In engine:

<script src="objparser.js"></script>

var g\_discofloor = []; // list of cubes

Lighting data:

spotlight: new Vector3(0.0, 1.0, 0.0).normalize(),

            spotlight\_dir: new Vector3(1.0, -1.0, 0.0).normalize(),

            spotlight\_color: new Vector3(1.0, 0.0, 0.0),

            spotlight\_limit: 0.9,

            spotlight2: new Vector3(0.0, 1.0, 0.0).normalize(),

            spotlight\_dir2: new Vector3(-1.0, -1.0, 0.0).normalize(),

            spotlight\_color2: new Vector3(0.0, 0.0, 1.0),

in init:

// make disco floor

            let cube\_mesh = Primitives.Cube.getMesh(gl);

            let temp\_material = {tint: [1.0, 1.0, 1.0, 1.0]} // temp white

            for (let i = -1; i < 2; i++)

            {

                for(let j = -1; j < 2; j++)

                {

                    let new\_cube = new Model(new Transform(), cube\_mesh, temp\_material);

                    new\_cube.transform.position.set(2\*i, 0.0 , 2\*j);

                    new\_cube.transform.scale.set(2.0, 0.1, 2.0);

                    g\_discofloor.push(new\_cube);

                }

            }

In draw

Matrix4x4.transformV3(Matrix4x4.rotationMatrixY(deltaTime\*75),

                 g\_lightingData.spotlight\_dir, g\_lightingData.spotlight\_dir);

            Matrix4x4.transformV3(Matrix4x4.rotationMatrixY(-deltaTime\*75),

                 g\_lightingData.spotlight\_dir2, g\_lightingData.spotlight\_dir2);

            // chose color for floor - change to if beat change odd boxes to rand color

            for (let i = 0; i < g\_discofloor.length; i++)

            {

                if(i % 2 == 0)

                {

                    g\_discofloor[i].material.tint = [0, 0, 0, 1];

                } else {

                    g\_discofloor[i].material.tint = [1, 1, 1, 1];

                }

                g\_discofloor[i].update();

                g\_discofloor[i].render(g\_standardShader, g\_camera, g\_lightingData);

            }

Shaders:

<script type="vertex-shader" id="standardVertexShader">

        attribute vec3 a\_position; // the position of each vertex

        attribute vec3 a\_normal;   // the surface normal of each vertex

        attribute vec2 a\_texcoord;

        uniform mat4 u\_matrixM; // the model matrix of this object

        uniform mat4 u\_matrixV; // the view matrix of the camera

        uniform mat4 u\_matrixP; // the projection matrix of the camera

        // inverse transpose to change normals into world space, this will make sure

        // our normals are not stretched when the object is scaled.

        uniform mat3 u\_matrixInvTransM;

        uniform vec3 u\_viewPos;    // position of the camera in world space

        uniform vec3 u\_spotlight; // spotlight position

        uniform vec3 u\_spotlight2;

        // forward to the fragment shader

        varying vec3 v\_normal;

        varying vec2 v\_texcoord;

        varying vec3 v\_surfaceToLight;

        varying vec3 v\_surfaceToLight2;

        varying vec3 v\_surfaceToView;

        void main() {

            v\_normal = normalize(u\_matrixInvTransM \* a\_normal); // set normal data for fragment shader

            v\_texcoord = a\_texcoord;

            // calculate new position

            gl\_Position = u\_matrixP \* u\_matrixV \* u\_matrixM \* vec4 (a\_position, 1);

            // calc surfaceToLight and surfaceToView

            vec3 world\_pos = (u\_matrixM \* vec4(a\_position, 1)).xyz;

            v\_surfaceToLight = u\_spotlight - world\_pos;

            v\_surfaceToLight2 = u\_spotlight2 - world\_pos;

            v\_surfaceToView = u\_viewPos - world\_pos;

        }

    </script>

    <script type="fragment-shader" id="standardFragmentShader">

        precision highp float; //float precision settings

        uniform vec4 u\_tint;            // the tint color of this object

        // directional lighting

        uniform vec3 u\_directionalLight;// directional light in world space

        uniform vec3 u\_directionalColor;// light color

        uniform vec3 u\_ambientColor;    // intensity of ambient light

        // spotlight

        uniform vec3 u\_spotlightDir;

        uniform vec3 u\_spotlightColor;

        uniform vec3 u\_spotlightDir2;

        uniform vec3 u\_spotlightColor2;

        uniform float u\_limit;          // bounds of the spotlight

        varying vec3 v\_normal;  // normal from the vertex shader

        varying vec3 v\_surfaceToLight;

        varying vec3 v\_surfaceToView;

        varying vec3 v\_surfaceToLight2;

        // main texture sampler

        uniform sampler2D u\_mainTex;    // texture

        varying vec2 v\_texcoord; // vector for texture coords

        void main(void){

            vec3 normal = normalize(v\_normal);

            // lambertian diffuse lighting

            float diffuse = max(0.0, dot(normal, -u\_directionalLight));

            vec3 diffuseColor = u\_directionalColor \* diffuse;

            // spotlighting

            vec3 surfaceToLightDirection = normalize(v\_surfaceToLight);

            vec3 surfaceToViewDirection = normalize(v\_surfaceToView);

            vec3 halfVec = normalize(surfaceToLightDirection + surfaceToViewDirection);

            float dotFromDirection = dot(surfaceToLightDirection, -u\_spotlightDir);

            // check if within the spotlight bounds

            float spotlightMag = 0.0;

            if(dotFromDirection >= u\_limit)

            {

                spotlightMag = dot(normal, surfaceToLightDirection);

                // do spec if desired here

            }

            vec3 spotlight = u\_spotlightColor \* spotlightMag;

            vec3 surfaceToLightDirection2 = normalize(v\_surfaceToLight2);

            vec3 halfVec2 = normalize(surfaceToLightDirection2 + surfaceToViewDirection);

            float dotFromDirection2 = dot(surfaceToLightDirection2, -u\_spotlightDir2);

            float spotlightMag2 = 0.0;

            if(dotFromDirection2 >= u\_limit)

            {

                spotlightMag2 = dot(normal, surfaceToLightDirection2);

                // do spec if desired here

            }

            vec3 spotlight2 = u\_spotlightColor2 \* spotlightMag2;

            // combine lighting from directional and ambient light

            vec3 ambientDiffuse = u\_ambientColor + spotlight + spotlight2;

            // make sure light intensity does not exceed 1.0

            ambientDiffuse = clamp(ambientDiffuse, vec3(0.0,0.0,0.0), vec3(1.0,1.0,1.0));

            // texture and tint color

            vec4 textureColor = texture2D(u\_mainTex, v\_texcoord);

            vec4 combinedColor = u\_tint; //textureColor \* u\_tint

            // TODO: use multiplicative color blending to combine lighting and color

            gl\_FragColor = vec4(ambientDiffuse \* combinedColor.rgb, combinedColor.a);

            //gl\_FragColor = vec4(v\_texcoord, 0, 1);

        }

    </script>