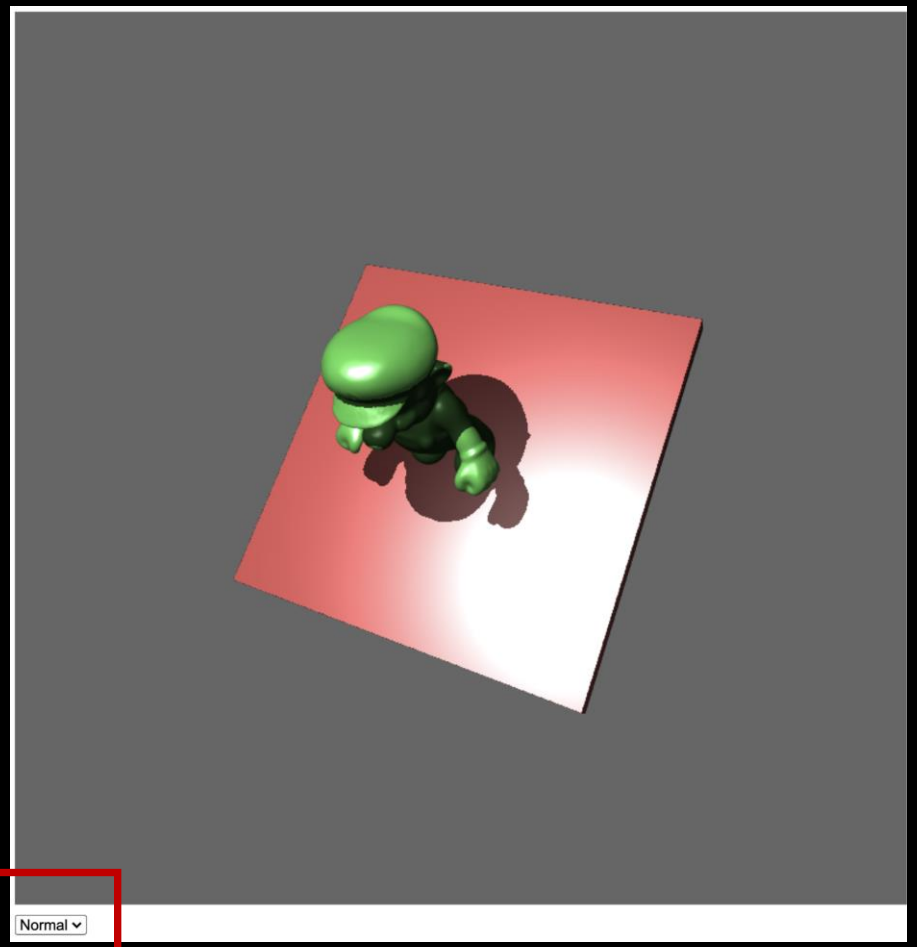
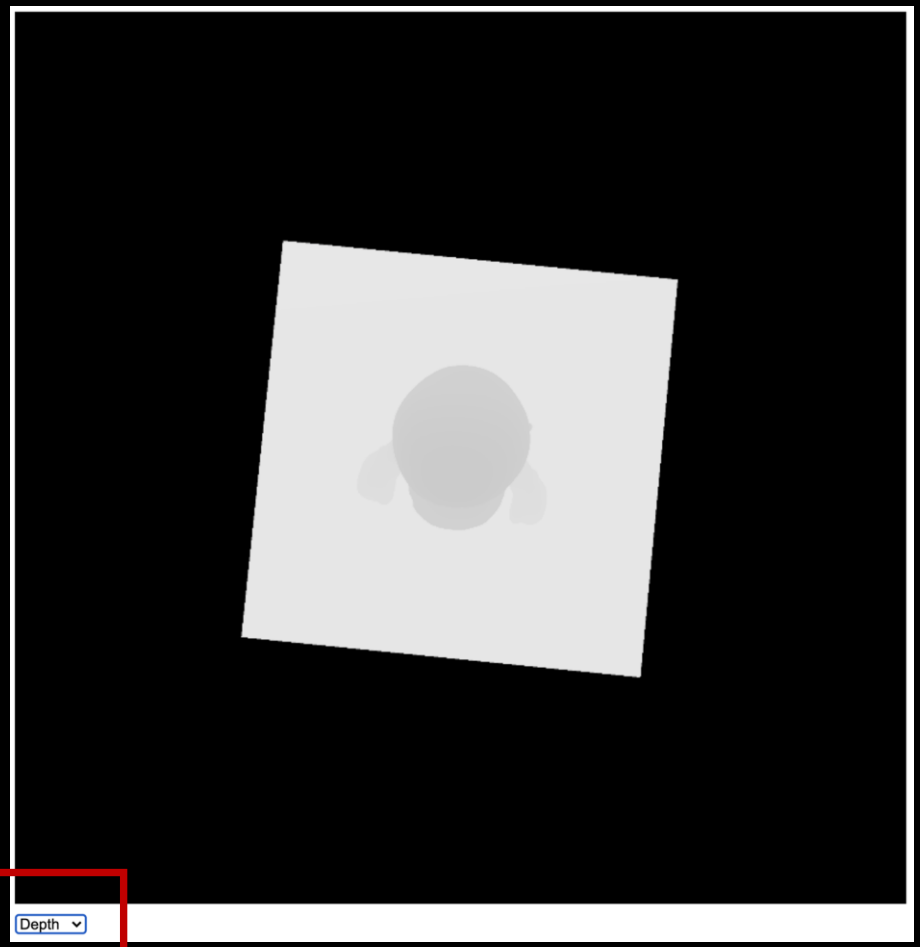
Lab 9

- Download the lab9 template
- Use the dropdown menu to show what the “light” sees
- https://www.youtube.com/watch?v=tObYAhA2iGo&ab_channel=Ko-ChihWang

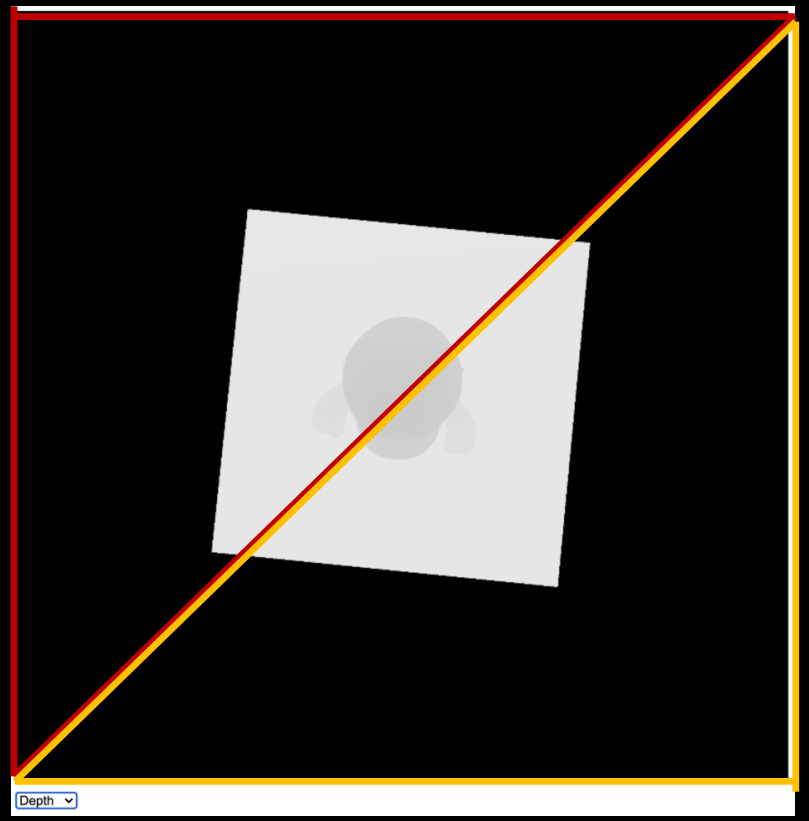




- Bottom of the main() function
- If users switch the drop-down menu, the value of ”normalMode” is modified to indicate what the canvas should show now.

```
    canvas.onmousedown = function(ev){mouseDown(ev)};
    canvas.onmousemove = function(ev){mouseMove(ev)};
    canvas.onmouseup = function(ev){mouseUp(ev)};
    var menu = document.getElementById("menu");
    menu.onchange = function() {
        if(this.value == "normal") normalMode = true;
        else normalMode = false;
        draw();
    }
}
```

- The idea of show a texture on canvas
  - Just draw a quad to cover whole canvas
    - Trick: set the quad coordinate to ([-1, +1], [-1, +1]) match the x-y range of coordinate of clip space
    - Directly set the quad coordinate to gl_Position in vertex shader
  - Look up the texture to color the quad in the fragment shader

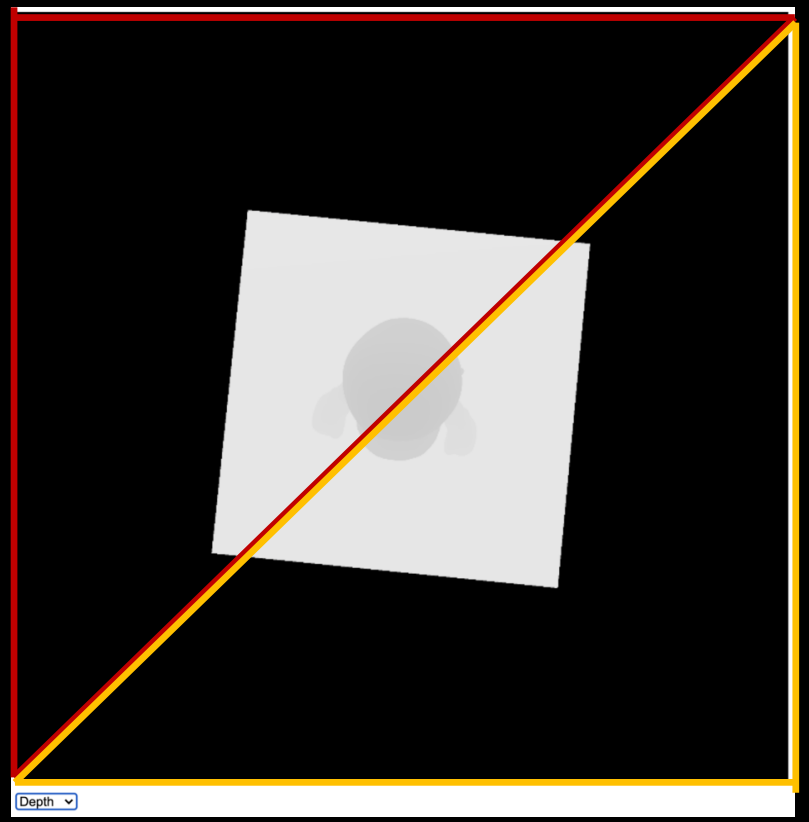



```
var rotateAngle = 0;
var normalMode = true;
var quadObj;

async function main(){
    canvas = document.getElementById('webgl');
    gl = canvas.getContext('webgl2');
    if(!gl){
        console.log('Failed to get the rendering context for WebGL');
        return ;
    }

    var quad = new Float32Array(
      [
        -1, -1, 0,
         1, -1, 0,
        -1,  1, 0,
        -1,  1, 0,
         1, -1, 0,
         1,  1, 0
      ]); //just a quad

    //setup shaders and prepare shader variables
    shadowProgram = compileShader(gl, VSHADER_SHADOW_SOURCE, FSHADER_SHADOW_SOURCE);
    shadowProgram.a_Position = gl.getAttribLocation(shadowProgram, 'a_Position');
    shadowProgram.u_MvpMatrix = gl.getUniformLocation(shadowProgram, 'u_MvpMatrix');

    program = compileShader(gl, VSHADER_SOURCE, FSHADER_SOURCE);
    program.a_Position = gl.getAttribLocation(program, 'a_Position');
    program.a_Normal = gl.getAttribLocation(program, 'a_Normal');
    program.u_MvpMatrix = gl.getUniformLocation(program, 'u_MvpMatrix');
    program.u_modelMatrix = gl.getUniformLocation(program, 'u_modelMatrix');
    program.u_normalMatrix = gl.getUniformLocation(program, 'u_normalMatrix');
    program.u_LightPosition = gl.getUniformLocation(program, 'u_LightPosition');
    program.u_ViewPosition = gl.getUniformLocation(program, 'u_ViewPosition');
    program.u_MvpMatrixOfLight = gl.getUniformLocation(program, 'u_MvpMatrixOfLight');
    program.u_Ka = gl.getUniformLocation(program, 'u_Ka');
    program.u_Kd = gl.getUniformLocation(program, 'u_Kd');
    program.u_Ks = gl.getUniformLocation(program, 'u_Ks');
    program.u_shininess = gl.getUniformLocation(program, 'u_shininess');
    program.u_ShadowMap = gl.getUniformLocation(program, "u_ShadowMap");
    program.u_Color = gl.getUniformLocation(program, 'u_Color');

    quadProgram = compileShader(gl, VSHADER_QUAD_SOURCE, FSHADER_QUAD_SOURCE);
    quadProgram.a_Position = gl.getAttribLocation(quadProgram, 'a_Position');
    quadProgram.u_ShadowMap = gl.getUniformLocation(quadProgram, "u_ShadowMap");
```

Btw, we compile the quadProgram here

- The idea of show a texture on canvas
  - Just draw a quad to cover whole canvas
    - Trick: set the quad coordinate to ([-1, +1], [-1, +1]) match the x-y range of coordinate of clip space
    - Directly set the quad coordinate to gl_Position in vertex shader
  - Look up the texture to color the quad in the fragment shader (this is your TODO-2)

```
var VSHADER_QUAD_SOURCE = `
    attribute vec4 a_Position;
    void main(){
        gl_Position = a_Position;
    }
`;

var FSHADER_QUAD_SOURCE = `
    precision mediump float;
    uniform sampler2D u_ShadowMap;
    void main(){
        //TODO-2: look up the depth from u_ShaodowMap and draw on quad (just one line)
    }
`;
```



- If the mode is not normal Mode (normalMode == false)
  - In order to show what the “light” sees, you should use ”quadProgram”(shader) to draw a quad, pass the fbo.texture to the shader to color the quad
  - This is your TODO-1

```
function draw(){
  ///// off scree shadow
  gl.useProgram(shadowProgram);
  gl.bindFramebuffer(gl.FRAMEBUFFER, fbo);
  gl.viewport(0, 0, offScreenWidth, offScreenHeight);
  gl.clearColor(0.0, 0.0, 0.0, 1);
  gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
  gl.enable(gl.DEPTH_TEST);
  //cube
  let cubeMdlMatrix = new Matrix4();
  cubeMdlMatrix.setScale(2.0, 0.1, 2.0);
  let cubeMvpFromLight = drawOffScreen(cubeObj, cubeMdlMatrix);
  //mario
  let marioMdlMatrix = new Matrix4();
  marioMdlMatrix.setTranslate(0.0, 1.4, 0.0);
  marioMdlMatrix.scale(0.02,0.02,0.02);
  let marioMvpFromLight = drawOffScreen(marioObj, marioMdlMatrix);

  ///// on scree rendering
  if( normalMode ){
    gl.useProgram(program);
    gl.bindFramebuffer(gl.FRAMEBUFFER, null);
    gl.viewport(0, 0, canvas.width, canvas.height);
    gl.clearColor(0.4,0.4,0.4,1);
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
    gl.enable(gl.DEPTH_TEST);
    //cube
    drawOneObjectOnScreen(cubeObj, cubeMdlMatrix, cubeMvpFromLight, 1.0, 0.4, 0.4);
    //mario
    drawOneObjectOnScreen(marioObj, marioMdlMatrix, marioMvpFromLight, 0.4, 1.0, 0.4);
  }else{
    //TODO-1:
    //draw the shadow map (the quad)
    //active the quadProgram
    //switch the destination back to normal canvas color buffer
    //pass fbo.texture into the quadProgram
    //draw the quad ()
  }
}
```

What You Should Do for “Submission”

Submission Instruction

- Create a folder
  - Put the html and js files in the folder
  - Zip the folder
  - Rename the zip file to your student ID
    - For example, if your student ID is “40312345s”, rename the zip file to “40312345s.zip”
  - Submit the renamed zip file to Moodle

- Make sure
  - you put all files in the folder to zip
  - You submit the zip file with correct name
- You won’t get any point if
  - the submitted file does not follow the naming rule,
  - TA cannot run your code,
  - or cannot unzip your zip file.