2023학년도 1학기

컴퓨터애니메이션실습

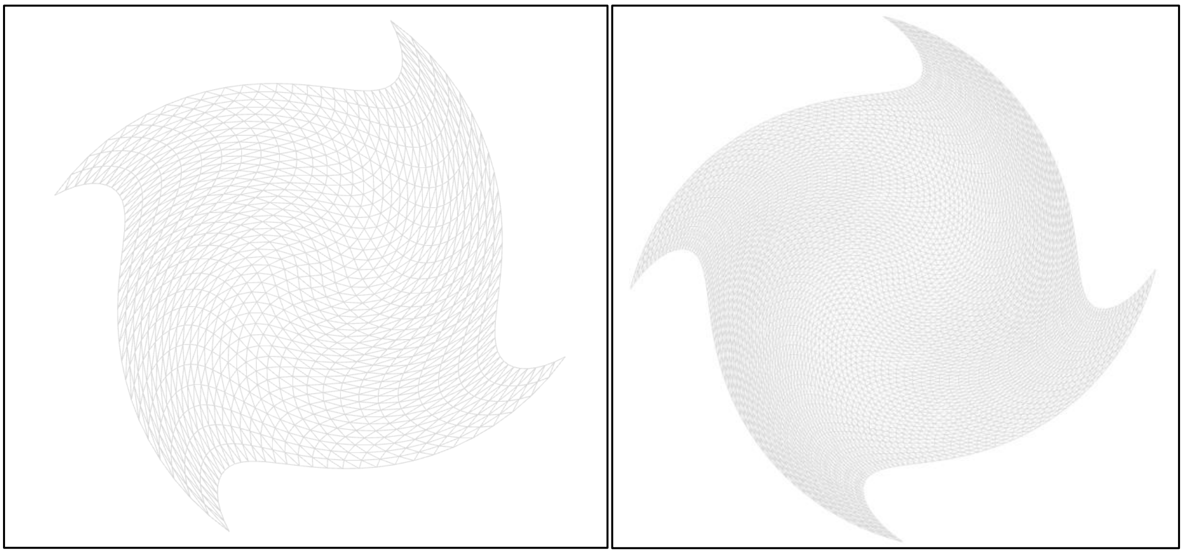
컴퓨터 애니메이션 실습 보고서

Self-Scoring Table

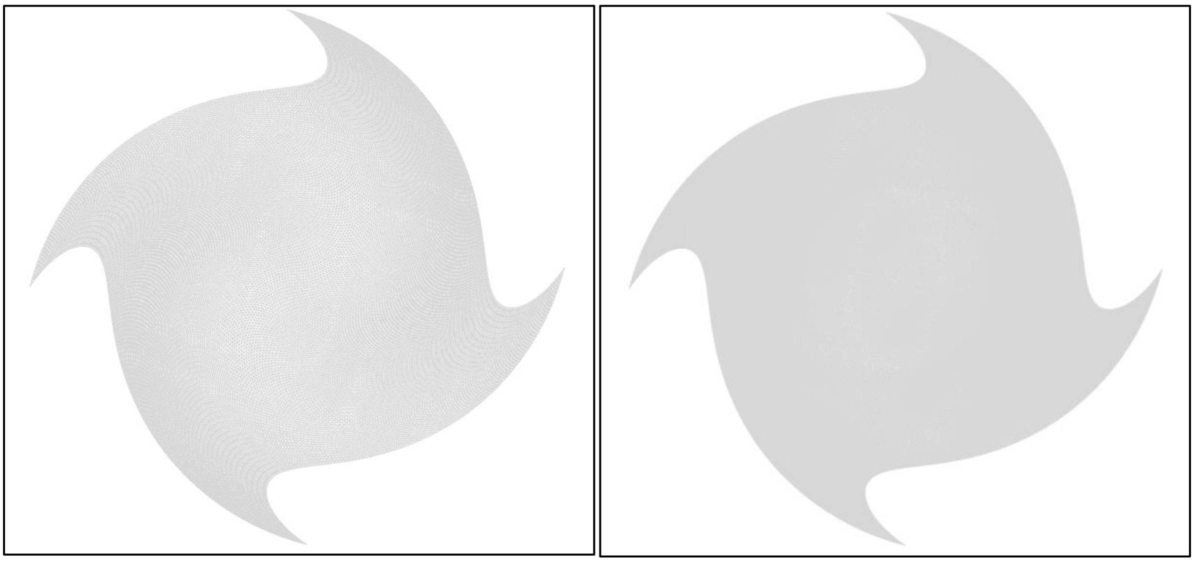
	P1	P2	E1	Total
Score	1	1	1	3

정보융합학부 2018204058 김민교

P1 - Twisting Deformer



plane32.off, plane64.off 파일을 트위스트한 결과이다.



plane128.off, plane256.off 파일을 트위스트한 결과이다.

[Position]

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

트위스트를 구현하려면 x, y를 z축을 중심으로 회전해야 한다. 원 x, y에 회전행렬을 적용시켜서 트위스트 후 버텍스 포지션을 구할 수 있다.

```
    // New position due to twisting
    // 2D 회전
    float x = cosLength * VertexPosition.x - sinLength * VertexPosition.y;
    float y = sinLength * VertexPosition.x + cosLength * VertexPosition.y;
    // x,y만 바뀌고 z 값은 그대로
    vec4    newVertexPosition = vec4(x,y,VertexPosition.z,1.0);
```

버텍스 쉐이더에서 구현한 코드이다.

[Normal]

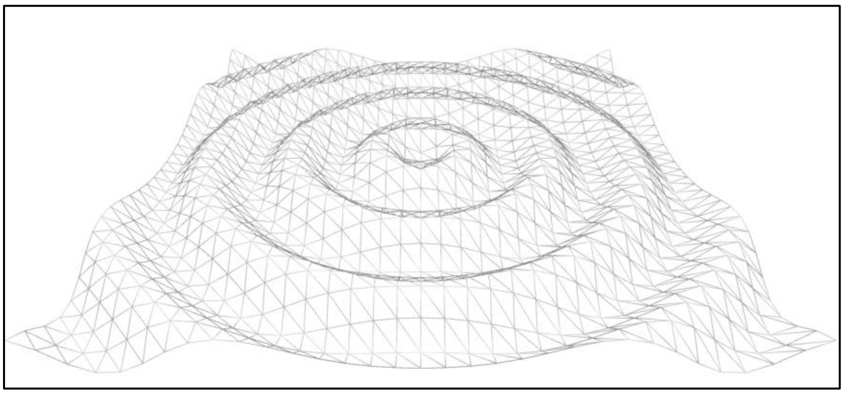
$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}^{-T} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}^{T} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Normal 벡터를 트랜스폼 하기위해선 트랜스폼 매트릭스의 전치행렬의 역행렬을 곱해줘야 한다. 회전행렬의 역행렬과 전치행렬은 똑같아서 그냥 원래 행렬이랑 똑같다.

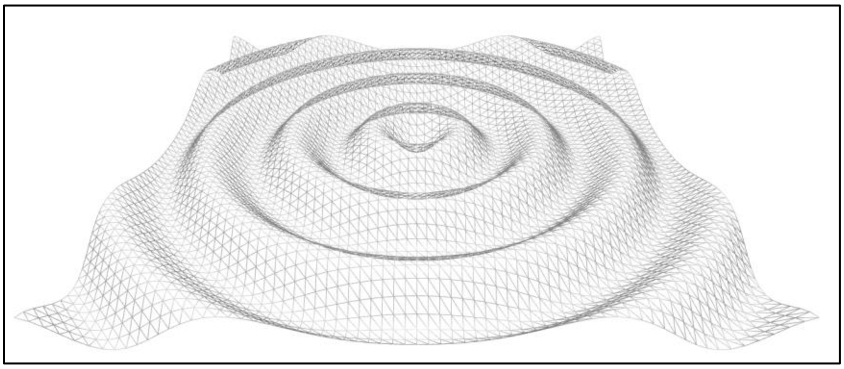
```
    // New normal due to twisting
    x = cosLength * VertexNormal.x - sinLength*VertexNormal.y;
    y = sinLength * VertexNormal.x + cosLength*VertexNormal.y;
    vec3    newVertexNormal = vec3(x,y,VertexNormal.z);
```

버텍스 쉐이더에서 구현한 코드이다.

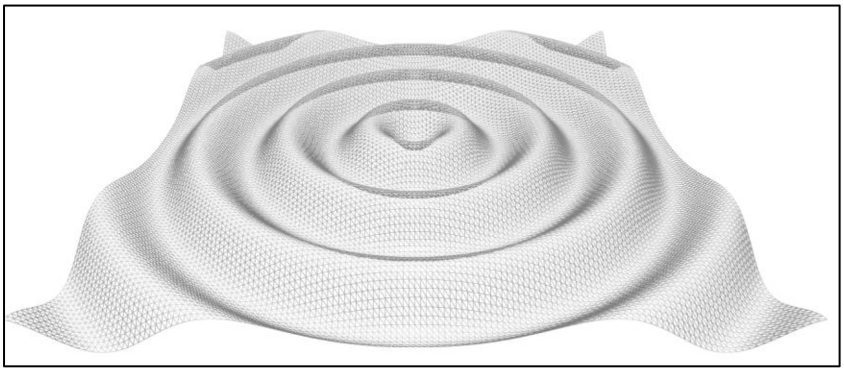
P2 - Wave Deformer



plane32.off 파일을 웨이브한 결과이다.



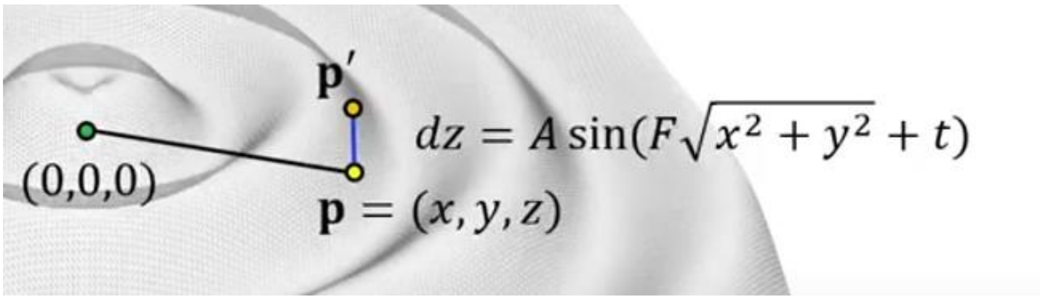
plane64.off 파일을 웨이브한 결과이다.



plane128.off 파일을 웨이브한 결과이다.

plane256.off 파일을 웨이브한 결과이다.

[Position]



웨이브 디포머는 표면이 위 아래로 움직이는 것이다. 그래서 일단 여기서는 z축이 변하면 된다. 웨이브 된 z축을 구하기 위해서는 "원래 z + z변화량" 으로 구할 수 있다.

```
    // Vertex position
    float l = length(VertexPosition.xy);
    float z = VertexPosition.z + A*sin(F*l + phase);
    vec4  newVertexPosition = vec4(VertexPosition.xy,z,1.0);
```

버텍스 쉐이더에서 구현한 코드이다.

[Normal]

Vertex normal

- Inverse transpose of the Jacobian of the deformation function

$$\mathbf{J}^{-\mathrm{T}}\mathbf{n} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \dfrac{A\cos(F\sqrt{x^2+y^2}+t)\,Fx}{\sqrt{x^2+y^2}+\epsilon} & \dfrac{A\cos(F\sqrt{x^2+y^2}+t)\,Fy}{\sqrt{x^2+y^2}+\epsilon} & 1 \end{bmatrix}^{-\mathrm{T}} \mathbf{n}$$

노말벡터를 트랜스폼하기 위해선 자코비안 행렬의 전치행렬의 역행렬을 구하면된다.

- Deformation function

$$\mathbf{f}(x,y,z) = \left(x, y, z + A\sin(F\sqrt{x^2+y^2}+t)\right)$$

Jacobian of the deformation function

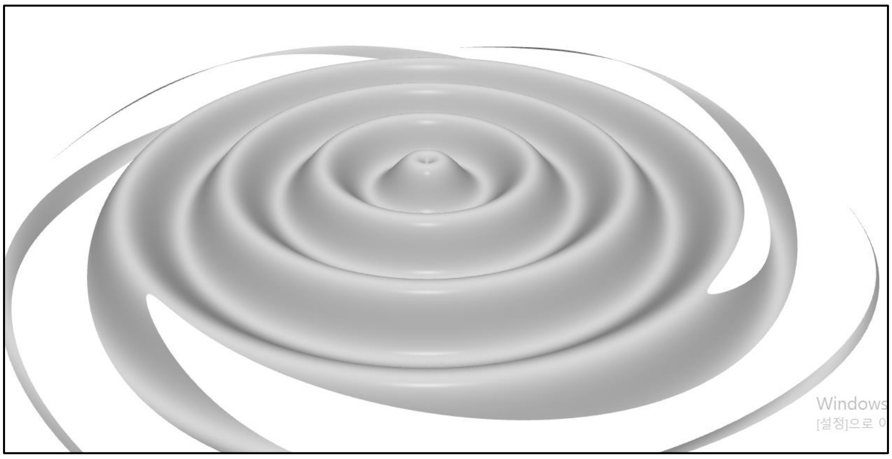
$$\mathbf{J} = \begin{bmatrix} \frac{\partial \mathbf{f}_x}{\partial x} & \frac{\partial \mathbf{f}_x}{\partial y} & \frac{\partial \mathbf{f}_x}{\partial z} \\ \frac{\partial \mathbf{f}_y}{\partial x} & \frac{\partial \mathbf{f}_y}{\partial y} & \frac{\partial \mathbf{f}_y}{\partial z} \\ \frac{\partial \mathbf{f}_z}{\partial x} & \frac{\partial \mathbf{f}_z}{\partial y} & \frac{\partial \mathbf{f}_z}{\partial z} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \dfrac{A\cos(F\sqrt{x^2+y^2}+t)\,Fx}{\sqrt{x^2+y^2}+\epsilon} & \dfrac{A\cos(F\sqrt{x^2+y^2}+t)\,Fy}{\sqrt{x^2+y^2}+\epsilon} & 1 \end{bmatrix}$$

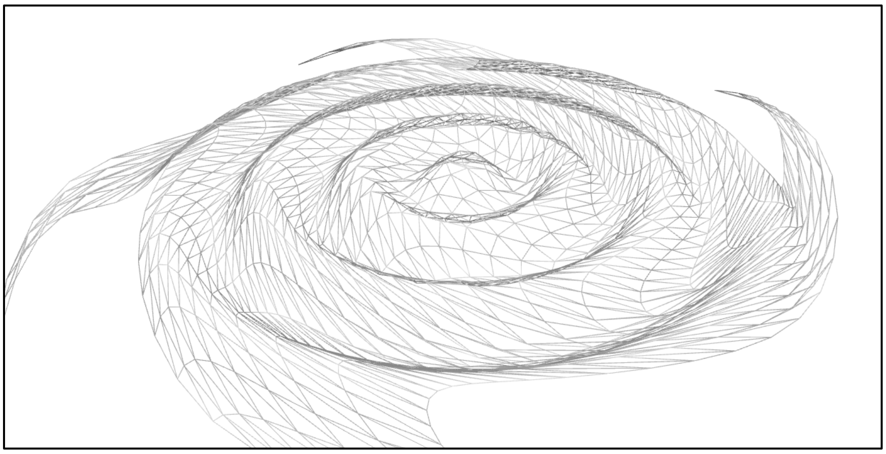
자코비안 행렬은 Deformation Function이 주어졌을 때 위 그림과 같이 구할 수 있다.

```
    // Jacobian of the wave deformation
    float eps = 0.0001; // 입실론, 입실론을 0으로 주면 문제가 생긴다. Normal vector가 문제가 생김.
    mat3  Jt;   //Jacobian matrix transpose. Note that Jt[column][row] in GLSL.

    Jt[0][0] = 1;   Jt[0][1] = 0;   Jt[0][2] = 0;
    Jt[1][0] = 0;   Jt[1][1] = 1;   Jt[1][2] = 0;
    Jt[2][0] = A*cos(F*l+phase)*F*VertexPosition.x/(l+eps);
    Jt[2][1] = A*cos(F*l+phase)*F*VertexPosition.y/(l+eps);
    Jt[2][2] = 1;

    // Inverse transpose of the Jacobian matrix (already transposed)
    vec3    newVertexNormal = inverse(Jt)*VertexNormal;
```

버텍스 쉐이더로 구현한 코드이다.

E1 - Compositing the wave and twisting deformers





[Position]

Wave deform을 진행하고 twist deform을 진행한다. wave deform을 할 때에는, z만 변하고, twist deform을 할 때에는 x, y만 변한다.

```
  // ----- Wave   // z값만 바뀌고 x,y는 그대로
  // Vertex position
  float l = length(VertexPosition.xy);
  float z = VertexPosition.z + A*sin(F*l + phase);
```

```
  // ---- Twist   // x,y만 바뀌고 z 값은 그대로
  // New position due to twisting
  // The twisting angle is proportional to the distance from the origin.
  float angle = twisting * length(VertexPosition.xy); // 2차원 벡터를 가져옴.
  float cosLength = cos(angle);
  float sinLength = sin(angle);
  float x = cosLength * VertexPosition.x - sinLength * VertexPosition.y;
  float y = sinLength * VertexPosition.x + cosLength * VertexPosition.y;
  vec4    newVertexPosition = vec4(x,y,z,1.0);
```

Practice 했던 것과 같이 x, y를 twist deform한 값으로 구하고 z를 wave deform한 값으로 구하여 새로운 버텍스 포지션을 계산했다.

[Normal]

Normal을 구하기 위해선 먼저 deformation function을 통해 자코비안 행렬을 구해야한다.

deformation function : $\boldsymbol{f(x,y,z) \to (x\cos\theta - y\sin\theta\,, x\sin\theta + y\cos\theta\,, z + A\sin(F(\sqrt{x^2+y^2}+t))}$이다.

(x, y는 twist 되었고, z는 wave 되었다)

자코비안 행렬 :

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \mathbf{f}_x}{\partial x} & \frac{\partial \mathbf{f}_x}{\partial y} & \frac{\partial \mathbf{f}_x}{\partial z} \\ \frac{\partial \mathbf{f}_y}{\partial x} & \frac{\partial \mathbf{f}_y}{\partial y} & \frac{\partial \mathbf{f}_y}{\partial z} \\ \frac{\partial \mathbf{f}_z}{\partial x} & \frac{\partial \mathbf{f}_z}{\partial y} & \frac{\partial \mathbf{f}_z}{\partial z} \end{bmatrix} = \begin{matrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ \dfrac{\boldsymbol{A\cos(F(\sqrt{x^2+y^2}+t)\,Fx}}{\boldsymbol{\sqrt{x^2+y^2}+\epsilon}} & \dfrac{\boldsymbol{A\cos(F(\sqrt{x^2+y^2}+t)\,Fy}}{\boldsymbol{\sqrt{x^2+y^2}+\epsilon}} & 1 \end{matrix}$$

```
    Jt[0][0] = cosLength;   Jt[0][1] = -sinLength;  Jt[0][2] = 0;
    Jt[1][0] = sinLength;   Jt[1][1] = cosLength;   Jt[1][2] = 0;
    Jt[2][0] = A*cos(F*l+phase)*F*VertexPosition.x/(l+eps);
    Jt[2][1] = A*cos(F*l+phase)*F*VertexPosition.y/(l+eps);
    Jt[2][2] = 1;

    // Inverse transpose of the Jacobian matrix (already transposed)
    vec3    newVertexNormal = inverse(Jt)*VertexNormal;
```

코드로 구현한 결과이다.