- smooth shading (vertex normals)
- center the model
- translation with mouse
- QTimer

Vertex Normal Calculations

→ Assume you're working with Vertex List / Face List like struct.

```
vector <Point> vertices
vector <Face> faces
vector <Point> vnorms      → this is where we store vertex normals, 1 per vertex
vector <vector <int> > vfaces → this is where we store the indices
                                to the faces that touch each vertex
```

```
for each face i in faces
    for each vertex index j in faces[i]
        vfaces[j].push_back(i);
```

```
for each vertex i in vertices
{   for each face index j in vfaces[i]
    {       ave += faces[j].normal;
            count += 1
    }
    vnorms[i] = ave / (float) count;
}
```

Centering the Model

→ You have to scale and translate the model

to some stadard size and position. You cannot assume that all MD2 models will play nice!

How?

Pos → ~~easy~~ ... find average (or median or whatever) for x, y, z. Then use this to translate model to where gluLookAt is pointing.

Scale → find x range, y range, z range. Use whichever is largest to provide scale for entire model. Then call glScalef(—) to place your model nicely within your projection.

How → go through all coordinates searching for x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max}.

Making a Nice UI

→ already you have slick rotating with mouse. Why not do translations with the mouse too?!?

Idea → If you click & hold the right button, you move along a plane parallel to the projection's near plane.

→ With default this is easy mouse.x changes cause translation of x coordinate, mouse y changes cause translation of y coordinate (inverted).

→ for general case we need an orthonormal basis. From what? The view direction

gluUnProject($\frac{w}{2}$, $\frac{h}{2}$, 0, mv, proj, vp, &ox, &oy, &oz)
gluUnProject($\frac{w}{2}$, $\frac{h}{2}$, 1, mv, proj, vp, &vdx, &vdy, &vdz)

viewdirection = (vdx, vdy, vdz) - (ox, oy, oz);
gluUnProject ($\frac{w}{2}$, $\frac{h}{2}$, 0, mv, proj, vp, &upx, &upy, &upz)

updirection = (upx, upy, upz) - (ox, oy, oz)

→ normalize & take cross product to get other axis.

Another way:
- mousePress // where mouse was first pressed
- mousePos // where mouse is

gluUnProject (mousePress.x, h - mousePress.y, 0, mv, proj, vp, &m1x, &m1y, &m1z);
gluUnProject (mousePos.x, h - mousePos.y, 0, mv, proj, vp, &m2x, &m2y, &m2z);

transvector = (m2x, m2y, m2z) - (m1x, m1y, m1z)

Timers → Idlefunc isn't the answer!
QT → QTimer

```
QTimer * timer;   // global

timer = new QTimer (this);
connect (timer, SIGNAL (timeout()), this, SLOT (yourSlotname));
timer -> start (100, false);
```

100 → in ms
false → continues indefinitely
true → single shot

- Remember to delete timer in your destructor!

GLUT → glutTimerFunc (100, yourhandlername, 0) → one shot!
100 → in ms

```
void yourhandlername (int time)
{
    glutPostRedisplay();
    glutTimerFunc (100, yourhandlername, time+1);
}
```