アドバンスト CG
第 2 回レポート

学籍番号：201811411
所属：情報学群情報メディア創成学類
氏名：加藤虎之介

2021 年 4 月 27 日

1　実行環境

1.1　実行に用いた OS

macOS Big ver11.2.3

1.2　プログラム起動時に表示される情報

OpenGL version: 4.1 ATI-4.2.15
GLSL version: 4.10
Vendor: ATI Technologies Inc.
Renderer: AMD Radeon Pro 5300M OpenGL Engine

2　課題結果

課題 1

プログラム

Code 1　phong.vert

```
1 #version 150 core
2
3 in vec4 vertexPosition;
4 in vec3 vertexNormal;
5
6 // TODO: define "out" variables
7 out vec3 eNormal;
8 out vec3 ePosition;
```

```
9
10 // TODO: uncomment this line
11 uniform mat3 modelViewInverseTransposed;
12 uniform mat4 modelViewMatrix;
13 uniform mat4 projMatrix;
14
15 void main()
16 {
17         // TODO: rewirte this function
18         // 法線ベクトルの計算
19         // モデルビュー変換や投影行列といった変換行列に非一様スケーリングを含む場合、法線に対しただ変
             換行列をかけるだけではだめ。→モデルビュー行列の逆行列の転置行列をかければ良い。
20         eNormal = modelViewInverseTransposed * vertexNormal;
21
22         // 位置ベクトルの計算
23         ePosition = (modelViewMatrix * vertexPosition).xyz;
24
25         gl_Position = projMatrix * vec4(ePosition, 1);
26 }
```

Code 2　phong.frag

```
1 #version 150 core
2
3 // TODO: add "in" variables
4 in vec3 eNormal;
5 in vec3 ePosition;
6
7 out vec4 fragColor;
8
9 // TODO: add uniform variables
10 uniform vec3 eLightDir;
11 uniform vec3 lightColor;
12 uniform vec3 diffuseCoeff;
13 uniform vec3 ambient;
14 uniform float shininess;
15
16 void main()
17 {
18         // TODO: rewirte this function
19         // 拡散反射光の計算
20         vec3 eNormalNormalized = normalize(eNormal);
21         float dotDiffuse = max(dot(eNormalNormalized, eLightDir), 0.0);
22         vec3 diffuseColor = dotDiffuse * lightColor * diffuseCoeff;
23
24         // 鏡面反射光の計算
25         vec3 specularColor = vec3(0, 0, 0);
```

```
26          if (dotDiffuse > 0.0){
27                  vec3 viewingDir = normalize(-ePosition);
28                  vec3 eRefDir = normalize(reflect(-eLightDir, eNormalNormalized));
29                  float dotSpecular = max(dot(viewingDir, eRefDir), 0.0);
30                  specularColor = pow(dotSpecular, shininess) * lightColor * diffuseCoeff;
31          }
32
33          fragColor = vec4(ambient + diffuseColor + specularColor, 1);
34  }
```

Code 3　blinn_phong.vert

```
1   #version 150 core
2
3   // TODO: add "in" variables
4   in vec3 eNormal;
5   in vec3 ePosition;
6
7   out vec4 fragColor;
8
9   // TODO: add uniform variables
10  uniform vec3 eLightDir;
11  uniform vec3 lightColor;
12  uniform vec3 diffuseCoeff;
13  uniform vec3 ambient;
14  uniform float shininess;
15
16  void main()
17  {
18          // TODO: rewirte this function
19          // 拡散反射光の計算
20          vec3 eNormalNormalized = normalize(eNormal);
21          float dotDiffuse = max(dot(eNormalNormalized, eLightDir), 0.0);
22          vec3 diffuseColor = dotDiffuse * lightColor * diffuseCoeff;
23
24          // 鏡面反射光の計算
25          vec3 specularColor = vec3(0, 0, 0);
26          if (dotDiffuse > 0.0) {
27                  vec3 viewingDir = normalize(-ePosition);
28                  vec3 halfVec = normalize(eLightDir + viewingDir);
29                  float dotSpecular = max(dot(eNormalNormalized, halfVec), 0.0);
30                  specularColor = pow(dotSpecular, shininess) * lightColor * diffuseCoeff;
31          }
32
33          fragColor = vec4(ambient + diffuseColor + specularColor, 1);
34  }
```

Code 4　blinn_phong.frag

```
#version 150 core

// TODO: add "in" variables
in vec3 eNormal;
in vec3 ePosition;

out vec4 fragColor;

// TODO: add uniform variables
uniform vec3 eLightDir;
uniform vec3 lightColor;
uniform vec3 diffuseCoeff;
uniform vec3 ambient;
uniform float shininess;

void main()
{
        // TODO: rewirte this function
        // 拡散反射光の計算
        vec3 eNormalNormalized = normalize(eNormal);
        float dotDiffuse = max(dot(eNormalNormalized, eLightDir), 0.0);
        vec3 diffuseColor = dotDiffuse * lightColor * diffuseCoeff;

        // 鏡面反射光の計算
        vec3 specularColor = vec3(0, 0, 0);
        if (dotDiffuse > 0.0) {
                vec3 viewingDir = normalize(-ePosition);
                vec3 halfVec = normalize(eLightDir + viewingDir);
                float dotSpecular = max(dot(eNormalNormalized, halfVec), 0.0);
                specularColor = pow(dotSpecular, shininess) * lightColor * diffuseCoeff;
        }

        fragColor = vec4(ambient + diffuseColor + specularColor, 1);
}
```

実行結果

- Phong Model

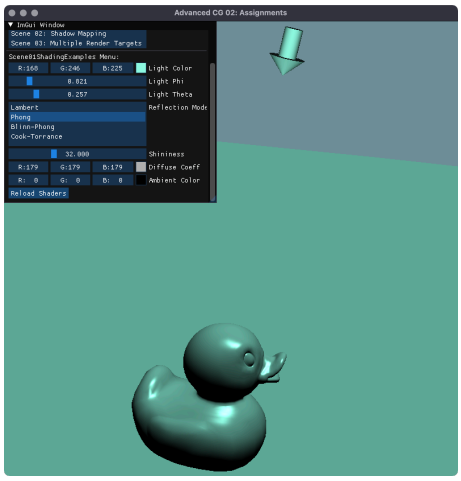

図 1　Phong Model

- Blinn-Phong Model

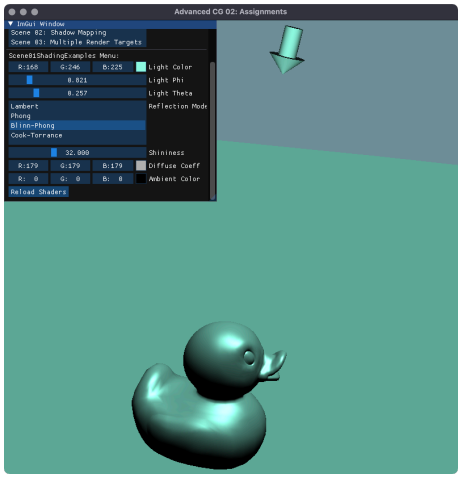

図 2　Blinn-Phong Model

課題 2

プログラム

Code 5　shadow_blinn_phong.vert

```
1 #version 150 core
2
3 in vec4 vertexPosition;
4 in vec3 vertexNormal;
5
6 // TODO: define "out" variables
7 out vec3 eNormal;
```

```
8  out vec3 ePosition;
9  out vec4 texCoord; // シャドウマップ参照用のテクスチャ座標
10
11 // TODO: uncomment these lines
12 uniform mat4 biasedShadowProjModelView;
13 uniform mat3 modelViewInverseTransposed;
14 uniform mat4 projMatrix;
15 uniform mat4 modelViewMatrix;
16
17 void main()
18 {
19         // TODO: rewirte this function
20         // [ビュー座標系]法線ベクトルの計算
21         // モデルビュー変換や投影行列といった変換行列に非一様スケーリングを含む場合、法線に対しただ変
               換行列をかけるだけではだめ。→モデルビュー行列の逆行列の転置行列をかければ良い。
22         eNormal = modelViewInverseTransposed * vertexNormal;
23
24         // [ビュー座標系]位置ベクトルの計算
25         ePosition = (modelViewMatrix * vertexPosition).xyz;
26
27         // [クリッピング座標系]シャドウマップ参照用のテクスチャ座標を計算
28         texCoord = biasedShadowProjModelView * vertexPosition;
29
30         // [クリッピング座標系]頂点座標の計算
31         gl_Position = projMatrix * vec4(ePosition, 1);
32 }
```

Code 6　shadow_blinn_phong.frag

```
1  #version 150 core
2
3  // TODO: define "in" variables
4  in vec3 eNormal;
5  in vec3 ePosition;
6  in vec4 texCoord; // シャドウマップ参照用のテクスチャ座標
7
8  out vec4 fragColor;
9
10 // TODO: uncomment this line
11 uniform sampler2DShadow shadowTex;
12 // TODO: define uniform variables
13 uniform vec3 eLightDir;
14 uniform vec3 lightColor;
15 uniform vec3 diffuseCoeff;
16 uniform vec3 ambient;
17 uniform float shininess;
18
```

```
19 void main()
20 {
21         // TODO: rewirte this function
22         // 拡散反射光の計算
23         vec3 eNormalNormalized = normalize(eNormal);
24         float dotDiffuse = max(dot(eNormalNormalized, eLightDir), 0.0);
25         vec3 diffuseColor = dotDiffuse * lightColor * diffuseCoeff;
26
27         // 鏡面反射光の計算
28         vec3 specularColor = vec3(0, 0, 0);
29         if (dotDiffuse > 0.0) {
30                 vec3 viewingDir = normalize(-ePosition);
31                 vec3 halfVec = normalize(eLightDir + viewingDir);
32                 float dotSpecular = max(dot(eNormalNormalized, halfVec), 0.0);
33                 specularColor = pow(dotSpecular, shininess) * lightColor * diffuseCoeff;
34         }
35
36         fragColor = vec4(ambient + diffuseColor + specularColor, 1) * textureProj(
            shadowTex, texCoord);
37 }
```

Code 7　pcf_shadow_blinn_phong.vert

```
1  #version 150 core
2
3  in vec4 vertexPosition;
4  in vec3 vertexNormal;
5
6  // TODO: define "out" variables
7  out vec3 eNormal;
8  out vec3 ePosition;
9  out vec4 texCoord; // シャドウマップ参照用のテクスチャ座標
10
11 // TODO: uncomment these lines
12 uniform mat4 biasedShadowProjModelView;
13 uniform mat3 modelViewInverseTransposed;
14 uniform mat4 projMatrix;
15 uniform mat4 modelViewMatrix;
16
17 void main()
18 {
19         // TODO: rewirte this function
20         // [ビュー座標系]法線ベクトルの計算
21         // モデルビュー変換や投影行列といった変換行列に非一様スケーリングを含む場合、法線に対しただ変
             換行列をかけるだけではだめ。→モデルビュー行列の逆行列の転置行列をかければ良い。
22         eNormal = modelViewInverseTransposed * vertexNormal;
23
```

```
24          // [ビュー座標系]位置ベクトルの計算
25          ePosition = (modelViewMatrix * vertexPosition).xyz;
26
27          // [クリッピング座標系]シャドウマップ参照用のテクスチャ座標を計算
28          texCoord = biasedShadowProjModelView * vertexPosition;
29
30          // [クリッピング座標系]頂点座標の計算
31          gl_Position = projMatrix * vec4(ePosition, 1);
32  }
```

Code 8　pcf_shadow_blinn_phong.frag

```
1   #version 150 core
2
3   // TODO: define "in" variables
4   in vec3 eNormal;
5   in vec3 ePosition;
6   in vec4 texCoord; // シャドウマップ参照用のテクスチャ座標
7
8   out vec4 fragColor;
9
10  uniform sampler2DShadow shadowTex;
11  uniform vec2 texMapScale;
12  // TODO: define uniform variables
13  uniform vec3 eLightDir;
14  uniform vec3 lightColor;
15  uniform vec3 diffuseCoeff;
16  uniform vec3 ambient;
17  uniform float shininess;
18
19  float offsetLookup(sampler2DShadow map, vec4 loc, vec2 offset)
20  {
21          return textureProj(map, vec4(loc.xy + offset * texMapScale * loc.w, loc.z, loc.
              w));
22  }
23
24  float samplingLimit = 3.5;
25
26  void main()
27  {
28          // HINT: The visibility (i.e., shadowed or not) can be fetched using the "
              offsetLookup" function
29          // in the double loops. The y and x loops ranges from -samplingLimit to
              samplingLimit
30          // with a stepsize 1.0. Finally, the summed visibility is divided by 64 to
              normalize.
31          float targetShadowTexVal = 0.0;
```

```
32          for (float y = -samplingLimit; y <= samplingLimit; y += 1.0) {
33                  for (float x = -samplingLimit; x <= samplingLimit; x += 1.0) {
34                          vec2 offset = vec2(x, y);
35                          float shadowTexVal = offsetLookup(shadowTex, texCoord, offset);
36                          targetShadowTexVal += shadowTexVal;
37                  }
38          }
39          targetShadowTexVal /= 64.0;
40
41          // TODO: rewirte this function
42          // 拡散反射光の計算
43          vec3 eNormalNormalized = normalize(eNormal);
44          float dotDiffuse = max(dot(eNormalNormalized, eLightDir), 0.0);
45          vec3 diffuseColor = dotDiffuse * lightColor * diffuseCoeff;
46
47          // 鏡面反射光の計算
48          vec3 specularColor = vec3(0, 0, 0);
49          if (dotDiffuse > 0.0) {
50                  vec3 viewingDir = normalize(-ePosition);
51                  vec3 halfVec = normalize(eLightDir + viewingDir);
52                  float dotSpecular = max(dot(eNormalNormalized, halfVec), 0.0);
53                  specularColor = pow(dotSpecular, shininess) * lightColor * diffuseCoeff;
54          }
55
56          fragColor = vec4(vec3(ambient + diffuseColor + specularColor) *
              targetShadowTexVal, 1);
57          // fragColor = vec4(ambient + diffuseColor + specularColor, 1) *
              targetShadowTexVal;
58  }
```

実行結果

- Shadow Mapping

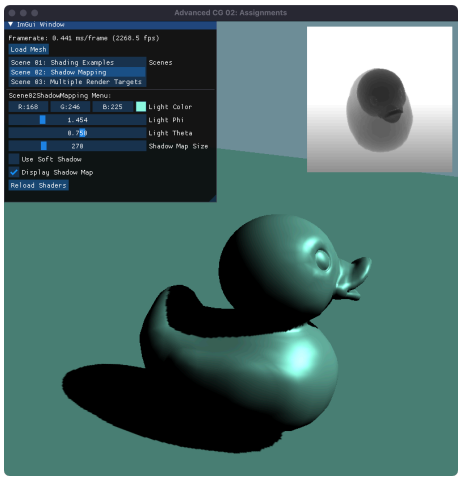

図 3　Shadow Mapping

- Percentage-closer filter によるソフトシャドウ

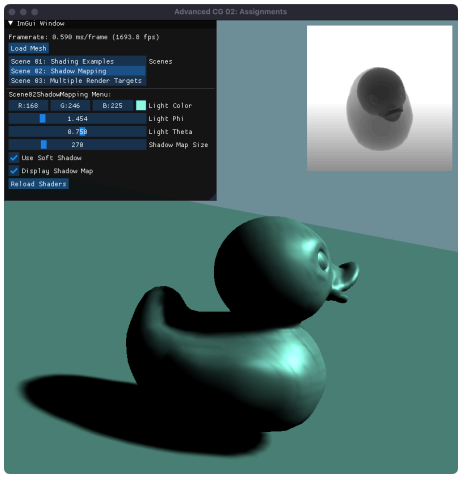

図 4　PCF

課題 3

プログラム

Code 9　mrt.vert

```
1 #version 150 core
2
3 in vec4 vertexPosition;
4 in vec3 vertexNormal;
5
6 // TODO: define "out" variables
7 out vec3 viewVertexNormal; // ビュー座標系(カメラ座標系)における法線ベクトル
```

```
8  out vec4 viewVertexPosition; // カメラ座標系における座標
9  out vec4 deviceVertexPosition; //正規化デバイス座標系における座標
10
11 uniform mat4 projMatrix;
12 uniform mat4 modelViewMatrix;
13 // add
14 uniform mat3 modelViewInvTransposed;
15
16 void main()
17 {
18         // TODO: rewrite this function
19         gl_Position = projMatrix * modelViewMatrix * vertexPosition;
20
21         viewVertexNormal = normalize(modelViewInvTransposed * vertexNormal);
22         viewVertexPosition = modelViewMatrix * vertexPosition;
23         deviceVertexPosition = projMatrix * viewVertexPosition;
24 }
```

Code 10　mrt.frag

```
1  #version 150 core
2
3  // TODO: define "in" variables
4  in vec3 viewVertexNormal;
5  in vec4 viewVertexPosition;
6  in vec4 deviceVertexPosition;
7
8  out vec4 fragColors[3];
9
10 void main()
11 {
12         fragColors[0] = viewVertexPosition; // 座標
13         fragColors[1] = vec4(0.5f * viewVertexNormal + 0.5f, 1.f); // 法線
14
15         float z = 0.5 * deviceVertexPosition.z + 0.5;
16         fragColors[2] = vec4(z, z, z, 1); // デプス値
17 }
```

実行結果

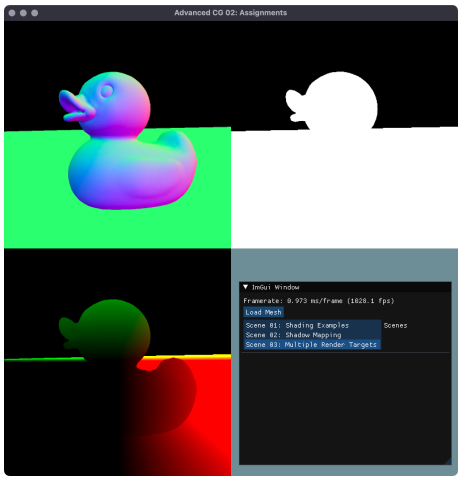

図 5　Multiple Render Target