**计算机图形学**

**实验报告**

课程名称： 计算机图形学实验报告

专 业： 计算机科学与技术 班级： 2018-5

姓 名： 刘澳

学 号： 201801011312

**山 东 科 技 大 学**

**2021年 6月 27日**

# 实验八 立方体纹理映射

1. 实验要求
2. 建立三维坐标系0xyz，原点位于屏幕客户区中心，x轴水平向右为正，y轴铅直向上为正，z轴垂直于屏幕指向观察者。
3. 设置屏幕背景色为黑色。
4. 读入6张构成天空盒的位图作为纹理映射到立方体的可见表面上。
5. 按下鼠标左键缩小立方体，按下鼠标右键增大立方体。
6. 使用键盘方向键旋转纹理立方体。
7. 使用动画按钮，播放或停止立方体动画。
8. 关键程序代码

function **drawScene**(gl, programInfo, buffers, texture, deltaTime) {

  gl.**clearColor**(0.0, 0.0, 0.0, 1.0);  *// Clear to black, fully opaque*

  gl.**clearDepth**(1.0);                 *// Clear everything*

  gl.**enable**(gl.DEPTH\_TEST);           *// Enable depth testing*

  gl.**depthFunc**(gl.LEQUAL);            *// Near things obscure far things*

  gl.**clear**(gl.COLOR\_BUFFER\_BIT | gl.DEPTH\_BUFFER\_BIT);

  const fieldOfView = 45 \* Math.PI / 180;   *// in radians*

  const aspect = gl.canvas.clientWidth / gl.canvas.clientHeight;

  const zNear = 0.1;

  const zFar = 100.0;

  const projectionMatrix = mat4.**create**();

  mat4.**perspective**(projectionMatrix,

                   fieldOfView,

                   aspect,

                   zNear,

                   zFar);

  const modelViewMatrix = mat4.**create**();

  mat4.**translate**(modelViewMatrix,     *// destination matrix*

                 modelViewMatrix,     *// matrix to translate*

                 [-0.0, 0.0, -6.0]);  *// amount to translate*

  mat4.**rotate**(modelViewMatrix,  *// destination matrix*

              modelViewMatrix,  *// matrix to rotate*

              cubeRotation,     *// amount to rotate in radians*

              [0, 0, 1]);       *// axis to rotate around (Z)*

  mat4.**rotate**(modelViewMatrix,  *// destination matrix*

              modelViewMatrix,  *// matrix to rotate*

              cubeRotation \* .7,*// amount to rotate in radians*

              [0, 1, 0]);       *// axis to rotate around (X)*

  const normalMatrix = mat4.**create**();

  mat4.**invert**(normalMatrix, modelViewMatrix);

  mat4.**transpose**(normalMatrix, normalMatrix);

  {

    const numComponents = 3;

    const type = gl.FLOAT;

    const normalize = false;

    const stride = 0;

    const offset = 0;

    gl.**bindBuffer**(gl.ARRAY\_BUFFER, buffers.position);

    gl.**vertexAttribPointer**(

        programInfo.attribLocations.vertexPosition,

        numComponents,

        type,

        normalize,

        stride,

        offset);

    gl.**enableVertexAttribArray**(

        programInfo.attribLocations.vertexPosition);

  }

  {

    const numComponents = 2;

    const type = gl.FLOAT;

    const normalize = false;

    const stride = 0;

    const offset = 0;

    gl.**bindBuffer**(gl.ARRAY\_BUFFER, buffers.textureCoord);

    gl.**vertexAttribPointer**(

        programInfo.attribLocations.textureCoord,

        numComponents,

        type,

        normalize,

        stride,

        offset);

    gl.**enableVertexAttribArray**(

        programInfo.attribLocations.textureCoord);

  }

  {

    const numComponents = 3;

    const type = gl.FLOAT;

    const normalize = false;

    const stride = 0;

    const offset = 0;

    gl.**bindBuffer**(gl.ARRAY\_BUFFER, buffers.normal);

    gl.**vertexAttribPointer**(

        programInfo.attribLocations.vertexNormal,

        numComponents,

        type,

        normalize,

        stride,

        offset);

    gl.**enableVertexAttribArray**(

        programInfo.attribLocations.vertexNormal);

  }

  gl.**bindBuffer**(gl.ELEMENT\_ARRAY\_BUFFER, buffers.indices);

  gl.**useProgram**(programInfo.program);

  gl.**uniformMatrix4fv**(

      programInfo.uniformLocations.projectionMatrix,

      false,

      projectionMatrix);

  gl.**uniformMatrix4fv**(

      programInfo.uniformLocations.modelViewMatrix,

      false,

      modelViewMatrix);

  gl.**uniformMatrix4fv**(

      programInfo.uniformLocations.normalMatrix,

      false,

      normalMatrix);

  gl.**activeTexture**(gl.TEXTURE0);

  gl.**bindTexture**(gl.TEXTURE\_2D, texture);

  gl.**uniform1i**(programInfo.uniformLocations.uSampler, 0);

  {

    const vertexCount = 36;

    const type = gl.UNSIGNED\_SHORT;

    const offset = 0;

    gl.**drawElements**(gl.TRIANGLES, vertexCount, type, offset);

  }

  cubeRotation += deltaTime;

}

1. 程序运行结果、界面等



1. 思考与总结等

此次实验同上个实验类似，也运用的是WebGL库。 并且在上个实验的基础上，使用了WebGL库的贴图函数



此函数可以方便快捷的将图片贴至物体表面。