human readable comment line.

Wavefront File Format

- **Plane.obj**

```
# This file uses centimeters as units for non-parametric coordinates.

v -100.000000 0.000000 100.000000
v 100.000000 0.000000 100.000000
v -100.000000 0.000000 -100.000000
v 100.000000 0.000000 -100.000000

vt 0.000000 0.000000
vt 1.000000 0.000000
vt 0.000000 1.000000
vt 1.000000 1.000000

vn 0.000000 1.000000 0.000000
vn 0.000000 1.000000 0.000000
vn 0.000000 1.000000 0.000000
vn 0.000000 1.000000 0.000000

f 1/1/1 2/2/2 4/4/3 3/3/4
```

comment

vertices position information

vertices texture coordinate information

vertices normal information

polygonal face element

Wavefront File Format

- **Vertex position**
  - Position is specified in a line starting with the letter "v". That is followed by (x,y,z[,w]) coordinates.
  - w is optional and defaults to 1.0
  - v -111.249123 23.548453 59.670455
- **Texture coordinate**
  - Texture coordinate is specified in a line starting with the letter "vt" followed by (u,v[,w]) coordinates.
  - These usually vary between 0 and 1.
  - w is optional and defaults to 0.0
  - vt 0.500 1
- **Vertex normal**
  - Normal is specified in a line starting with the letter "vn" followed by (x,y,z) coordinates.
  - normals might not be unit vectors.
  - vn 0.707 0.000 0.707

Wavefront File Format

- **Face elements**
  - Faces are defined using lists of vertex, texture and normal indices.
  - Polygons such as quadrilaterals can be defined by using more than three vertex/texture/normal indices.
  - Each index starts from **1** and matches the corresponding each element of a previously defined list.

  - f v1 v2 v3
  - f v1/vt1 v2/vt2 v3/vt3
  - f v1/vt1/vn1 v2/vt2/vn2 v3/vt3/vn3
  - f v1//vn1 v2//vn2 v3//vn3

Drawing Stage

- **Plane.obj**

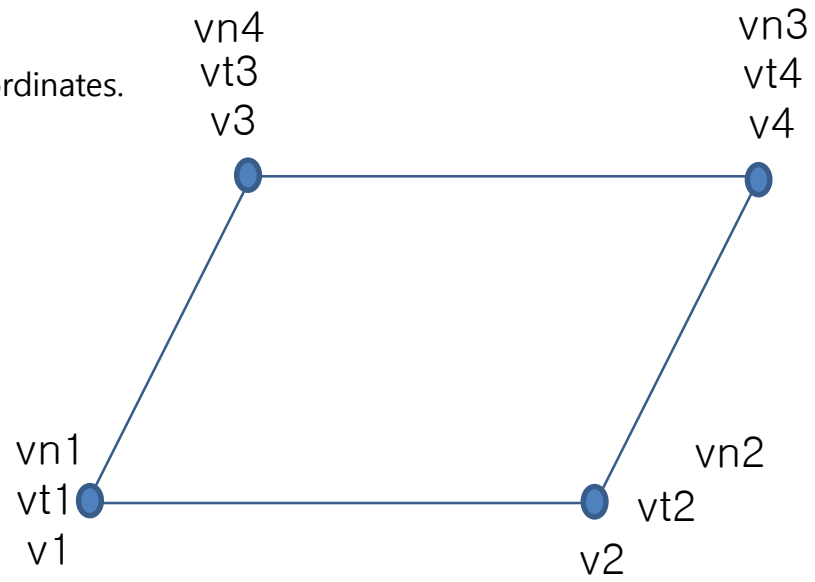
```
# This file uses centimeters as units for non-parametric coordinates.

v -100.000000 0.000000 100.000000
v 100.000000 0.000000 100.000000
v -100.000000 0.000000 -100.000000
v 100.000000 0.000000 -100.000000

vt 0.000000 0.000000
vt 1.000000 0.000000
vt 0.000000 1.000000
vt 1.000000 1.000000

vn 0.000000 1.000000 0.000000
vn 0.000000 1.000000 0.000000
vn 0.000000 1.000000 0.000000
vn 0.000000 1.000000 0.000000

f 1/1/1 2/2/2 4/4/3 3/3/4
```




```
glBegin(GL_QUAD);
glTexCoord(vt1); glNormal(vn1); glVertex(v1);
glTexCoord(vt2); glNormal(vn2); glVertex(v2);
glTexCoord(vt4); glNormal(vn3); glVertex(v4);
glTexCoord(vt3); glNormal(vn4); glVertex(v3);
glEnd();
```

OBJ Loader Example

```
//Global variables
float *vertices;
float *normals;
int *indices;

void ReadOBJ(const char* filename){
    char string[100];
    FILE *handler = fopen(filename, "r");
    int vert = 0;
    int text = 0;
    int norm = 0;
    int face = 0;
    while(fgets(string, 100, handler) != NULL){
        if (strncmp(string, "v ",2) == 0) vert++;
        else if (strncmp(string, "vt", 2) == 0) text++
        else if (strncmp(string, "vn", 2) == 0) norm++;
        else if (strncmp(string, "f ", 2) == 0) face++;

    }
    vertices = (float*)malloc(3* vert *sizeof(float));
    textCoord = (float*)malloc(2*sizeofInd * sizeof(float));
    normals = (float*)malloc(3 * norm * sizeof(float));
```

Counting Size of Data

```
int i = 0;
vert = 0;norm = 0;face = 0;
rewind(handler);
while (fgets(string, 100, handler)){
    char* ptr = strtok(string, " ");
    if (strcmp(ptr, "v") == 0){
        ptr = strtok(NULL, " ");
        while (i > -1){
            vertices[3*vert+i] = atof(ptr);
            ptr = strtok(NULL, " ");
            i++;
            if (ptr == NULL) i = -1;
        }
        i = 0;vert++;
    }
    else if (strcmp(ptr, "vt") == 0) {
        ……
    }
    else if (strcmp(ptr, "vn") == 0) {
        ……
    }
}
}
```

Extracting Vertex data

Extracting Texture data

Extracting Normal data

OBJ Loader Example -Cont.

```
else if (strcmp(ptr, "f") == 0) {
    if(text){
        ptr = strtok(NULL, "/");
        while (i > -1) {
            vindices[3 * face + i] = atoi(ptr) - 1;
            ptr = strtok(NULL, "/");
            tindices[3 * face + i] = atoi(ptr) - 1;
            ptr = strtok(NULL, " ");
            nindices[3 * face + i] = atoi(ptr) - 1;
            ptr = strtok(NULL, "/");
            i++;
            if ((ptr == NULL)||(i == 3)) i = -1;
        }
    }
    else{
    ptr = strtok(NULL, "/");
        while (i > -1) {
            vindices[3 * face + i] = atoi(ptr)-1;
            ptr = strtok(NULL, "/ ");
            nindices[3 * face + i] = atoi(ptr)-1;
            ptr = strtok(NULL, "/ ");
            i++;
            If ((ptr == NULL)||(i == 3)) i = -1;
        }
    }i = 0;face++;
}
```

Extracting Index with texture
(v0 /vt0/ vn0 v1 /vt1/ vn1 v2 /vt2/ vn2 Case)

Extracting Index w/o texture
(v0 // vn0 v1 // vn1 v2 // vn2 Case)

OBJ Loader Example –Cont2.

```
for (int i = 0 ; i < sizeofInd/3 ; i++){
    for(int j = 0; j < 3; j++){
        vertices[3*(3*i + j) + 0] = t_vert[3*vindices[3*i + j] + 0];
        vertices[3*(3*i + j) + 1] = t_vert[3*vindices[3*i + j] + 1];
        vertices[3*(3*i + j) + 2] = t_vert[3*vindices[3*i + j] + 2];
        if(text){
            textCoord[2*(3*i + j) + 0] = t_text[2*tindices[3*i + j] + 0];
            textCoord[2*(3*i + j) + 1] = t_text[2*tindices[3*i + j] + 1];
        }
        normals[3*(3*i + j) + 0] = t_norm[3*nindices[3*i + j] + 0];
        normals[3*(3*i + j) + 1] = t_norm[3*nindices[3*i + j] + 1];
        normals[3*(3*i + j) + 2] = t_norm[3*nindices[3*i + j] + 2];
    }
}
```

Re_Arrange data by index

Image

- We can draw OBJ File using VBO, EBO and VAO