Tipos de Dados:

APP.JS

- **MAX_LIGHTS** – Numero Maximo de luzes – Agora definido a 3 (definido aqui para nao conseguirmos criar mais do que o numero maximo de luzes)
- **Lights[]** – Vetor com todas as lights e os seus atributos (active – se esta ativa, ambient – luz ambiente, diffuse – difusao da cor depende do material, specular – reflexao especular, position – posiçao da luz(w indica se a luz é direcional), axis – direçao da luz , aperture - abertura, cutoff – atenuador da intensidade da luz)

FRAGMENT SHADER

- **MAX_LIGHTS** – Numero Maximo de luzes – Agora definido a 3 (usado porque “i o i não pode ser comparado com uma expressão não constante num for loop)
- **nLights** – Numero de luzes que efetivamente existem
- **Uniforms** – variaveis para calcular como a luz reflete nos materiais (material ambient, material diffusion, material speculation, shininess), variaveis para as luzes (mode, active, ambient, diffuse, specular, position, axis, aperture, cutoff).

Ter mais do que 3 luzes:

Teríamos de alterar o MAX_LIGHTS para um número maior, tanto no fragment shader como na app.js, o que poderá causar problemas de incompatibilidade. No entanto, por não poder passar uma constante para o fragment shader, nesta versão do webgl, tivemos de fazer assim.

Implementação de Spotlight:

```
//for spotLight
if(lights[i].spotlight && lights[i].position[3] == 1.0){

    float lightIntensity = 0.0;

    float angle = abs(acos(dot(normalize(-1.0 *  lights[i].axis), fLight)));

    if(angle <= (lights[i].aperture)){

        lightIntensity = 1.0 + pow(abs(cos(angle)), lights[i].cutoff);

    }


gl_FragColor.xyz += (ambientColor + diffuse + specular) * lightIntensity;


}else{
```

A implementação do spotlight foi questão de fazer como o professor disse, vendo o angulo entre o vetor da luz ao ponto e do Axis e descontando de acordo com o cutoff (se sequer fosse preciso ter luz)

Desafio:

Para o desafio, nós fizemos um eventListener que vai alterando duas variaveis locais (mouseX, mouseY). Depois calculamos a nova posição da camera.eye, camera.at e camera.up. De seguida, se de facto o rato for pressionado alteramos o lookAt para a nova mView (view matrix), e fazemos load dessa mesma matrix. Esta nova viewMatrix tem a camera. Eye alterada para conseguir fazer rotações tendo em conta o eixo X e Y, a camera.at é vec3(0,0,0) porque queremos olhar sempre para o meio do chão (ground) e camera.up é vec3(0,1,0) porque queremos considerar o y, a “up

```
function renderCamera(){

    mProjection = perspective(camera.fovy, aspect, camera.near, camera.far);
    gl.uniformMatrix4fv(gl.getUniformLocation(program, "mProjection"), false, flatten(mProjection));


    // In your rendering loop, check if the mouse is being clicked and update the camera's position and orientation
    if (isMouseDown) {

        let cameraRadius = 5;

        // Use the mouse movement to update the camera's orientation
        let rotationX = (camera.eye[0] - mouseY) / (canvas.height / 2);
        let rotationY = (camera.eye[1] - mouseX) / (canvas.width / 2);

        let absRotationX = Math.abs(rotationX);

        // Calculate the scale factor for the rotation around the y-axis
        let yScaleFactor = 1 + absRotationX / 90;

        // Scale the rotation around the y-axis by the calculated factor
        rotationY *= yScaleFactor;

        let cameraPosition = vec3(
            camera.eye[0] - cameraRadius * Math.cos(rotationY * 2),
            camera.eye[1] - cameraRadius * Math.sin(rotationX*2),
            camera.eye[2] - cameraRadius * Math.sin(rotationY*2)
        );
```

```
let mouseX = 0;
let mouseY = 0;
let isMouseDown = false;

// Set up event listeners to track mouse movement
canvas.addEventListener("mousedown", function(event) {
  isMouseDown = true;
});

canvas.addEventListener("mousemove", function(event) {
  // Update the mouse position
  mouseX = event.clientX;
  mouseY = event.clientY;
});

canvas.addEventListener("mouseup", function(event) {
  isMouseDown = false;
});
```

direction” do nosso referencial.

```
        let mView = lookAt(cameraPosition, vec3(0, 0, 0), vec3(0,1,0));
        gl.uniformMatrix4fv(gl.getUniformLocation(program, "mView"), false, flatten(mView));
        gl.uniformMatrix4fv(gl.getUniformLocation(program, "mViewNormals"), false, flatten(normalMatrix(mView)));

        loadMatrix(mView);
    }
    else{
        let mView = lookAt(camera.eye, camera.at, camera.up);
        gl.uniformMatrix4fv(gl.getUniformLocation(program, "mView"), false, flatten(mView));
        gl.uniformMatrix4fv(gl.getUniformLocation(program, "mViewNormals"), false, flatten(normalMatrix(mView)));

        loadMatrix(mView);}
```

Projeto 3

G01-03

Ricardo Gonçalo(60519) & Rita Barbosa(64925)