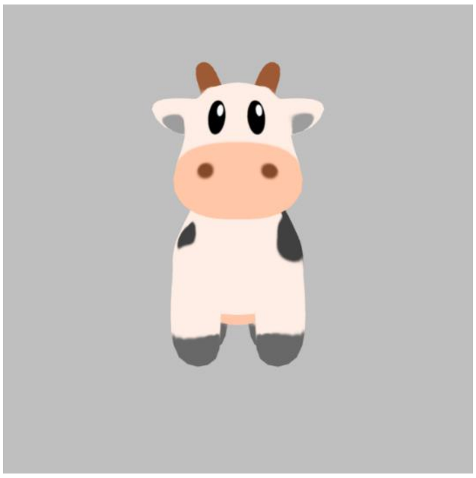
Computer Graphics Lab 4

Sean Sanii Nejad

A1i.png Image



A1ii.png Image

A1i code

```
// A1-A2 -- CHANGE THIS
// combined colour
//gl_FragColor = vec4((0.5*ambient + diffuse + 0.01*specular).rgb, 1.0);
gl_FragColor = texture2D(sampler: texture,coord: map);
```

A1ii code

```
// A1 -- CHANGE THIS
// uniform grey
vec4 material_colour = texture2D(sampler: texture,coord: map);
```

Explanation

The lit model does not have any textures that resembles a cow, just a white outline of the model. The textured model in A1i resembles a cow, however it is lacking shadows/lighting details. The lit+texture model as shown in A1ii looks a lot better than the previous two. Having both texture and lighting gives additional depth that is pleasing on the eyes.

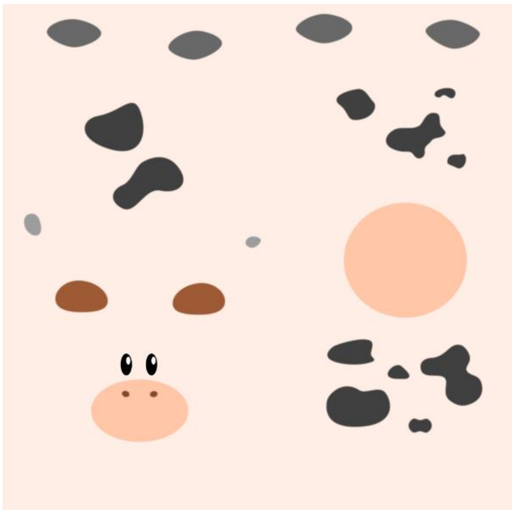
A2i.png image

A2i code

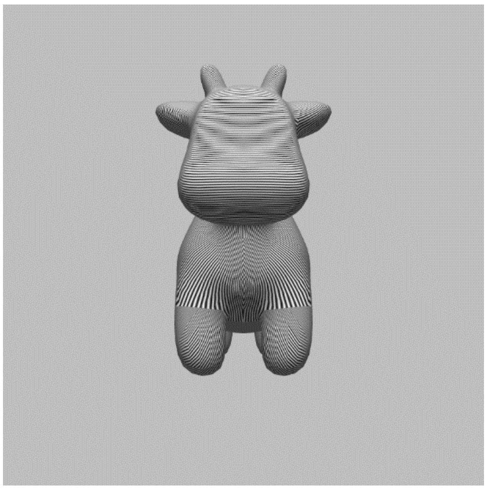
```
    // combined colour
    //gl_FragColor = vec4((0.5*ambient + diffuse + 0.01*specular).rgb, 1.0);
    //gl_FragColor = texture2D(texture,map);
    gl_FragColor = vec4(v0: map.s, v1: map.t, v2: 0.0, v3: 1.0);
```

Explanation

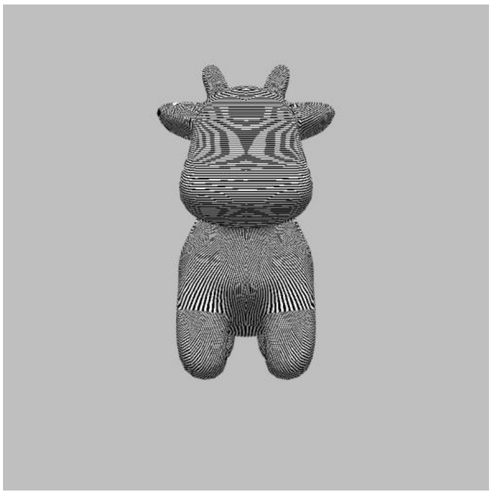
I believe the head is darker than the rest of the body because these values map to the bottom left of the texture map. Lower value (near 0) is darker than the top right which are higher values of s and t (near 1). This is illustrated in the cow texture map below.



A3i.png image



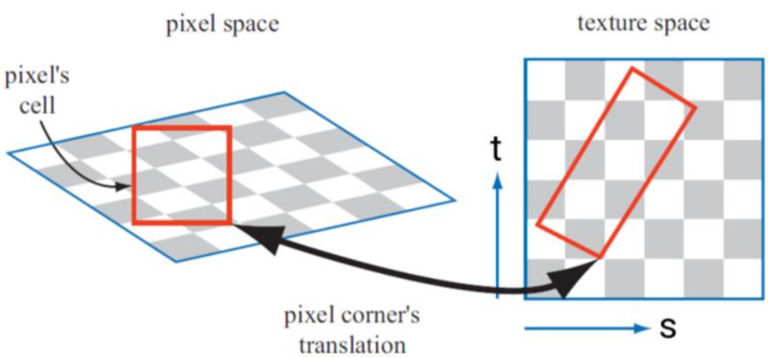
A3ii.png image



A3ii.png code

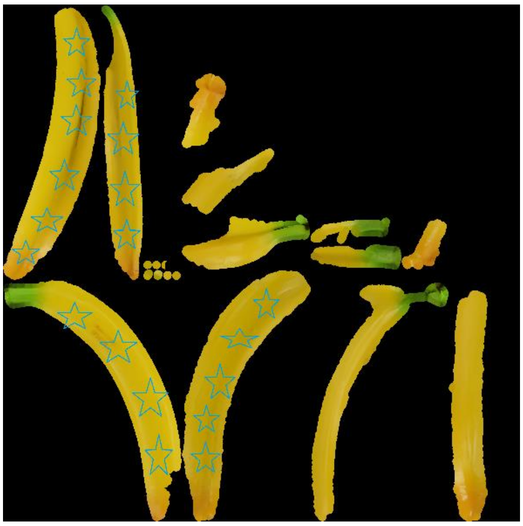
```
    // A3 -- MODIFY THESE
    // interpolation method for shrinking and enlarging the texture, respectively:
    // gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR_MIPMAP_LINEAR);
    // gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.LINEAR);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.NEAREST);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.NEAREST);
```

Explanation

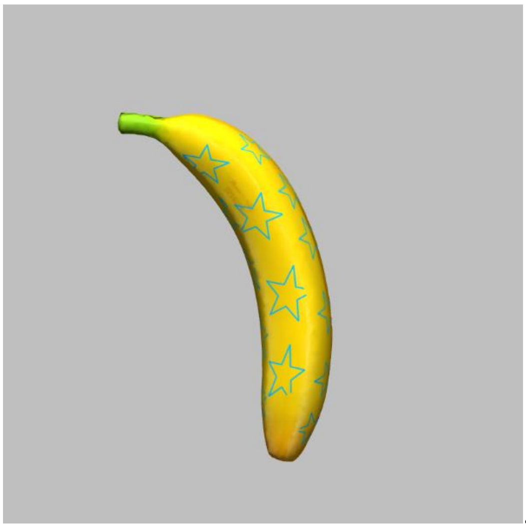


I believe these artefacts exist due to a large texture space being fit into a small pixel space. There is not enough pixels to accurately represent all the data that exists within the texture. Bilinear/mipmap seems to cause some blurring effect which is a lot more appealing to the eye than using the nearest neighbour method which causes severe aliasing.

A4i.png image



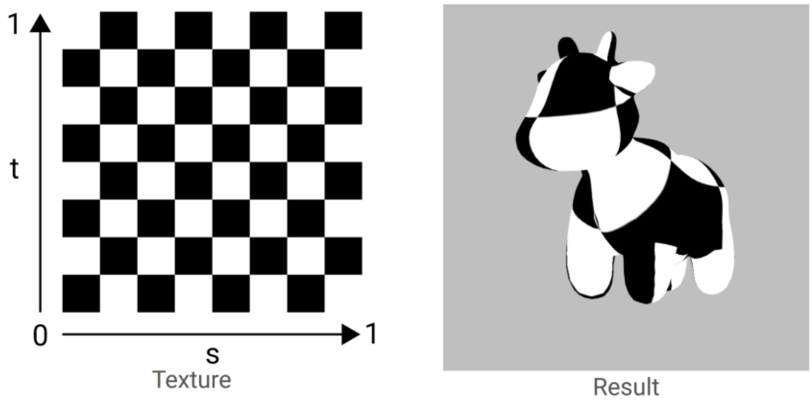
A4ii.png image



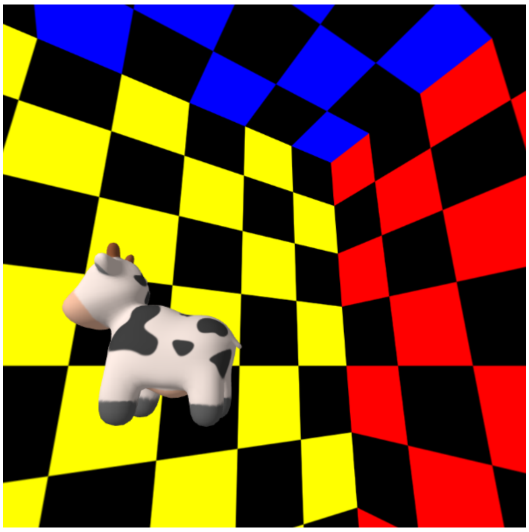
.

Explanation

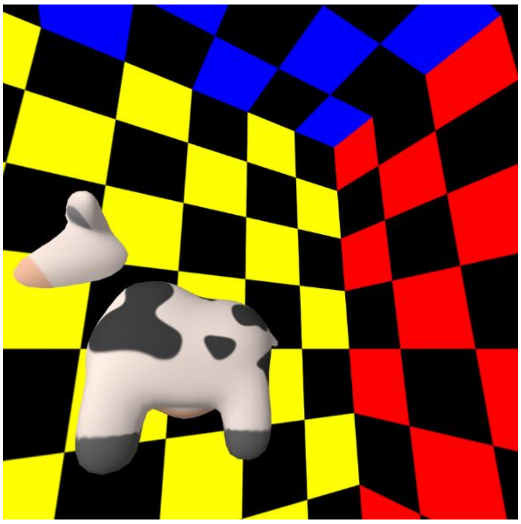
As I edited the png file being used to map the image onto the model. The corresponding model with the texture edited shows the changes (A4ii.png). This is because Webgl maps whatever is on the png to the model as shown below.



B1i.png image



B1ii.png image



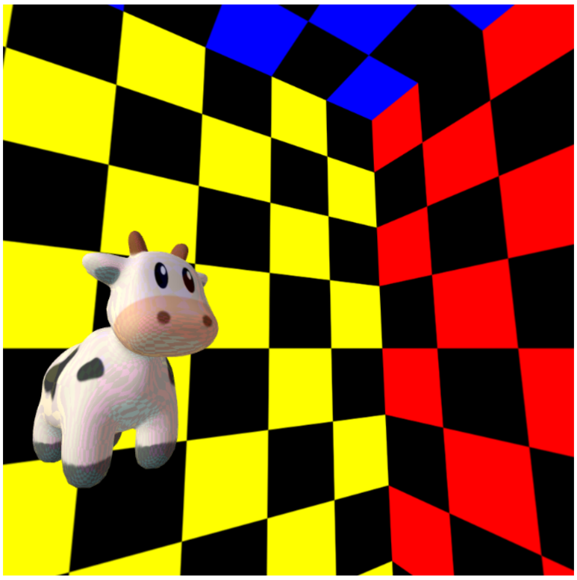
B1 code

```
    gl.disable(gl.DEPTH_TEST);
    // B1 -- UNCOMMENT THIS
    render_skybox();
```

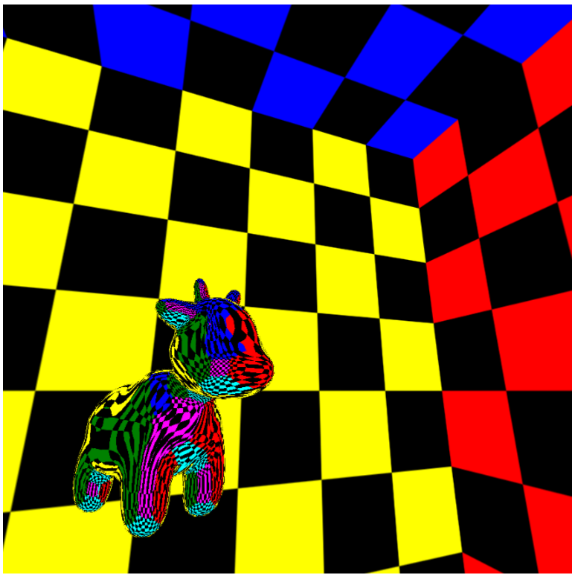
Explanation

I believe this effect is desirable as we do not want to render objects that are infront of other objects. We also do want to disregard fragments that fail the depth test. Primitives that fall outside the skybox should be removed as they are not supposed to be there and only waste computation.

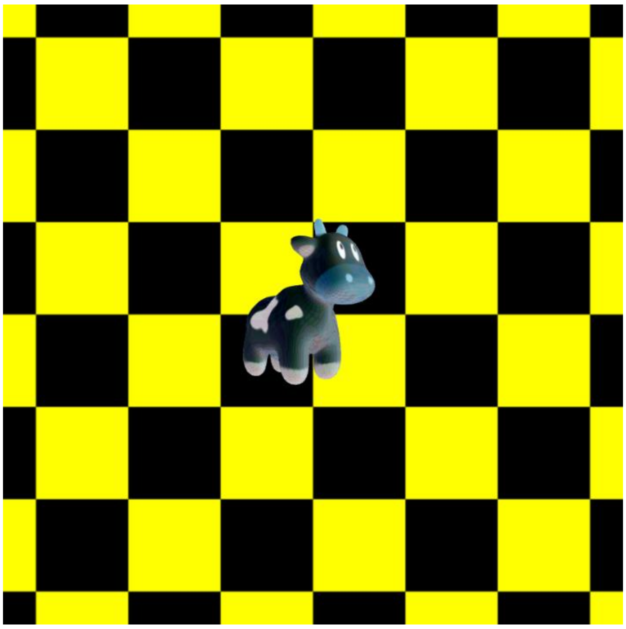
B2i.png image



B2ii.png image



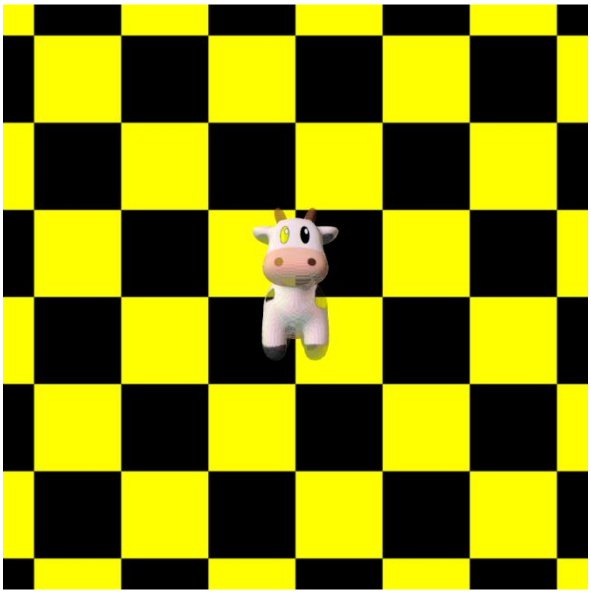
C1i.png image



C1i code

```
if(render_texture) {
    // B2 -- MODIFY
    gl_FragColor = vec4(v0: -1.0*(0.5 * ambient +
                        0.5 * diffuse +
                        0.01 * specular +
                        0.1 * reflection_colour).rgb+1.0, v1: 1.0);
}
```

C2i.png image



C2i code

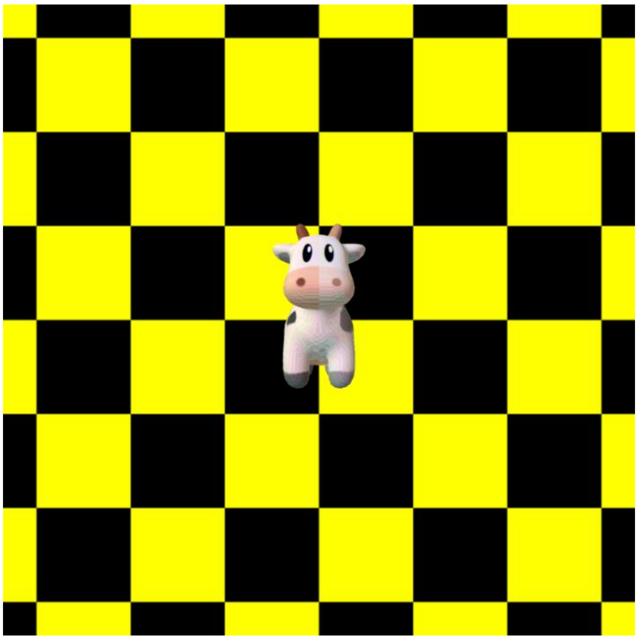
```
    if(!gl_FrontFacing) {
        // fragment faces away from camera
        discard;
    }
    // combined colour
    if(render_texture) {
        // B2 -- MODIFY

        float length = length(x: vec3(value: (0.5 * ambient +
                                 0.5 * diffuse +
                                 0.01 * specular +
                                 0.1 * reflection_colour).rgb));

        gl_FragColor = vec4(v0: (0.5 * ambient +
                                 0.5 * diffuse +
                                 0.01 * specular +
                                 0.1 * reflection_colour).rgb, v1: length);
    }
```

Explanation

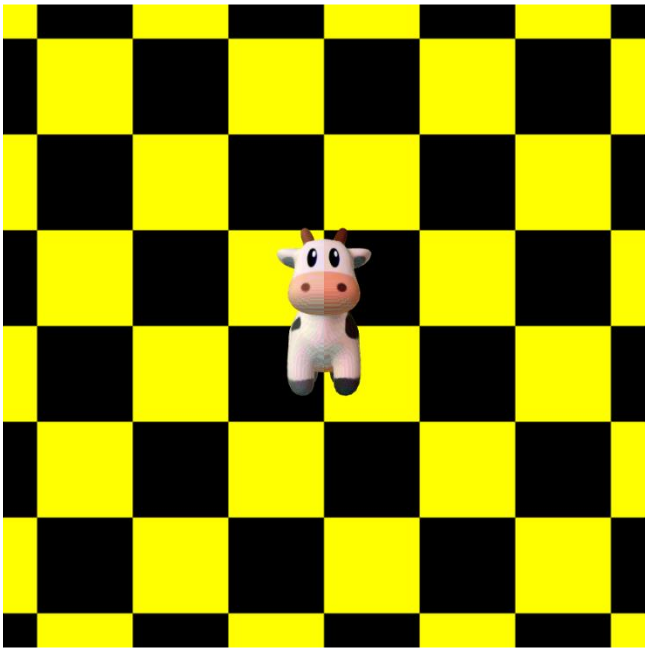
I used the length method to calculate how dark a fragment is and assigned it to the variable length. I then used this variable in the parameter for opacity when assigning the vec4 to the gl_Fragcolor. This achieved the transparency on dark fragments as required.

If fragments are not facing the front (normals), they will not be rendered. So when something is transparent, we won’t see the back of the model if it’s not transparent as well.

C3i.png



C3ii.png

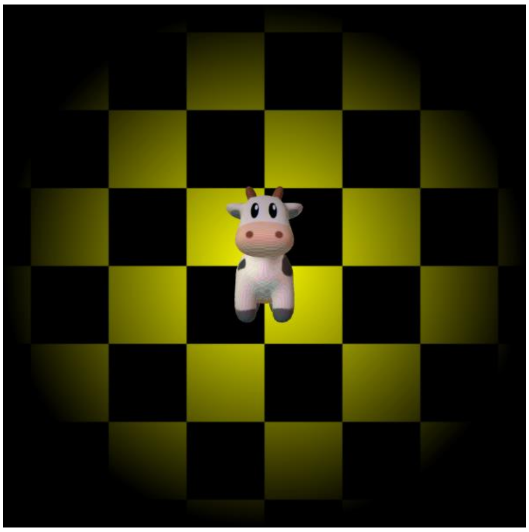


C3 code

```
vec4 gamma_transform(vec4 colour, float gamma)
{
    return vec4(v0: pow(x: colour.rgb, y: vec3(value: gamma)), v1: colour.a);
}

void main()
{
    vec3 n = normalize(v: m);

    if(render_skybox) {
        gl_FragColor = textureCube(sampler: cubemap,coord: vec3(v0: -d.x,v1: d.y,v2: d.z));
    }
    else {

        // object colour
        vec4 material_colour = texture2D(sampler: texture,coord: map);

        if(gl_FragCoord.x > 850.0/2.0) {
            // do gamma transformation here
            material_colour = gamma_transform(colour: material_colour, gamma: 0.5);
        }
```

C4i.png image

C4ii.png image



C4 code

```
float vignette()
{
    float rx = gl_FragCoord.x - radius;
    float ry = gl_FragCoord.y - radius;
    return (1.0 - sqrt(x: pow(x: rx,y: 2.0) + pow(x: ry,y: 2.0)) / radius);
}
```

```
        }
        gl_FragColor.rgb *= vignette();
    }
```

D1i.png image

D2i.png image

D2ii.png image

Gamma 0.5

D3i.png image

D3ii.png image

D3 code

```
float vignette()
{
    float rx = gl_FragCoord.x - radius;
    float ry = gl_FragCoord.y - radius;
    return (1.0 - sqrt(x: pow(x: rx,y: 2.0) + pow(x: ry,y: 2.0)) / radius);
}

vec4 gamma_transform(vec4 colour, float gamma)
{
    return vec4(v0: pow(x: colour.rgb, y: vec3(value: gamma)), v1: colour.a);
}
```

```
vec4 material_colour = texture2D(sampler: texture,coord: map);
gamma_transform(colour: material_colour, gamma: 0.5);
```

```

<img id ='cubemap_img0' src='../shared/models/leadenhall-1/right.png' hidden></img>
<img id ='cubemap_img1' src='../shared/models/leadenhall-1/left.png' hidden></img>
<img id ='cubemap_img2' src='../shared/models/leadenhall-1/over.png' hidden></img>
<img id ='cubemap_img3' src='../shared/models/leadenhall-1/under.png' hidden></img>
<img id ='cubemap_img4' src='../shared/models/leadenhall-1/front.png' hidden></img>
<img id ='cubemap_img5' src='../shared/models/leadenhall-1/back.png' hidden></img>
```

```
    if(render_lines) {
        gl_FragColor = vec4(v0: 0.56, v1: 0.56, v2: 0.56, v3: 1.0);
    }
```

```
function render_object()
{
    gl.uniform1i(gl.getUniformLocation(program,'render_skybox'), false);
    gl.bindBuffer(gl.ARRAY_BUFFER, mesh.vertexBuffer);
    gl.vertexAttribPointer(vertex_loc, mesh.vertexBuffer.itemSize, gl.FLOAT, false, 0, 0);

    gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, mesh.indexBuffer);

    // D2 MODIFY HERE
    if(render_lines == true){
        gl.drawElements(gl.LINES, mesh.indexBuffer.numItems, gl.UNSIGNED_SHORT, 0);
    }
    else {
        gl.drawElements(gl.TRIANGLES, mesh.indexBuffer.numItems, gl.UNSIGNED_SHORT, 0);
    }
}
```

```
    if(event.keyCode == 85) {
        render_object();

        render_lines = !render_lines;
        gl.uniform1i(gl.getUniformLocation(program,'render_lines'), render_lines);

```

Explanation

Applied vignette

Applied gamma at 0.5

Added leandenhall-1 background

Implemented key U to render lines or triangle primitives for the Cow

Added Spot (Cow)