Computação Gráfica e Interfaces - Projecto III

Tipos de dados usados para representar as luzes

Javascript: Existe um array *lights* em que são inseridas as luzes (com as suas propriedades manualmente definidas) a usar no programa.

Shaders: As luzes são só representadas no *fragment shader*. Existe um array de uniformes do tipo *LightInfo* (um *struct* definido dentro do shader para guardar todas as propriedades de cada luz).

Construção da interface e manipulação de parâmetros

Para ter um número de luzes maior que 3 é necessário apenas acrescentar mais uma entrada no array *lights*, pois o nosso programa passa o número efetivo de luzes (*length* de *lights*) como um uniforme, tendo também uma constante *MAX_LIGHTS* no fragment shader para tratar uma limitação do WebGL 1.0 especificada no enunciado.

Cada propriedade de cada luz é enviada para a respetiva propriedade no *uniform* do tipo *LightInfo* da respetiva luz no fragment shader o que possibilita a alteração das propriedades de uma dada luz na interface só se aplicarem à luz respetiva. O mesmo acontece com o material do *bunny* para o qual existe um *uniform* do tipo *MaterialInfo* que guarda as propriedades *Ka*, *Kd*, *Ks* e *shininess*. Os outros materiais têm propriedades pré-definidas.

Implementação da luz do tipo spotlight

```
if(uLights[i].cutoff < 0.0)
    attenuation = 1.0; // there is no attenuation
else {
    vec3 LL = normalize(-uLights[i].axis);
    if(dot(L, LL) < cos(radians(uLights[i].aperture))) // aperture in radians
    // Total attenuation outside the cone
        attenuation = 0.0;
    else
       // dot(L,LL) with both L and LL normalized will be equal to cos(alpha)
       // where alpha is the angle between the two vectors L and LL
       attenuation = pow(dot(L,LL), uLights[i].cutoff);
}
```

Se o *cutoff* for negativo não há atenuação. Caso haja *cutoff*, calcula-se o vetor unitário do simétrico da direção da luz (propriedade *axis*) e faz-se o produto escalar entre esse vetor e o vetor da luz já calculado anteriormente. Caso esse produto seja menor que o cosseno do ângulo de abertura da luz (propriedade *aperture*) a atenuação é total, o que acontece fora do "cone" da luz, caso contrário, calcula-se a atenuação com a fórmula dada pelos professores. No final, a atenuação multiplica pela luz difusa e especular apenas.

Implementação do desafio

O desafio foi implementado utilizando event listeners no canvas para mousedown, mouseup e mousemove, que ajudam a calcular os ângulos das rotações a fazer na *mView* em x e em y quando se clica e arrasta o rato.

Comentários

Na implementação do desafio, é de notar que o movimento e a orientação da câmara utilizando o rato funciona bem, mas não altera os valores de *eye*, *at* e *up* representados na interface dat.gui. Isto acontece porque faz-se as rotações diretamente sobre a *mView*. Como o enunciado não implica ter de se atualizar a interface, não foi dedicado tempo a este problema.

Nuno Nogueira 60599, Martim Costa 64901