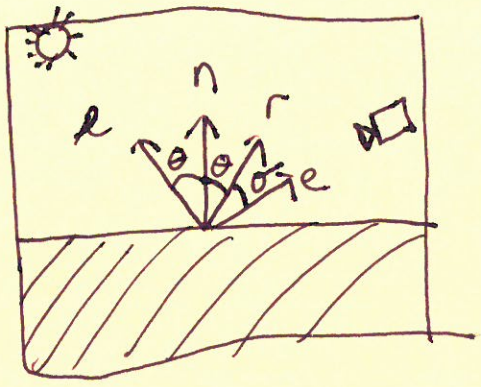
Phong Shading

March 1

How to model using math tools:

1) ~~matrix mult~~
2) dot products
3) ~~cross products~~



Let's try dot:

$$C = C_l * (e \cdot r)$$

C_l: color of light

⇐ Why will this not work?

1) it may be neg

Lets try clamping

$$C = C_l * \max(0, r \cdot e)$$

⇐ Shiny patch is too big

Lets try phong exponet

$$C = C_l * \left(\max(0, r \cdot e)\right)^p \quad \text{w/ } p \in \mathbb{R}^+$$

How to compute r?

$$r = -\ell + 2(\ell \cdot n)n$$

Combine w/ diffuse appearance

Recall $$c = c_r\left(c_a + c_\ell \max(0, n \cdot \ell)\right)$$

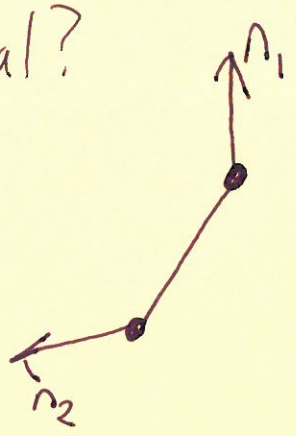
- c_r: diffuse reflectance
- c_a: ambient color
- c_ℓ: light color

Combine and we get

$$c = c_r\left(c_a + c_\ell \max(0, n \cdot \ell)\right) + c_\ell \left[\max(0, e \cdot r)\right]^p$$

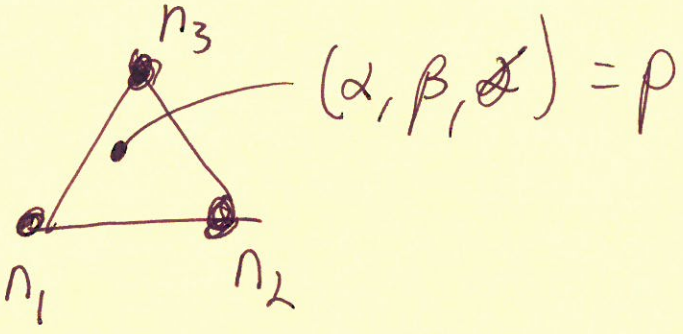
To implement, in fragment shader

how do I interpolate normal?



Interpolate normal

normal at $p = \alpha n_1 + \beta n_2 + \gamma n_3$



Polygons

Rep for Ding

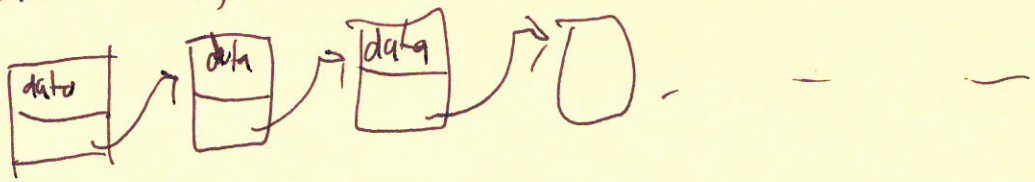
- ~~list~~ linked list

STL w/ vectors

```
# include <vector>
```

```
std::vector<float> coords;
coords.push_back(1);
coords.push_back(2);
      ⋮
coords[5];
```

w/ Linked lists



```
                                    head = coords.begin();
std::vector<float>::iterator        ~~begin~~
std::vector<float>::iterator        tail = coords.end();

                                          cur
for (std::vector<float> iterator ~~begin~~ = coords.begin();
     cur != coords.end(); cur++) {
    std::cout << *cur << std::endl;


}
```