Computer Graphics - Exercise 08

Maximilian Richter

Winter Semester 2022/23

1 Rasterization I: Transformations

a)
```cpp
void SimpleRasterizer::RenderMesh(const Mesh *mesh)
{
  if (mesh == NULL)
    return;

  // Exercise 8.1 a)

  const mat4 modelTransform = mesh->GetGlobalTransformation();
  const mat4 modelTransformNormals = inverseTranspose(modelTransform);

  for (Triangle t: mesh->GetTriangles()) {
    this->TransformAndLightTriangle(t, modelTransform, modelTransformNormals);
    this->DrawTriangle(t);
  }
}
```

b)
```cpp
const mat4 viewTransformation = lookAt(
    camera->GetEye(),
    camera->GetLookAt(),
    camera->GetUp()
);

// Generates a really hard-to-read matrix, but a normal, standard 4x4 matrix nonetheless
const mat4 projectionMatrix = perspective(
    camera->GetFov(),
    camera->GetAspect(),
    camera->GetNearClip(),
    camera->GetFarClip()
);

this->viewProjectionTransform = projectionMatrix * viewTransformation;
```

c)
```cpp
void SimpleRasterizer::TransformAndLightTriangle(Triangle &t,
                                                  const mat4 &modelTransform,
                                                  const mat4 &modelTransformNormals)
{
```

```
  // Exercise 8.1 c)

  for (int i=0; i<3; i++) {
    // Apply model transform to go from model coordiantes to world coordinates
    vec4 worldCoords =  modelTransform * vec4(t.position[i], 1.0f);
    vec3 worldNormal = normalize(vec3(modelTransformNormals * vec4(t.normal[i], 0.0f)));

    // Light vertex in world coordinates
    t.color[i] = LightVertex(worldCoords, worldNormal, t.color[i]);

    // Get clip coordinates by ViewProjectionTransformation
    vec4 clipCoords = this->viewProjectionTransform * worldCoords;

    // Get normalized device coordinates (i.e. x,y in [-1,1])
    clipCoords.x /= clipCoords.w;
    clipCoords.y /= clipCoords.w;
    clipCoords.z /= clipCoords.w;

    // Apply viewport transform to get windows coordinates and set new positions
    t.position[i].x = (clipCoords.x +1.0)*(image->GetWidth()/2.0);
    t.position[i].y = (-1.0f*clipCoords.y +1.0)*(image->GetHeight()/2.0);
  }
}
```

2 Clipping

First we find the Outcodes with Outcode $= \{x < x_{\min}, x > x_{\max}, y < y_{\min}, y > y_{\max}\}$

First we start with the bottom line with $P_1 = (1, 3)$ and $P_2 = (6, 2)$:
Outcode(P_1)=1000
Outcode(P_2)=0100

Cohen-Sutherland:

1. Outcode(P_1)$\vee$Outcode(P_2) $\neq 0 \quad \Rightarrow \quad$ no trivial accept
2. Outcode(P_1)$\wedge$Outcode(P_2) $== 0 \quad \Rightarrow \quad$ no trivial reject
3. Outcode(P_1) "left" bit set $\quad \Rightarrow \quad$ Calculate S_1

$$y = -\frac{x}{5} + \frac{16}{5} \quad \Rightarrow \quad S_1 = (2, \frac{14}{5})$$

   $\Rightarrow$ proceed with S_1P_2

4. Outcode(S_1)$\wedge$Outcode(P_2) $\neq 0 \quad \Rightarrow \quad$ no trivial accept
5. Outcode(S_1)$\wedge$Outcode(P_2) $== 0 \quad \Rightarrow \quad$ no trivial reject
6. Outcode(P_2) "right" bit set $\quad \Rightarrow \quad$ Calculate S_2

$$S_2 = (5, \frac{11}{5})$$

proceed with S_1S_2

7. Outcode(S_1)$\vee$Outcode(S_2) $== 0 \quad \Rightarrow \quad$ trivial accept (S_1, S_2)

Now the top line with $P_1 = (3,3)$ and $P_2 = (5,6)$:
Outcode(P_1)=0000
Outcode(P_2)=0001

Cohen-Sutherland:

1. Outcode(P_1)$\vee$Outcode(P_2) $\neq 0 \quad \Rightarrow \quad$ no trivial accept
2. Outcode(P_1)$\wedge$Outcode(P_2) $== 0 \quad \Rightarrow \quad$ no trivial reject
3. Outcode(P_1) "left" bit set $\quad \Rightarrow \quad$ Calculate S

$$y = \frac{3x}{2} + \frac{3}{2} \quad \Rightarrow \quad S = (\frac{13}{3}, 5)$$

$\Rightarrow$ proceed with P_1S

4. Outcode(P_1)$\vee$Outcode(S) $== 0 \quad \Rightarrow \quad$ trivial accept (P_1, S)