```
==============
Vertex Shader
==============
varying vec3 lightDir,normal;

void main() {
        lightDir = normalize(vec3(gl_LightSource[0].position));
        normal = gl_NormalMatrix * gl_Normal;

        gl_Position = ftransform();
}

================
Fragment Shader
================
varying vec3 lightDir,normal;

void main() {
        float intensity;
        vec4 color;

        // normalizing the lights position to be on the safe side
        vec3 n = normalize(normal);

        intensity = dot(lightDir,n);
        if (intensity > 0.95)
                color = vec4(1.0,0.5,0.5,1.0);
        else if (intensity > 0.5)
                color = vec4(0.6,0.3,0.3,1.0);
        else if (intensity > 0.25)
                color = vec4(0.4,0.2,0.2,1.0);
        else
                color = vec4(0.2,0.1,0.1,1.0);
        gl_FragColor = color;
}

================================================
Use this in your C program to load the shaders
================================================
void setShaders() {
        char *vs = NULL,*fs = NULL,*fs2 = NULL;

        v = glCreateShaderObjectARB(GL_VERTEX_SHADER_ARB);
        f = glCreateShaderObjectARB(GL_FRAGMENT_SHADER_ARB);
        f2 = glCreateShaderObjectARB(GL_FRAGMENT_SHADER_ARB);

        vs = textFileRead("toonf2.vert");
        fs = textFileRead("toonf2.frag");
        const char * vv = vs;
        const char * ff = fs;
        glShaderSourceARB(v, 1, &vv,NULL);
        glShaderSourceARB(f, 1, &ff,NULL);
        free(vs);free(fs);

        glCompileShaderARB(v);
        glCompileShaderARB(f);

        printInfoLog(v);
        printInfoLog(f);
        printInfoLog(f2);

        p = glCreateProgramObjectARB();
        glAttachObjectARB(p,v);
        glAttachObjectARB(p,f);

        glLinkProgramARB(p);
        printInfoLog(p);

        glUseProgramObjectARB(p);
        loc = glGetUniformLocationARB(p,"time");
}
```