Tarea 2.1: Movimiento de la cámara

Alumno: Jeanne LE GALL-BOTTE

El objetivo de este programa es representar un cilindro tridimensional mediante WebGL y permitir al usuario explorar la escena moviendo la cámara alrededor del objeto. Para ello, se implementa un sistema de control basado en botones en pantalla y teclas del teclado, que modifican los parámetros de la cámara en coordenadas esféricas.

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Examen - Gráficas por Computadora</title>
    <style>
        #controls {
            position: absolute;
            top: 10px;
            left: 10px;
            display: inline-block;
        }
        #controls div {
            display: flex;
            justify-content: center;
        }
        .smallButton {
            width: 50px;
            height: 50px;
            margin: 3px;
            font-size: 18px;
            cursor: pointer;
            background-color: rgba(255, 255, 255, 0.5);
            border: none;
            border-radius: 10px;
        }
        button:hover {
            background-color: rgba(255, 255, 255, 0.8);
```

```
        .reset {
            width: 156px;
            height: 50px;
            margin: 3px;
            font-size: 18px;
            cursor: pointer;
            background-color: rgba(255, 255, 255, 0.5);
            border: none;
            border-radius: 10px;
        }
    </style>
</head>
<body>
    <canvas id="canvas" width="800" height="800"></canvas>
    <div id="controls">
        <div>
            <button class="smallButton" onclick="zoomIn()">+</button>
            <button class="smallButton" onclick="up()">↑</button>
            <button class="smallButton" onclick="zoomOut()">-</button>
        </div>
        <div>
            <button class="smallButton" onclick="left()">←</button>
            <button class="smallButton" onclick="down()">↓</button>
            <button class="smallButton" onclick="right()">→</button>
        </div>
```

El programa comienza creando un canvas de 800 por 800 píxeles donde se renderiza la escena en WebGL. En el lado superior izquierdo del canva hay botones interactivos que permiten controlar la cámara: los símbolos de flechas sirven para moverse hacia arriba, abajo, izquierda y derecha, los símbolos “+” y “-” permiten el zoom de la cámara, y finalmente un botón con la etiqueta “Reset” devuelve la cámara a su estado original. Además, estos mismos controles pueden activarse desde el teclado utilizando las teclas de flechas y los símbolos + y -.

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/gl-matrix/2.4.0/gl-matrix.js"></script>
<script>
    var canvas = document.getElementById('canvas');
    var gl = canvas.getContext('webgl');

    var radius = 20;
    var theta = Math.PI / 2;
    var phi = 0.0;

    function up() {
        phi += 0.1;
        if (phi > Math.PI / 2) phi = Math.PI / 2;
    }

    function down() {
        phi -= 0.1;
        if (phi < -Math.PI / 2) phi = -Math.PI / 2;
    }

    function left() {
        theta -= 0.1;
        if (theta < 0) theta += 2 * Math.PI;
    }

    function right() {
        theta += 0.1;
        if (theta > 2 * Math.PI) theta -= 2 * Math.PI;
    }

    function zoomIn() {
        radius -= 0.5;
    }

    function zoomOut() {
        radius += 0.5;
    }
```

El movimiento de la cámara está basado en coordenadas esféricas. Se definen tres parámetros principales: radius, que representa la distancia al objeto y controla el zoom, theta, que marca el ángulo de rotación horizontal alrededor del eje Y y phi, que corresponde a la elevación vertical de la cámara.

Las funciones *up()* y *down()* permiten incrementar o disminuir el ángulo phi. Este parámetro está limitado entre -π/2 y π/2 para evitar que la cámara se invierta por completo en el eje vertical. De forma similar, las funciones *left()* y *right()* modifican el ángulo theta para girar la vista alrededor del objeto. Los valores de theta se mantienen siempre en el rango entre 0 y 2π, garantizando un movimiento continuo y cíclico.

El acercamiento y alejamiento de la cámara se realizan con las funciones *zoomIn()* y *zoomOut(),* que ajustan el valor del radio en incrementos o decrementos de 0.5 unidades. Finalmente, la función *reset()* devuelve los tres parámetros radius, theta y phi a sus valores iniciales, de manera que el usuario pueda recuperar en cualquier momento la vista original.

Una vez definido el entorno gráfico, el programa genera un cilindro mediante la función *Cylinder()*.

```
// Dibujar escena
var timeOld = 0;
var animate = function (time) {
    var dt = time - timeOld;
    timeOld = time;

    var projMatrix = mat4.create();
    mat4.perspective(projMatrix, 40 * Math.PI / 180, canvas.width / canvas.height, 1, 100);
    gl.uniformMatrix4fv(Pmatrix, false, projMatrix);

    var eye = [
        radius * Math.cos(theta) * Math.cos(phi),
        radius * Math.sin(phi),
        radius * Math.sin(theta) * Math.cos(phi)
    ];

    var viewMatrix = mat4.create();
    mat4.lookAt(viewMatrix, eye, [0, 0, 0], [0, 1, 0]);

    var modelViewMatrix = mat4.clone(viewMatrix);
    mat4.translate(modelViewMatrix, modelViewMatrix, [
        4 * Math.cos(0.001*time),
        4 * Math.sin(0.001*time),
        0
    ]);
    mat4.rotateZ(modelViewMatrix, modelViewMatrix, 0.001*time + Math.PI / 2);
    mat4.rotateY(modelViewMatrix, modelViewMatrix, 4*0.001*time);

    gl.uniformMatrix4fv(MVmatrix, false, modelViewMatrix);

    gl.enable(gl.DEPTH_TEST);
    gl.depthFunc(gl.LEQUAL);
    gl.clearColor(0.5, 0.5, 0.5, 0.9);
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
```

El programa utiliza principalmente dos matrices. La primera es la matriz de proyección *Pmatrix* generada con mat4.perspective. La segunda es la matriz modelo-vista *MVmatrix*, que combina la vista de la cámara y las transformaciones aplicadas al objeto. La vista se obtiene con *mat4.lookAt*, que recibe la posición de la cámara (eye), el punto al que debe mirar (el centro de la escena [0,0,0]) y un vector vertical de referencia ([0,1,0]).

Sobre esta matriz se aplican transformaciones al modelo. La función *mat4.translate* desplaza el cilindro en el plano XY siguiendo una trayectoria circular dependiente del tiempo, de manera que el objeto parece moverse en órbita. A continuación, se utilizan dos rotaciones *mat4.rotateZ* y *mat4.rotateY* para hacer girar el cilindro sobre sí mismo en torno a los ejes Z e Y. El efecto combinado de estas transformaciones es que el objeto adquiere un movimiento constante, mientras la cámara puede seguir orbitando alrededor de él.