Relatório

Computação Gráfica e Interfaces

Hugo Monteiro 56755

Nádia Mendes 53175

Para representar as luzes usadas no programa usamos no fragment shader uma struct a qual chamamos LightInfo esta continha as variáveis necessárias para a luz, nomeadamente três vetores do tipo vec3 ambiente, diffuse e specular, um vector do tipo vec4 o position e três variáveis booleanas active, spotlight e directional. Esta estrutura e depois carregada numa array a qual chamamos uLight.

```
struct LightInfo {
    vec3 ambient;
    vec3 diffuse;
    vec3 specular;

    vec4 position;

    bool active;
    bool spotlight;
    bool directional;
};
```

```
uniform LightInfo uLight[MAX_LIGHTS];
```

No vertex shader necessitamos apenas a variável lightPosition, sendo esta um vec4.

```
uniform vec4 lightPosition;
```

Já parte do JavaScript usamos as seguintes variáveis:

```
let defaultLight = {
    // default values
    ambient: vec3(75, 75, 75),
    diffuse:  vec3(175, 175, 175),
    specular: vec3(255, 255, 255),
    position: vec4(-5,-5,-5,0),
    axis: vec3(1,1,1),
    aperture: 10,
    cuttoff: 10,
    active: true,
    spotlight: false,
    directional: false
}
```

De seguida igualamos as variáveis contidas no defaultLight a uma nova estrutura denominada light

```
let light = { ...defaultLight };
```

E por fim relacionamos estas com as variáveis contidas no fragment shader para que os valores fossem passados para o shader.

```
function Light(i){
    pushMatrix();
        gl.uniform1i(gl.getUniformLocation(program, "uLight[" + i + "].active"), lights[i].active)
        gl.uniform1i(gl.getUniformLocation(program, "uLight[" + i + "].directional"), lights[i].directional);
        gl.uniform1i(gl.getUniformLocation(program, "uLight[" + i + "].spotlight"), lights[i].spotlight);
        gl.uniform3fv(gl.getUniformLocation(program, "uLight[" + i + "].ambient"), flatten(scale(1 / 255, lights[i].ambient)));
        gl.uniform3fv(gl.getUniformLocation(program, "uLight[" + i + "].diffuse"), flatten(scale(1 / 255, lights[i].diffuse)));
        gl.uniform3fv(gl.getUniformLocation(program, "uLight[" + i + "].specular"), flatten(scale(1 / 255, lights[i].specular)));
        gl.uniform4fv(gl.getUniformLocation(program, "uLight[" + i + "].position"), flatten(lights[i].position));

        gl.uniform4fv(gl.getUniformLocation(program, "lightPosition"), flatten(lights[i].position));

        console.log(lights[i].position);

        gl.uniform1i(gl.getUniformLocation(program, "uNLights"), lights.length);

        multTranslation([lights[i].position]);
        multScale([0.125, 0.125, 0.125]);
        uploadModelView();
        SPHERE.draw(gl, program, mode);
    popMatrix();
}
```

Ou seja carregamos as luzes que são cada uma como referido assim uma struct, num array de seguida na chamada ao shader é fornecido o conteúdo desse array.

Para que no programa em vez de utilizarmos 3 luzes, se pudesse implementar um número maior teríamos apenas de ativar um maior número de luzes contidas no array sendo que este no nosso projeto suporta até 8 luzes.

Array usado no fragmente shader, e variável de limitação do numero máximo de luzes suportadas.

```
uniform LightInfo uLight[MAX_LIGHTS];
```

```
const int MAX_LIGHTS = 8;
```

No JavaScript usamos o segunte array

```
let lights = [];
```

Lamentamos, mas não conseguimos implementar a luz do tipo spotlight por falta de tempo, uma vez que estamos no fim do semestre e tivemos uma serie de avaliações e outros projetos a entregar em simultâneo.