Light It Up

(1) Define your lights
(2) Assign Normals
(3) Assign Materials

(1) Define : · Light position / direction
· Light color

```
GLfloat light0_pos[] = { 0, 2, 1, 0 };
GLfloat light1_pos[] = { 1, 0, -8, 0 };
```

^ both directions → explain a position

```
glLightfv(GL_LIGHT0, GL_POSITION, light0_pos);
glLightfv(GL_LIGHT1, GL_POSITION, light1_pos);
```

Light Color → 3 types of light
ambient → background light
diffuse → light on a sphere
speculer → shiny bits

```
GLfloat white_light[] = { 1, 1, 1, 1 };

glLightfv(GL_LIGHT0, GL_DIFFUSE, white_light)
   "     (   "     1      "         "      )
   "     (   "     0, GL_SPECULAR,  "      )
   "     (   "     1, GL_SPECULAR,  "      )
```

ambient, can be per light but BAD ... why?

```
GLfloat lmodel_ambient[] = { .2, .2, .2, 1 };
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, lmodel_ambient);

glEnable(GL_LIGHTING); glEnable(GL_LIGHT0);
glEnable(GL_LIGHT1);
```

(2) Assigning Normals

```
glNormal3f(n.x, n.y, n.z);   → like color, must be before
glVertex3f(v.x, v.y, v.z);

glEnable(GL_NORMALIZE);   → Why!!
```

(3) Assigning Materials

Each object has seperate ambient, diffuse, & specular materials. You specify these like you do colors.

```
GLfloat red[] = {1, 0, 0, 1};
GLfloat green[] = {0, 1, 0, 1};
GLfloat white[] = {1, 1, 1, 1};
GLfloat shiny[] = {20};

glMaterialfv(GL_FRONT,           GL_AMBIENT, red)
             GL_BACK,            GL_DIFFUSE, green)
             GL_FRONT_AND_BACK,  GL_SPECULAR, white)
                                      ↓
                                 GL_AMBIENT_AND_DIFFUSE
glMaterialfv(GL_FRONT_AND_BACK, GL_SHININESS, shiny);
```

→ Alternatively you can still use glColor3f

```
glEnable(GL_COLOR_MATERIAL);
glColorMaterial(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE);
glColor3f(1, 0, 0);
```

→ example code

453 2008
Nov 4
Lec 15
(3)

Two Data Structures

(1) OBJ Style

- → list of vertices (Points)
- → list of faces
- → list of tex coords (2D Points)

```
Face {
  vector<int> vertex_indices
  vector<int> tex_indices
  Point normal;
}
```

(2) Half-Edge

```
Half Edge {
  Vertex* vertex
  Face * face
  HalfEdge* next
  HalfEdge* pair
}

Vertex {
  Half-Edge * halfedge
  Point position
  Point normal
}

Face {
  HalfEdge* halfedge;
}
```

- Note that (1) is very close to MD2 file format. Also you won't be adding/removing vertices (strength of HE).

→ Example operation