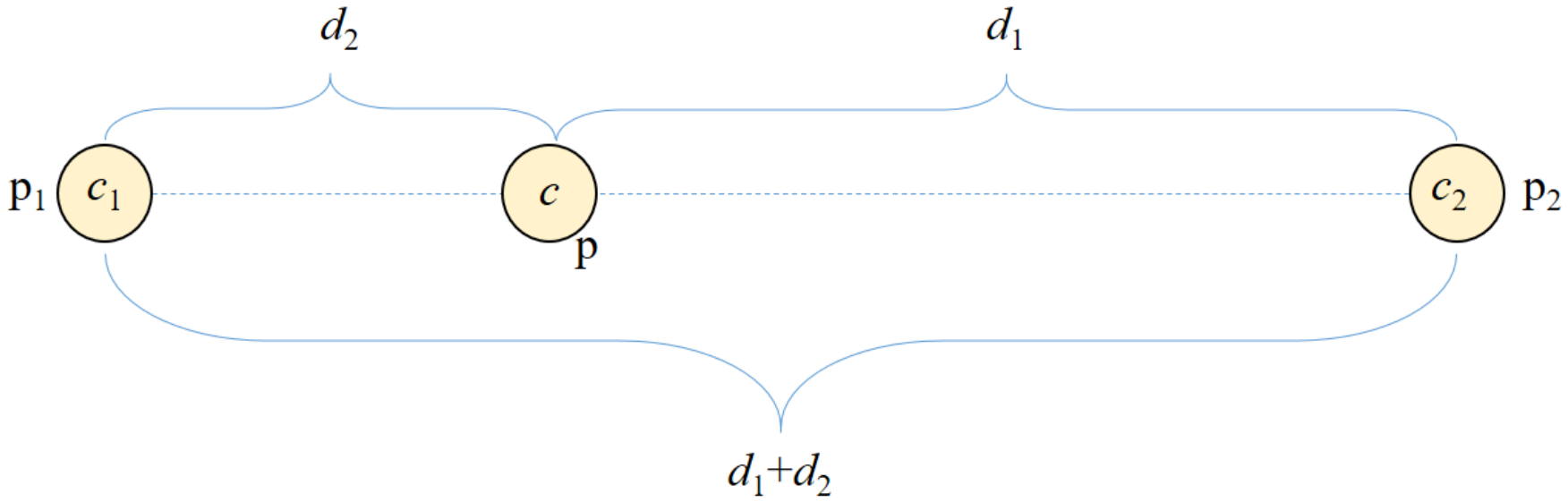
정점별 법선

동명대학교 게임공학과
강영민

보간(interpolation)

- 중간값 채우기

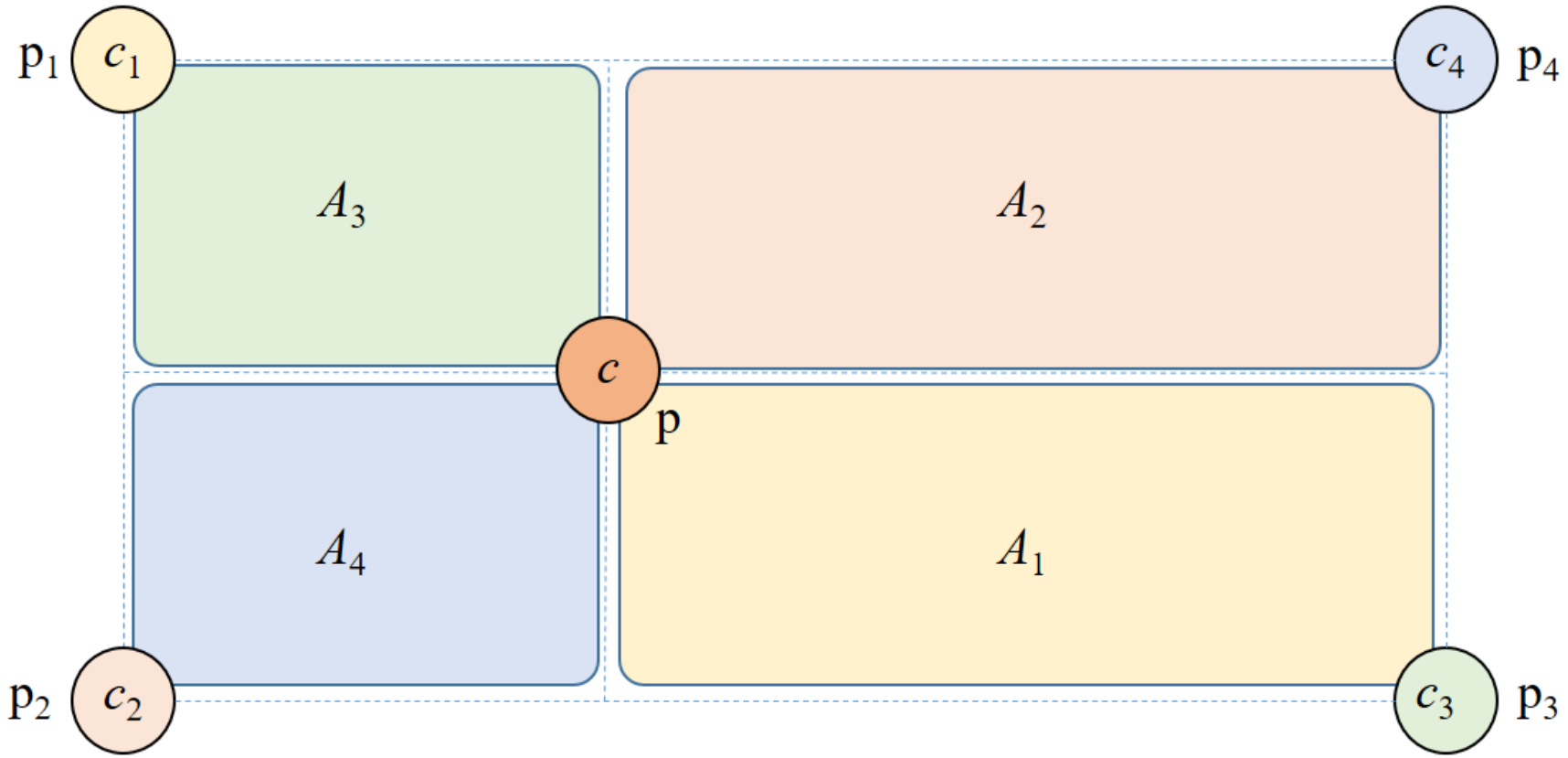
$$c = \frac{d_1 c_1 + d_2 c_2}{d_1 + d_2}$$



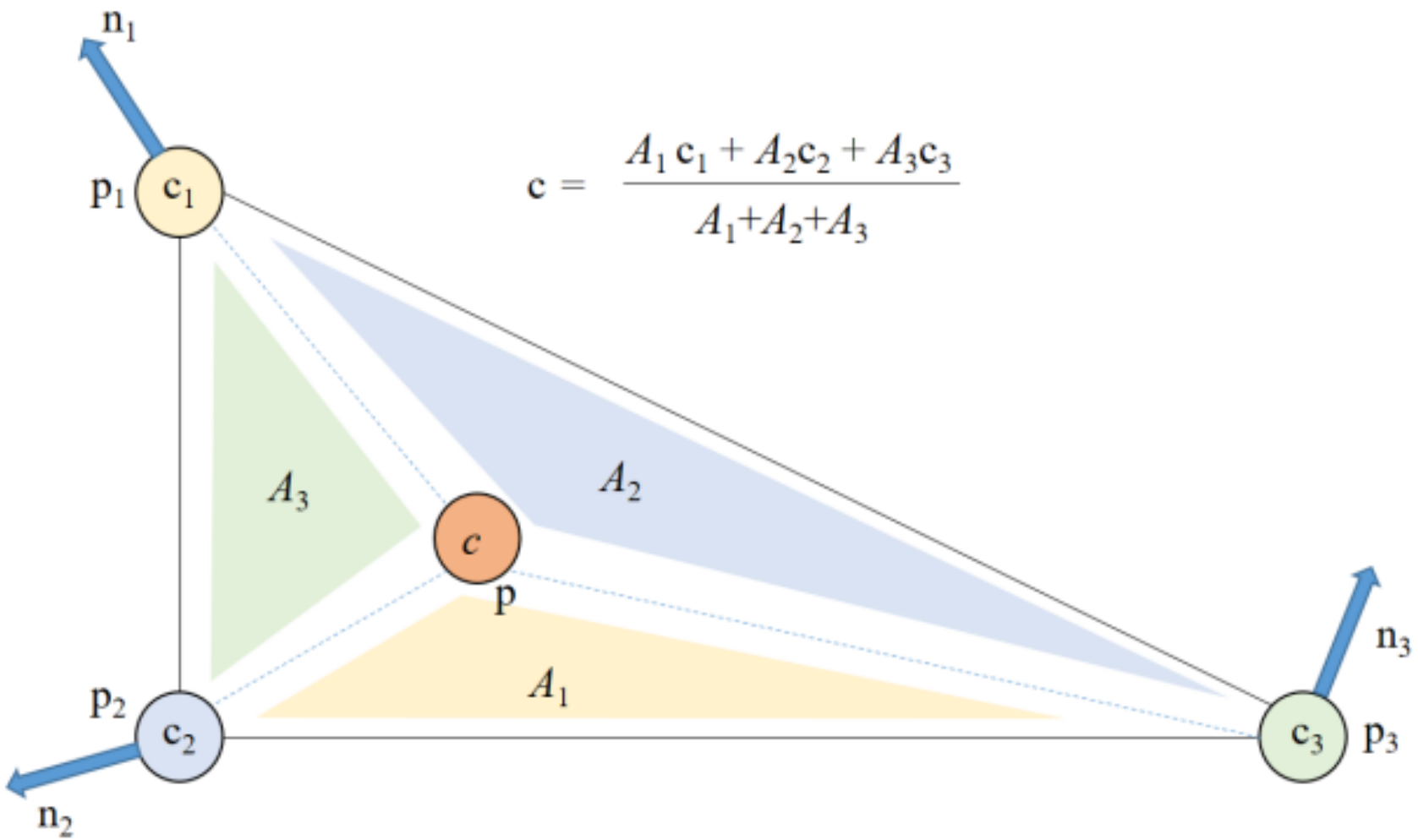
쌍선형 보간

- 2차원 보간

$$c = \frac{A_1 c_1 + A_2 c_2 + A_3 c_3 + A_4 c_4}{A_1 + A_2 + A_3 + A_4}$$

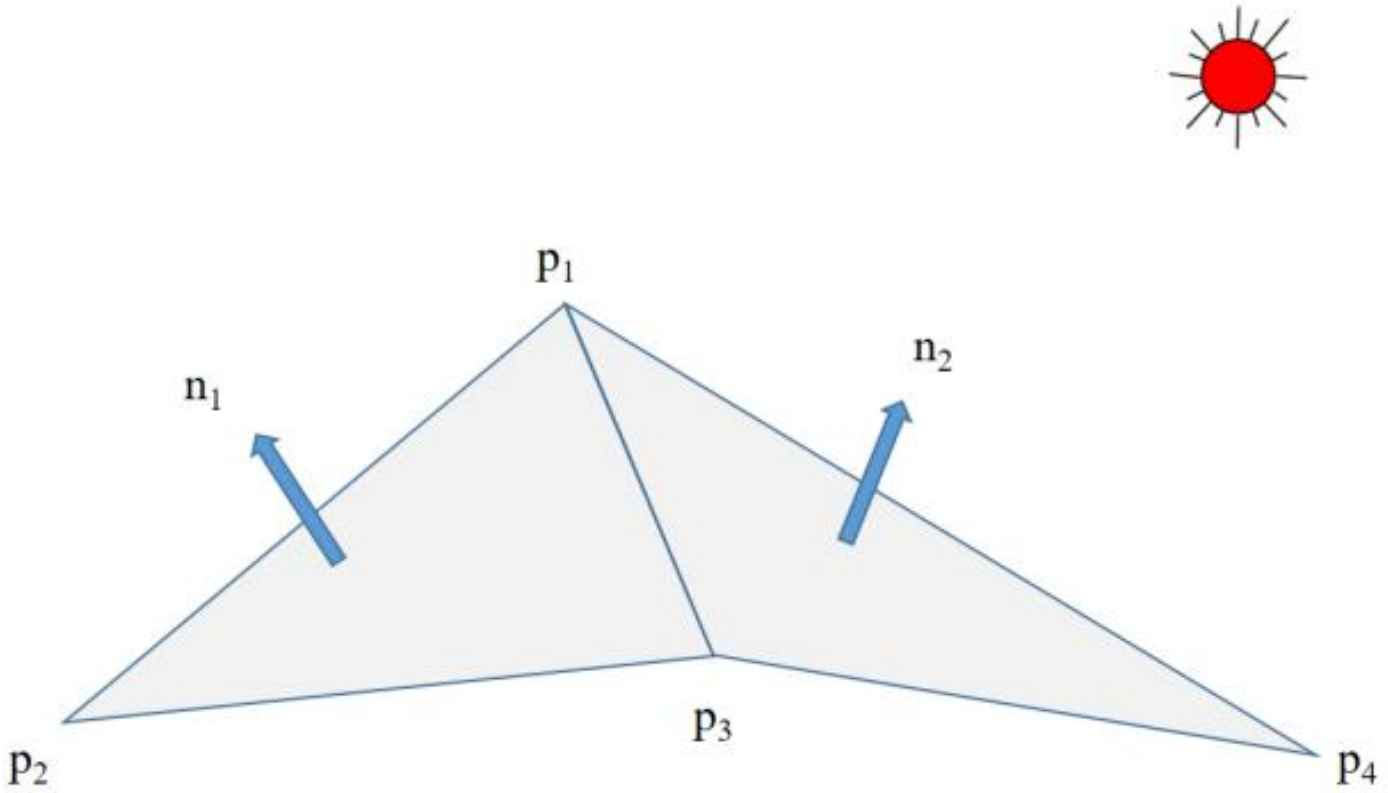


삼각형 내부의 보간

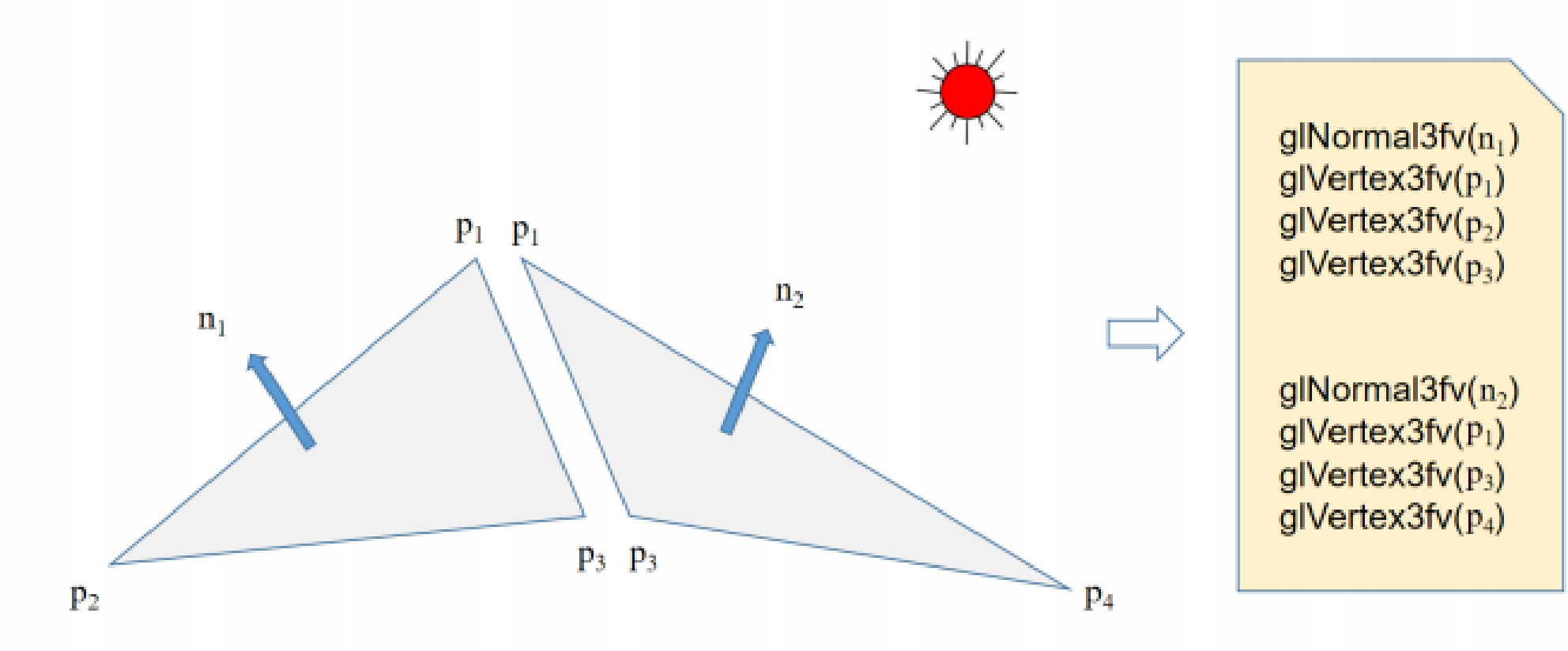


법선

- 퐁 모델을 이용한 색상 계산에 반드시 필요 – p1의 법선은?



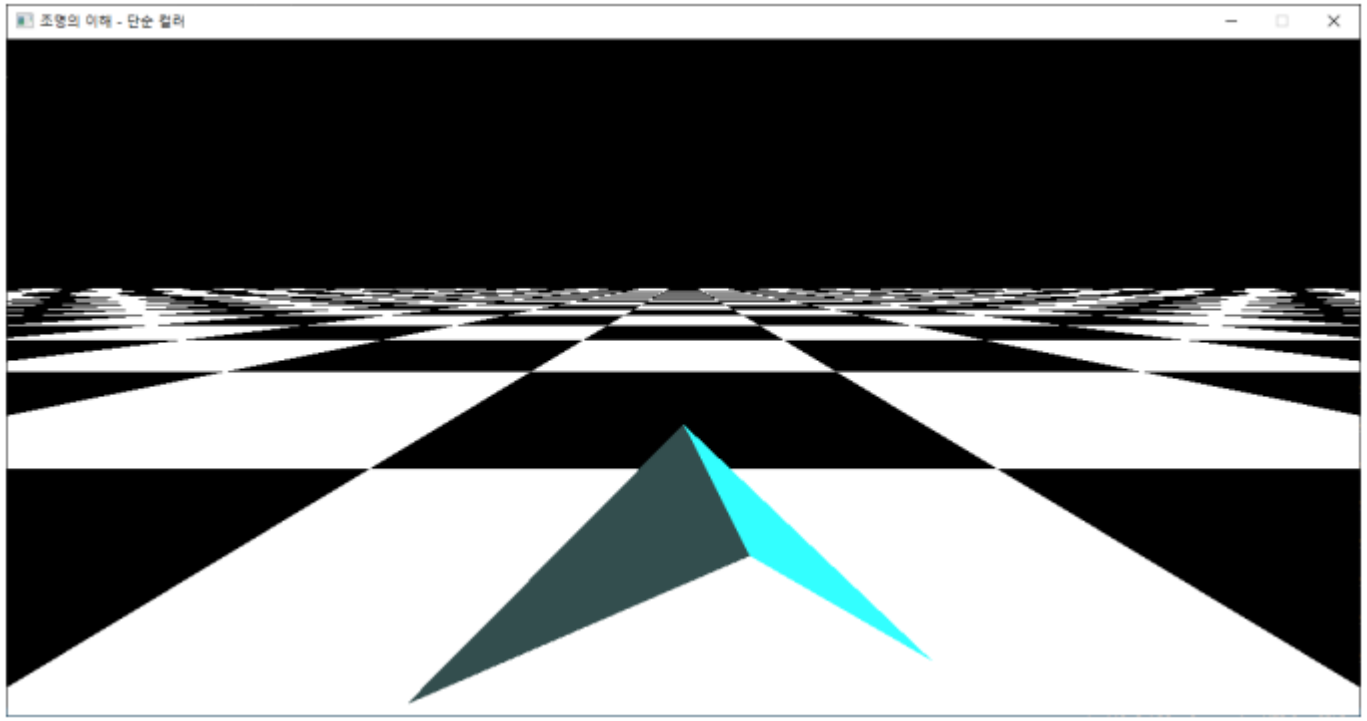
면별 법선 계산


p1 p1
n1
n2
p2
p3 p3
p4
glNormal3fv(n1)
glVertex3fv(p1)
glVertex3fv(p2)
glVertex3fv(p3)
glNormal3fv(n2)
glVertex3fv(p1)
glVertex3fv(p3)
glVertex3fv(p4)

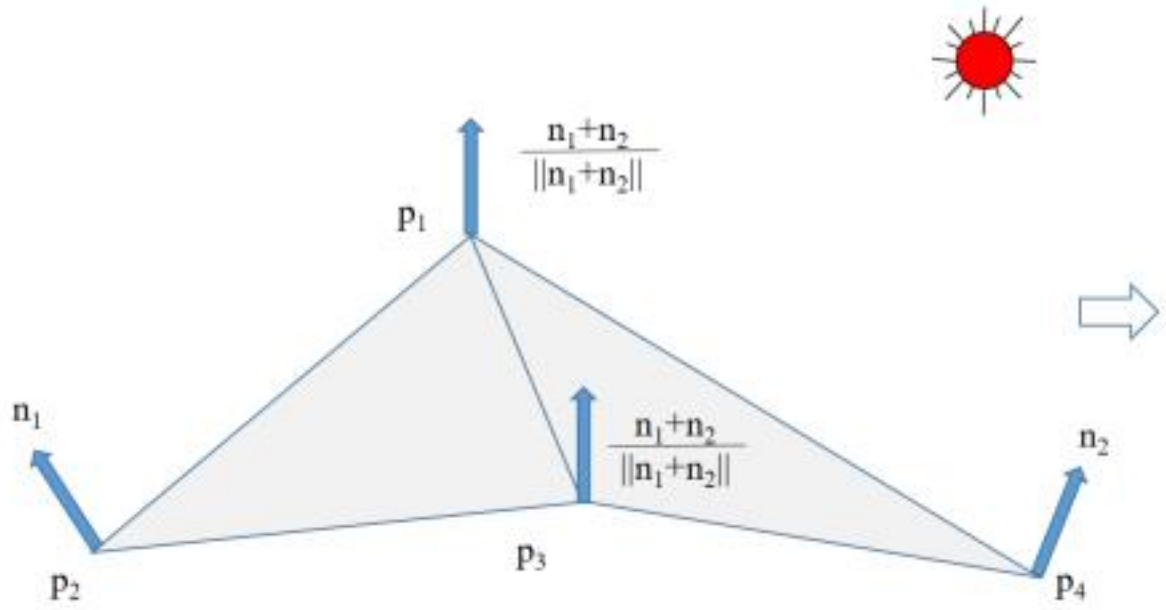
면별 법선 – 각진 모델

메시의 모양이 그대로 드러남 – Flat Shading

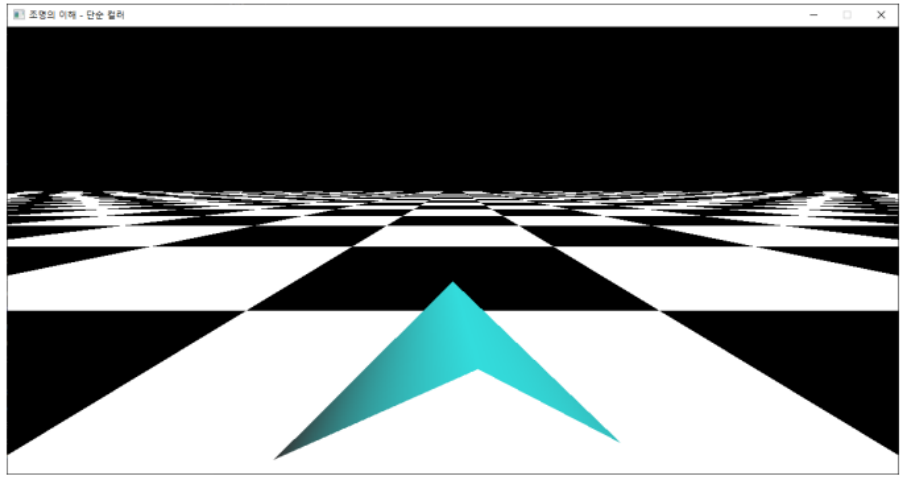
```
glEnable(GL_LIGHTING)
v = 1/math.sqrt(2)

glBegin(GL_TRIANGLES)
glNormal3f (-v, v, 0)
glVertex3f(0,1,0)
glVertex3f(-1,0,0)
glVertex3f(0,1,1)
glNormal3f (v, v, 0)
glVertex3f(0,1,0)
glVertex3f(0,1,1)
glVertex3f(1,0,0)
glEnd()
```



한 점의 법선 – 참여한 모든 면의 법선을 모으기




```
glNormal3fv( (n1+n2)/||n1+n2|| )
glVertex3fv(p1)
glNormal3fv(n1)
glVertex3fv(p2)
glNormal3fv( (n1+n2)/||n1+n2|| )
glVertex3fv(p3)

glNormal3fv( (n1+n2)/||n1+n2|| )
glVertex3fv(p1)
glNormal3fv( (n1+n2)/||n1+n2|| )
glVertex3fv(p3)
glNormal3fv(n1)
glVertex3fv(p4)
```



메시

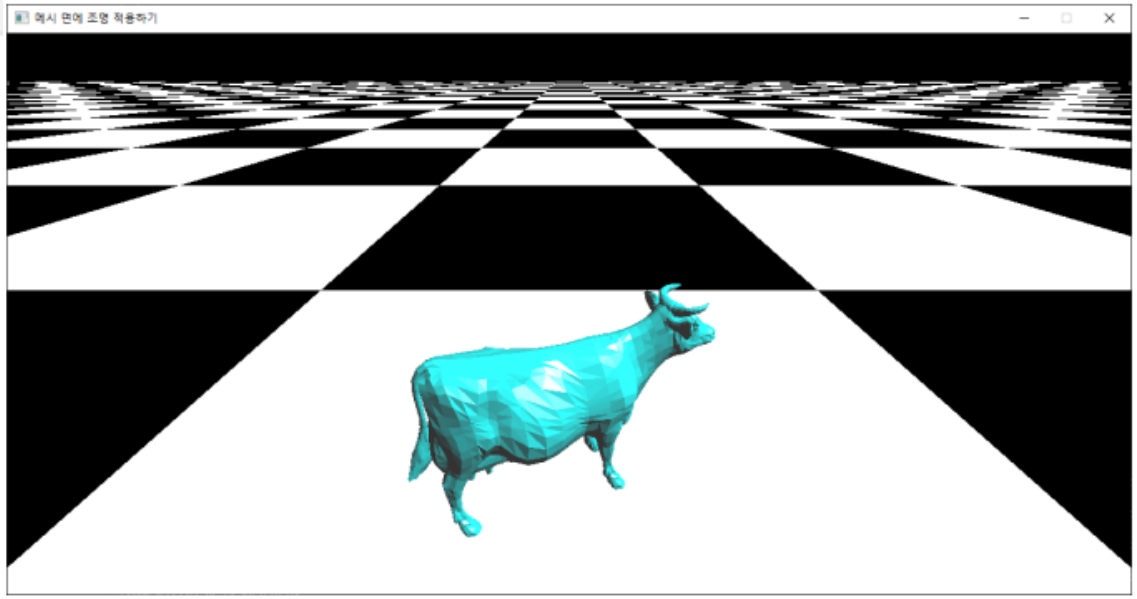
면별 법선의 계산

```
def loadData(self, filename):
    self.nF = int(next(inputfile))
    self.idxBuffer = np.zeros(shape=(self.nF*3, ), dtype=int)

    ### 법선벡터 저장을 위한 공간 준비
    self.normalBuffer = np.zeros(shape=(self.nF*3,), dtype=float )
```

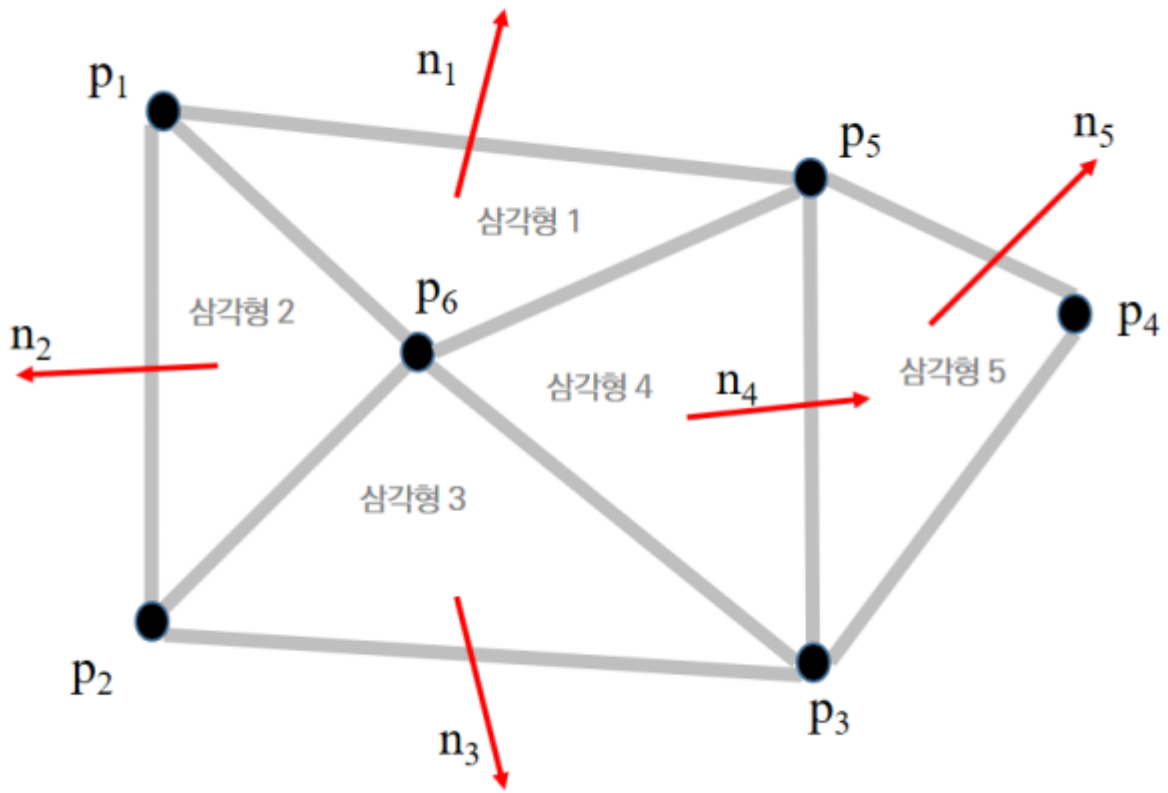
```
def loadData(self, filename):
    ...
    for i in range(self.nF):
        idx = next(inputfile).split()
        self.idxBuffer[i*3: i*3+3] = idx[1:4]
        index = self.idxBuffer[i*3: i*3+3]
        p0 = self.vertexBuffer[index[0]*3: index[0]*3 + 3]
        p1 = self.vertexBuffer[index[1]*3: index[1]*3 + 3]
        p2 = self.vertexBuffer[index[2]*3: index[2]*3 + 3]
        u = p1-p0
        v = p2-p0
        N = np.cross(u, v)
        norm = np.linalg.norm(N)
        N = N/norm
        self.normalBuffer[i*3: i*3+3] = N
```

면 그리기

```
def draw(self):

    glBegin(GL_TRIANGLES)
    for i in range(self.nF):
        verts = self.idxBuffer[i*3: i*3+3]
        glNormal3fv( self.normalBuffer[i*3: i*3+3] )
        glVertex3fv( self.vertexBuffer[verts[0]*3 : verts[0]*3 +3] )
        glVertex3fv( self.vertexBuffer[verts[1]*3 : verts[1]*3 +3] )
        glVertex3fv( self.vertexBuffer[verts[2]*3 : verts[2]*3 +3] )
    glEnd()
```



정점별 법선



정점별 법선을 저장할 공간

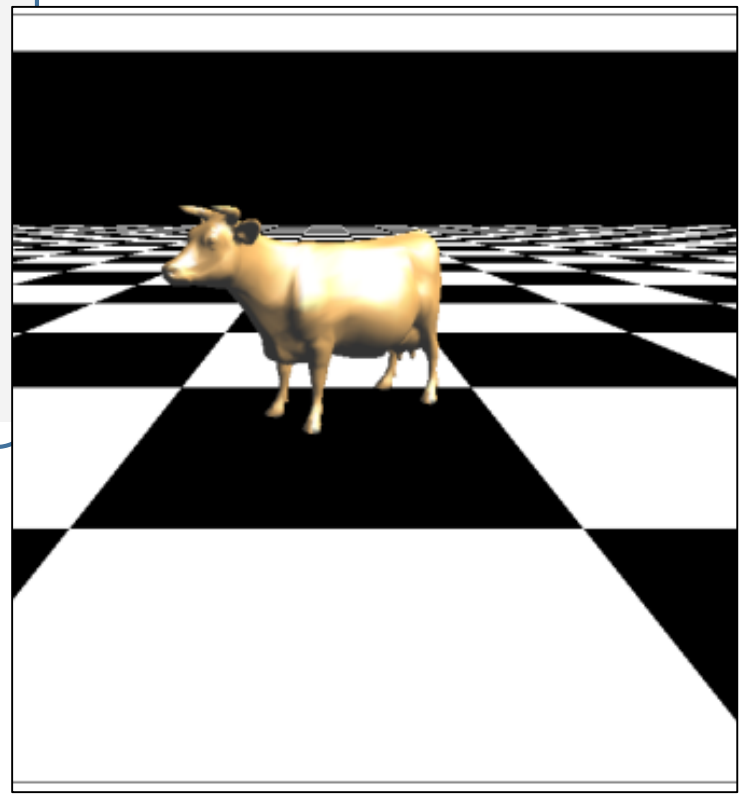
```
    def loadData(self, filename):
        with open(filename, 'rt') as inputfile:
            self.nV = int(next(inputfile))

            self.vertexBuffer = np.zeros(shape = (self.nV*3, ),
                                         dtype=float)
            ### 정점 별 법선벡터 저장을 위한 공간 준비
            self.normalBuffer = np.zeros(shape = (self.nV*3, ),
                                         dtype=float)
```

정점별 법선을 계산하기

```
    def loadData(self, filename):
        with open(filename, 'rt') as inputfile:
            self.nV = int(next(inputfile))

            ...

            for i in range(self.nF):
                idx = next(inputfile).split()
                self.idxBuffer[i*3: i*3+3] = idx[1:4]
                index = self.idxBuffer[i*3: i*3+3]
                p0 = self.vertexBuffer[index[0]*3: index[0]*3 + 3]
                p1 = self.vertexBuffer[index[1]*3: index[1]*3 + 3]
                p2 = self.vertexBuffer[index[2]*3: index[2]*3 + 3]
                u = p1-p0
                v = p2-p0
                N = np.cross(u, v)
                self.normalBuffer[index[0]*3: index[0]*3 + 3] += N
                self.normalBuffer[index[1]*3: index[1]*3 + 3] += N
                self.normalBuffer[index[2]*3: index[2]*3 + 3] += N

            for i in range(self.nV):
                N = self.normalBuffer[i*3: i*3 + 3]
                norm = np.linalg.norm(N)
                N = N/norm
                self.normalBuffer[i*3: i*3 + 3] = N
```

면 그리기 – 정점별 법선 설정

```
def draw(self):
    glBegin(GL_TRIANGLES)
    for i in range(self.nF):
        verts = self.idxBuffer[i*3: i*3+3]
        glNormal3fv( self.normalBuffer[verts[0]*3 : verts[0]*3 +3] )
        glVertex3fv( self.vertexBuffer[verts[0]*3 : verts[0]*3 +3] )
        glNormal3fv( self.normalBuffer[verts[1]*3 : verts[1]*3 +3] )
        glVertex3fv( self.vertexBuffer[verts[1]*3 : verts[1]*3 +3] )
        glNormal3fv( self.normalBuffer[verts[2]*3 : verts[2]*3 +3] )
        glVertex3fv( self.vertexBuffer[verts[2]*3 : verts[2]*3 +3] )
    glEnd()
```



광원