2023학년도 1학기

컴퓨터애니메이션실습

컴퓨터 애니메이션

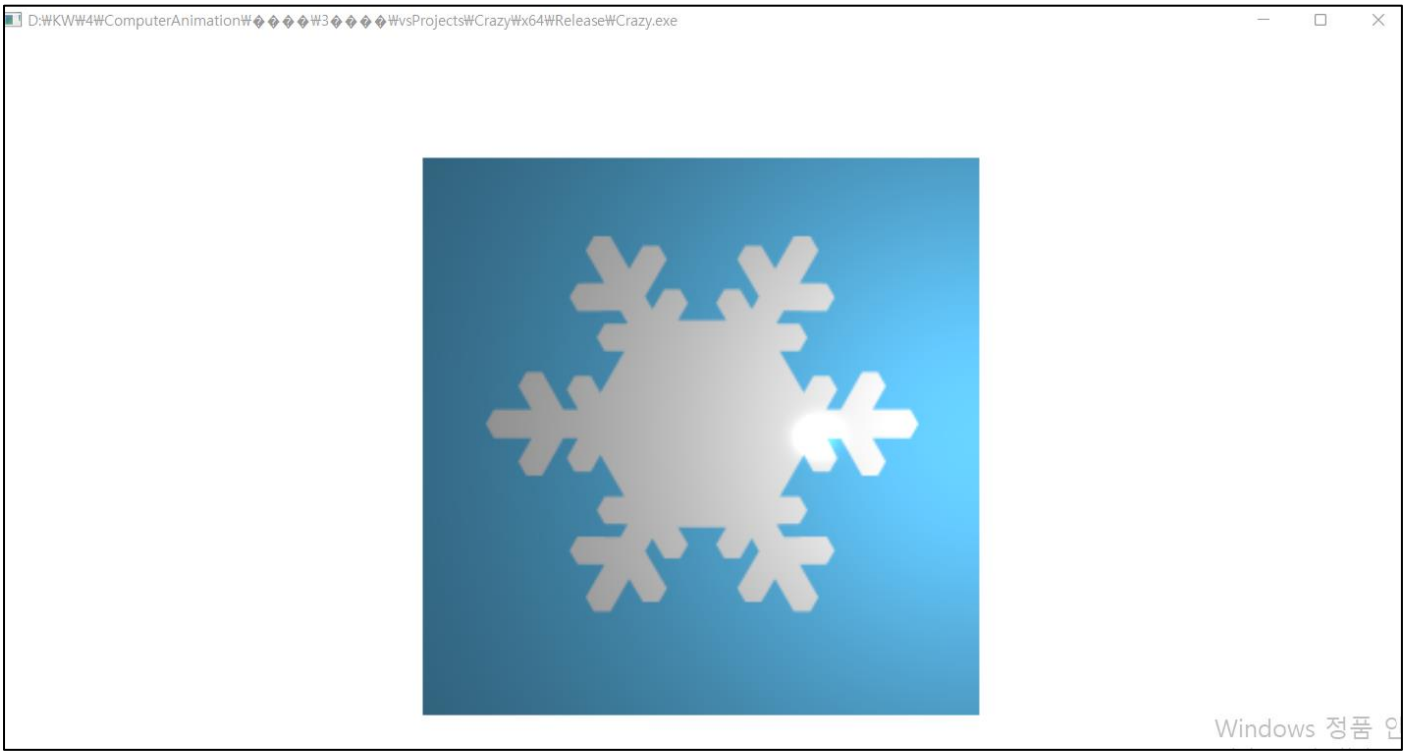
실습 보고서

Self-Scoring Table

	P1	P2	P3	P4	E1	Total
Score	1	1	1	1	1	5

정보융합학부 2018204058 김민교

P1 - Simple texturing



Simple texturing은 “m02_snow_color_map”파일을 이용해 quad에 텍스처를 입힌 것이다. PhongReflectionModel을 사용해서 라이팅을 진행한다.

Vertex shdaer에서 텍스처 좌표를 fragment shader에서 input으로 보낸다.

```
74        // Texture
75        vec4 color = texture(tex,texcoord); //
```

Fragment shader에서 텍스처와 텍스처 좌표를 보고 색깔을 가져온다.

```
77        // Lighting
78        outColor.rgb = phongReflectionModel(L0, position, normalize(normal))*color.rgb;
79        outColor.a = color.a;
```

이전 과제에서는 phongReflectionModel의 반환값이 색깔이었지만, 여기서는 반환값은 빛의 세기를 의미한다. 빛의 세기와 컬러를 곱해서 픽셀의 색깔을 결정한다.

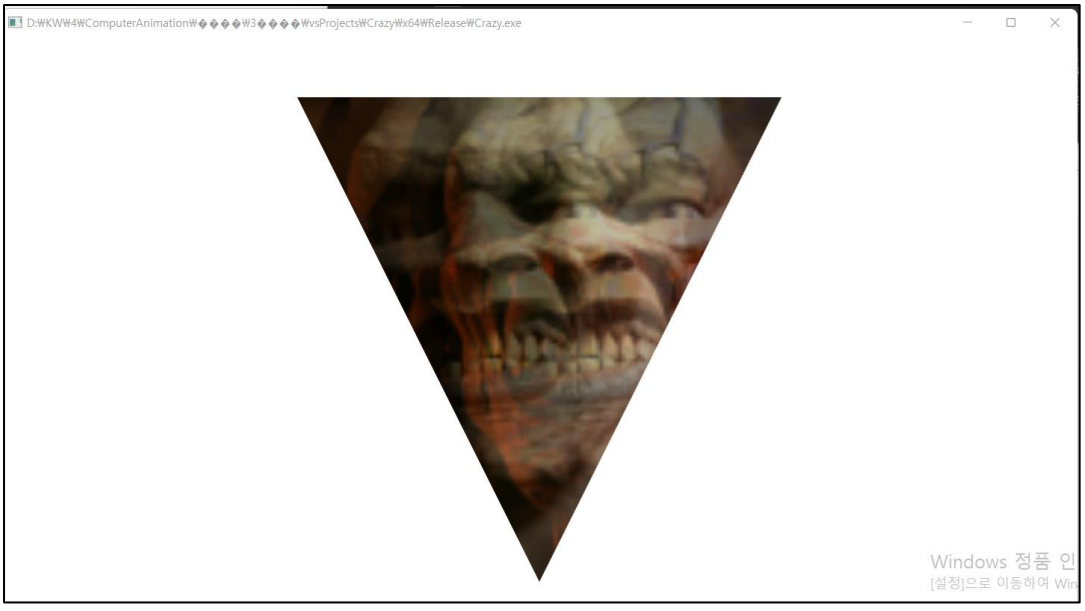
P2 - Alpha texturing

Alpha texturing은 “m02_logo.raw” 파일을 이용했다.

```
674         // Alpha texturing on
675         glEnable(GL_BLEND);
676         glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
677         isOK("glBlendFunc()", __FILE__, __LINE__);
```

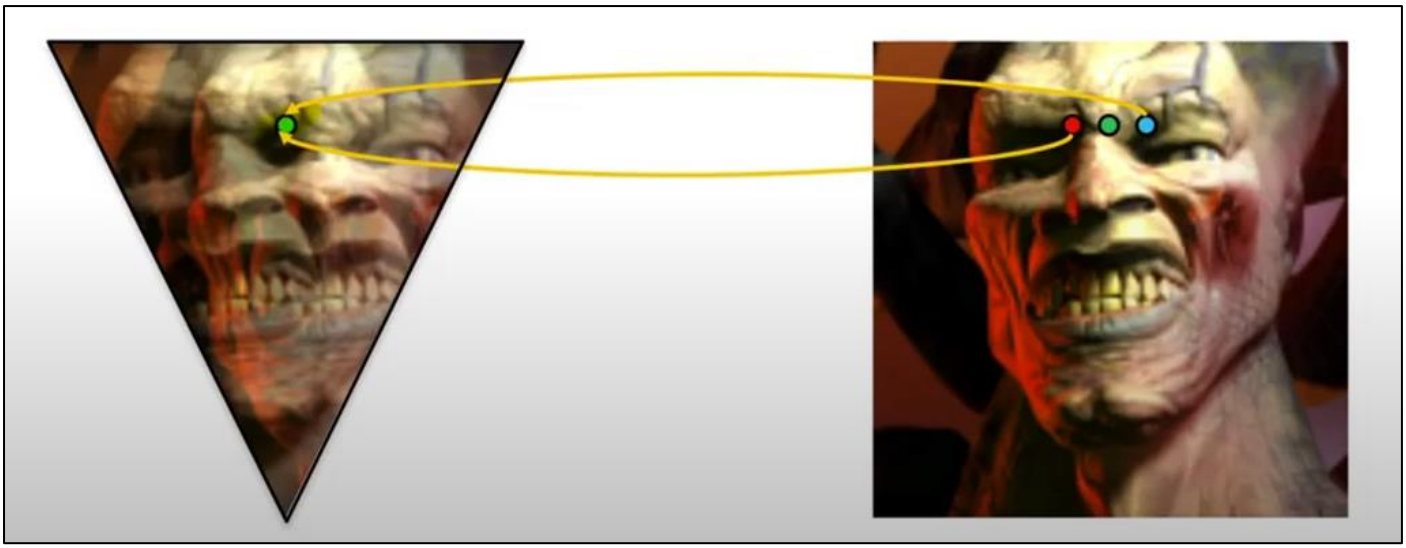
Alpha texturing을 하려면 glEnable(GL_BLEND)를 사용해야한다.

P3 - Double vision



Double vision은 “m02_demon_image.h”파일을 이용해 trianlge에 텍스처를 입힌 것이다.

PhongReflectionModel을 사용해서 라이팅을 진행한다.



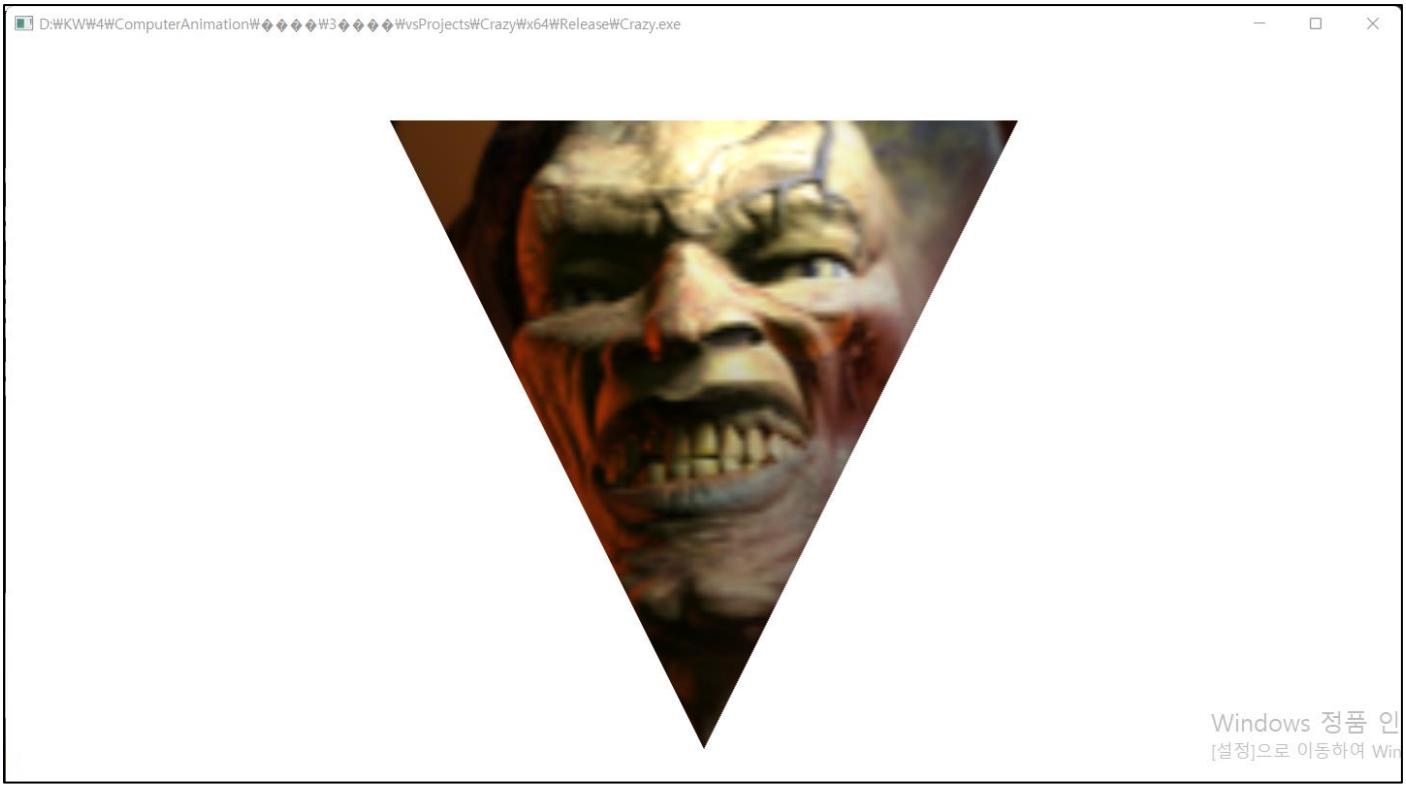
위의 그림과 같이 구현하려면 기준점에서 왼쪽 또는 오른쪽으로 얼만큼 떨어진 텍스처 좌표를 입힐 것인지를 결정해야 한다.

```
32        // Texture coordinates
33        leftTexcoord = VertexTexcoord  + leftSeparation;
34        rightTexcoord = VertexTexcoord + rightSeparation;
35    }
```

VertexShader에서 기존의 VertexTexcoord에 왼쪽으로 얼만큼 떨어진지의 값을 더하고, 기존의 VertexTexcoord에 오른쪽으로 얼만큼 떨어진지의 값을 더한다.

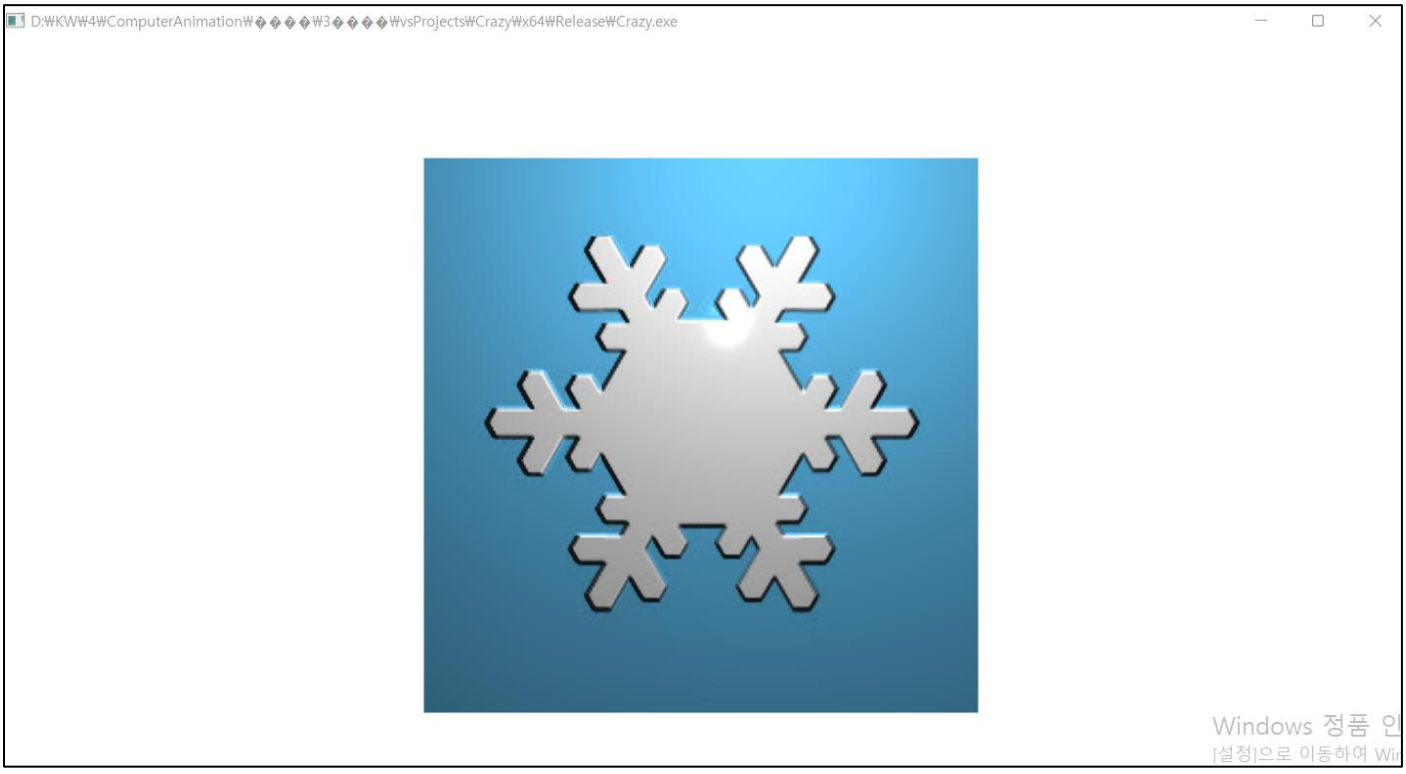
```
72    void
73    main(void)
74    {
75        // Texture
76        vec3 leftColor = texture(tex,leftTexcoord).rgb;
77        vec3 rightColor = texture(tex, rightTexcoord).rgb;
78        vec3 color = mix ( leftColor, rightColor,0.5); // interpolation
79
80        // Lighting
81        outColor = phongReflectionModel(L0, position, normalize(normal))*color;
82    }
```

FragmentShader에서 텍스처와 텍스처 좌표를 통해 색깔을 결정하고 보간한다. 그리고 픽셀에 색을 출력한다.

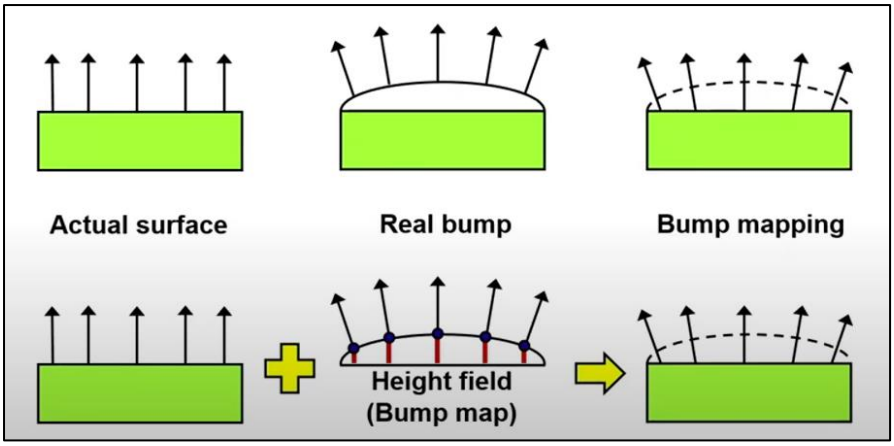


스페이스바를 눌러서 합친 모습.

P4 - Normal mapping



Normal mapping은 “m02_snow_normal_map.raw”을 이용한다.



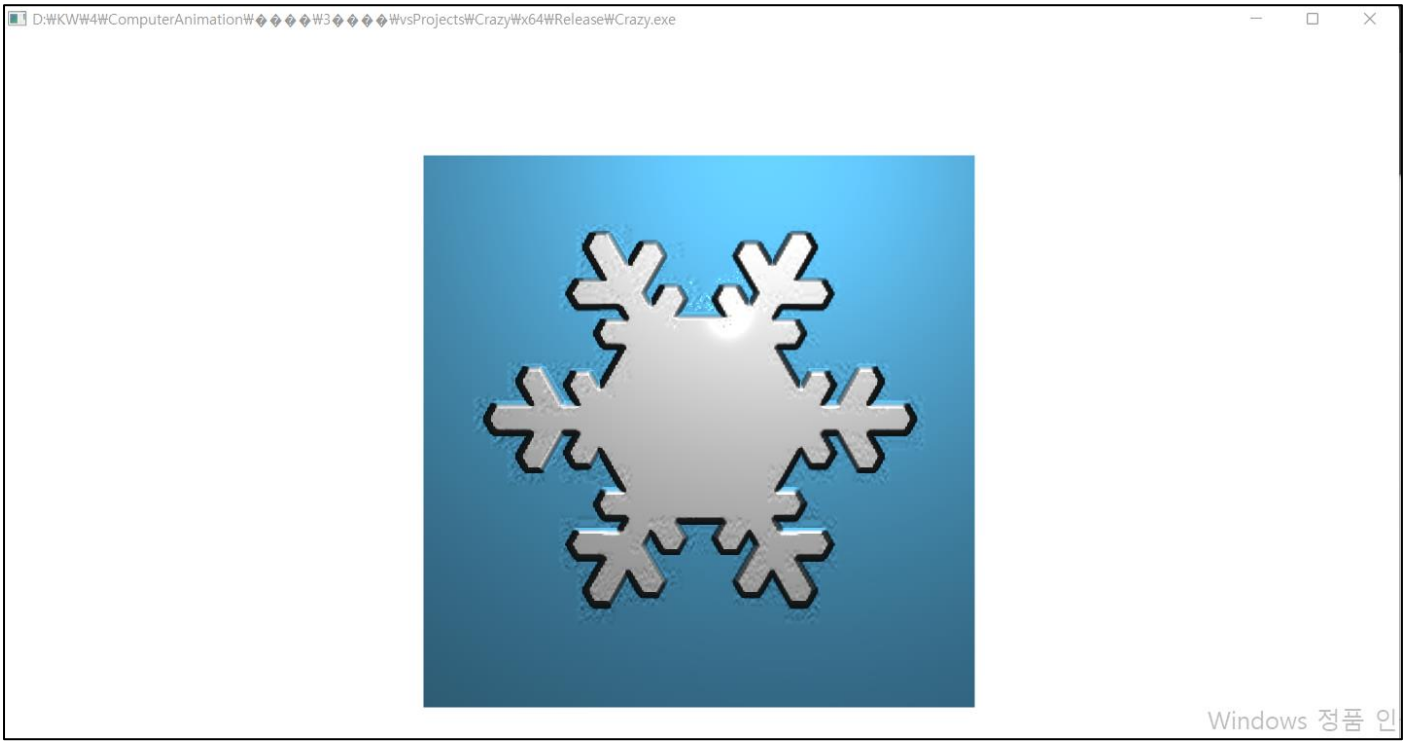
실제로 굴곡 있는 물체를 3D로 표현하기 위해서 많은 폴리곤들이 필요하다. 그리고 그 폴리곤의 노말벡터를 통해 라이팅을 진행해야 사실감 있는 물체를 그릴 수 있다. 그러나 폴리곤이 많아지고 그에 따라 노말벡터도 많아지면 실시간으로 계산해야 하는 량이 많다. 그런걸 방지하기 위해 노말 맵을 사용한다.

```
641         // Textures
642         setUniformi(pgNormalMapping.pg, "texDiffuse", 0);//컬러텍스처 // 0 for GL_TEXTURE0
643         setUniformi(pgNormalMapping.pg, "texNormal", 3);//노말텍스처 // 3 for GL_TEXTURE3
```

Shader로 데이터를 보낼 때, 텍스처를 두개 보낸다. 하나는 컬러텍스처, 하나는 노말 텍스처이다.

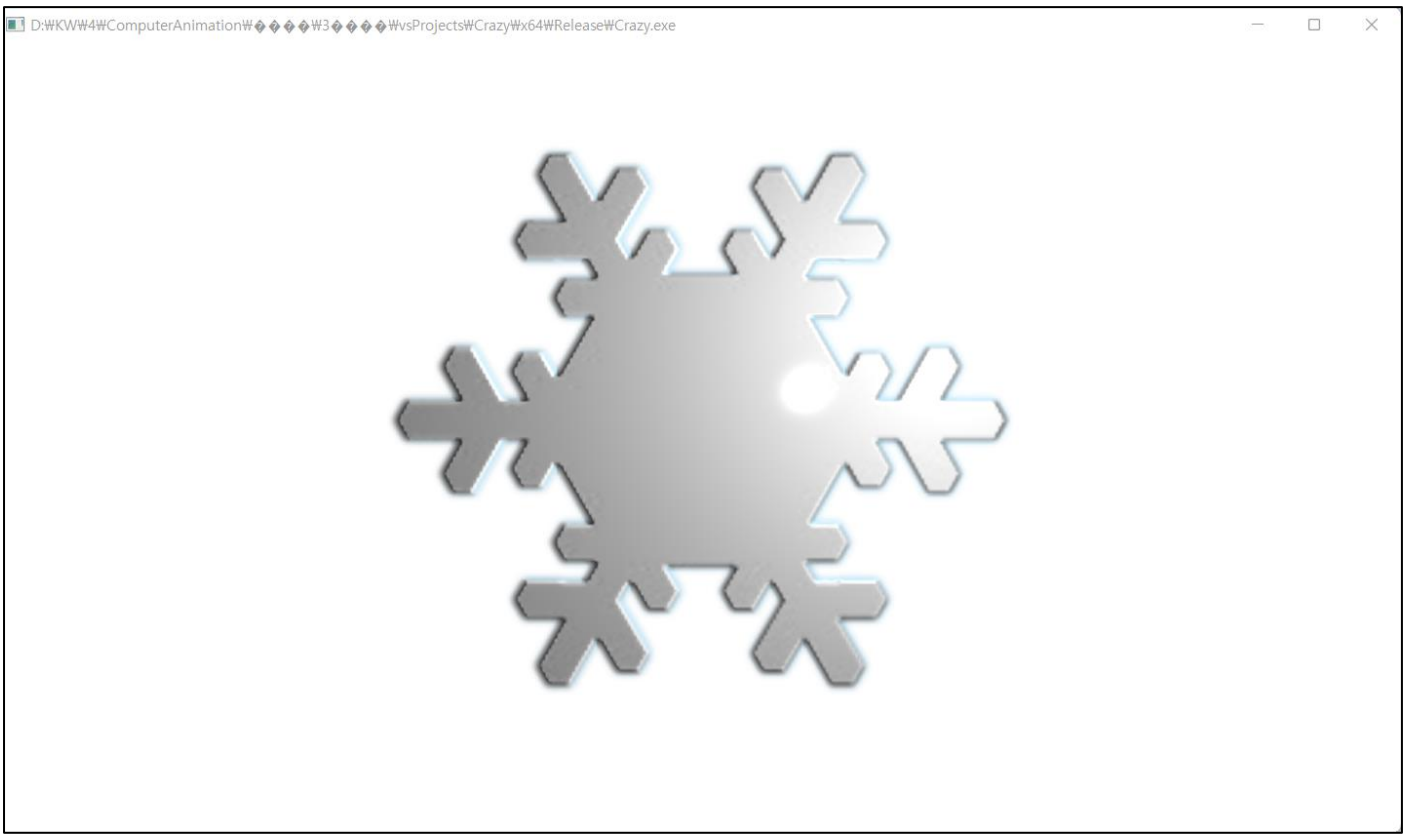
```
66      vec3    N = texture(texNormal, texcoord).rgb;
67              N = 2 * N - 1;
68              N.x *= scale;   N.y *= scale;
69              N = normalize(N);
```

Fragment Shader에서 노말 텍스처와 텍스처 좌표로 적합한 노말벡터를 구한다.



scale을 높인 모습. scale을 높인다는 것은 height를 높인다는 것을 말한다. 높아질수록 그림자가 더 지게된다.

E1 - Draw a color-, alpha-, normal - mapped snow



color, alpha, normal 맵핑을 전부 적용한 결과이다.

- **코드**

loadColorAlphaTexture 함수 구현

```
258     // Open the raw texture file (color)
259     ifstream    isC(filenameC, ios::binary);
260     if (isC.fail())
261     {
262         cout << "Can't open " << filenameC << endl;
263         return false;
264     }
265
266     // Open the raw texture file (alpha)
267     ifstream    isA(filenameA, ios::binary);
268     if (isA.fail())
269     {
270         cout << "Can't open " << filenameA << endl;
271         return false;
272     }
273
```

먼저, 컬러 텍스처와 알파 텍스처를 각각 읽어온다.

```
274     // Allocate memory for Color & Alpha
275     GLubyte* raw = new GLubyte[w * h * 4];  //RGBA
276     {
277         // Allocate memory for color
278         GLubyte* rawC = new GLubyte[w * h * 3];
279         // Read all the texels of the rgb channels
280         isC.read((char*)rawC, w * h * 3);
281 
282         // Alpha값을 읽는다.
283         GLubyte* rawA = new GLubyte[w * h];
284         // Read all the texels
285         isA.read((char*)rawA, w * h);
286         if (!isA)       cout << "Error: only" << isA.gcount() << "bytes could be read!" << endl;
287 
288         for (int i = 0; i < w * h; i++)
289         {
290             raw[4 * i + 0] = rawC[3*i+0];       // R
291             raw[4 * i + 1] = rawC[3 * i + 1];   // G
292             raw[4 * i + 2] = rawC[3 * i + 2];   // B
293             raw[4 * i + 3] = rawA[i];           // A
294         }
295         delete[] rawA;
296         delete[] rawC;
297     }
```

raw에는 최종 텍스처의 RGBA값이 들어올 것이다.

color만을 담기 위한 변수로 rawC를 선언하고, 컬러 텍스처를 통해 RGB값을 담는다.

alpha값 만을 담기 위한 변수로 rawA를 선언하고, 알파 텍스처를 통해 A값을 담는다

288~294 줄의 반복문을 통해 raw 변수에 R, G, B, A를 담는다.

renderColorAlphaNormalMappedQuad 함수 구현

```
708     // Model matrix
709     Affine3f    T;  T = Translation3f(0.0f, 0.0f, 0.0f)* Scaling(1.0f, 1.0f, 1.0f);
710     Matrix4f    ModelMatrix = T.matrix();
```

ModelMatrix를 정의한다.

```
715     // Alpha texturing on
716     glEnable(GL_BLEND);
717     glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
718     isOK("glBlendFunc()", __FILE__, __LINE__);
```

Alpha texturing을 위해 glEnable(GL_BLEND)를 선언한다.

```
720         // Textures
721         setUniformi(pgNormalMapping.pg, "texDiffuse", 4);//컬러텍스처 // 4 for GL_TEXTURE4
722         setUniformi(pgNormalMapping.pg, "texNormal", 3);//노말텍스처 // 3 for GL_TEXTURE3
723 
724         // Scale
725         setUniform(pgNormalMapping.pg, "scale", scale); // Height scale for normal mapping
726 
727         // Light Position
728         setUniform(pgNormalMapping.pg, "LightPosition", l);
729 
730         // Material is dependent of the object
731         setUniform(pgNormalMapping.pg, "Ka", Vector3f(0.10f, 0.10f, 0.10f));
732         setUniform(pgNormalMapping.pg, "Kd", Vector3f(0.95f, 0.95f, 0.95f));
733         setUniform(pgNormalMapping.pg, "Ks", Vector3f(0.50f, 0.50f, 0.50f));
734         setUniform(pgNormalMapping.pg, "Shininess", 128.0f);
735 
736         // Draw the mesh using the program and the vertex buffer object
737         glUseProgram(pgNormalMapping.pg);
738         drawVBO(quad.vao, quad.numTris);
739 
740         // Alpha texturing off
741         glDisable(GL_BLEND);
742     }
```

loadColorAlphaTexture 함수로 얻은 컬러 + 알파 텍스처와 노말 텍스처를 쉐이더로 보낸다