Philipp von Bachmann,
Mat.-Nr.: 4116220
David Ott,
Mat.-Nr.: 4185646

1	2	3	4	5	Σ

Assignment Nr. 9

9.1 Transformations

1. We scale by 4 in the x-direction and 5 in the y-direction
2. Shear top part to the right
3. rotate by 45 degree counter-clockwise
4. translate the Middle point to $(4, 5)^T$

So the final matrix is:

$$\begin{pmatrix}1 & 0 & 4-\sqrt{2}\\ 0 & 1 & 5-3.5\sqrt{2}\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}cos(45°) & -sin(45°) & 0\\ sin(45°) & cos(45°) & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 1 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}4 & 0 & 0\\ 0 & 5 & 0\\ 0 & 0 & 1\end{pmatrix}$$

The middle point after scaling, shear, rotation was found by:

$$\begin{pmatrix}cos(45°) & -sin(45°) & 0\\ sin(45°) & cos(45°) & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 1 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}4 & 0 & 0\\ 0 & 5 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}0.5\\ 0.5\\ 1\end{pmatrix} = \begin{pmatrix}\sqrt{2}\\ 3.5\sqrt{2}\\ 1\end{pmatrix}$$

9.2 Affine Spaces

a)

1)

let $p = (p_1, p_2, p_3, 1)^T$ and let $v = (v_1, v_2, v_3, 0)^T, w = (w_1, w_2, w_3, 0)^T$

$$\begin{aligned}(p+v)+w &= \left(\begin{pmatrix}p_1\\ p_2\\ p_3\\ 1\end{pmatrix}+\begin{pmatrix}v_1\\ v_2\\ v_3\\ 0\end{pmatrix}\right)+\begin{pmatrix}w_1\\ w_2\\ w_3\\ 0\end{pmatrix} = \begin{pmatrix}p_1+v_1+w_1\\ p_2+v_2+w_2\\ p_3+v_3+w_3\\ 1+0+0\end{pmatrix} = \begin{pmatrix}p_1\\ p_2\\ p_3\\ 1\end{pmatrix}+\left(\begin{pmatrix}v_1\\ v_2\\ v_3\\ 0\end{pmatrix}+\begin{pmatrix}w_1\\ w_2\\ w_3\\ 0\end{pmatrix}\right)\\ &= p+(v+w)\end{aligned}$$

2)

$v \in V$ exists with:

$$q - p = v$$

v is unique, proof by contradiction: Assume for unique but chosen p, q there exist two different v_1, v_2 with $p + v_1 = q$ and $p + v_2 = q$. Therefore:

$$\begin{aligned}p + v_1 &= q = p + v_2\\ p + v_1 &= p + v_2\\ v_1 &= v_2\end{aligned}$$

So $v_1 = v_2$ which contradicts the assumption, so v is unique.

3)

The associated vector space is $\{(x,0)^T | x \in \mathbb{R}^3\}$

b)

A point in this affine space has a 1 as its w while a vector has a 0 as its fourth coordinate.

9.3 Rotations

$$\begin{aligned}
\begin{pmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{pmatrix} &= \begin{pmatrix} 1 & \lambda_2 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \lambda_1 & 1 \end{pmatrix} \\
&= \begin{pmatrix} a & \lambda_2 b \\ 0 & b \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \lambda_1 & 1 \end{pmatrix} \\
&= \begin{pmatrix} a + \lambda_1 \lambda_2 b & \lambda_2 b \\ \lambda_1 b & b \end{pmatrix}
\end{aligned}$$

Therefore:

$$\begin{aligned}
b &= \cos(\phi) \\
\lambda_1 &= \frac{\sin(\phi)}{\cos(\phi)} \\
\lambda_2 &= \frac{-\sin(\phi)}{\cos(\phi)} \\
\cos(\phi) &= a + \lambda_1 \lambda_2 b = a + \frac{\sin(\phi)}{\cos(\phi)} \cdot \frac{-\sin(\phi)}{\cos(\phi)} \cdot \cos(\phi) \\
\cos(\phi) &= a + \frac{-\sin(\phi)^2}{\cos(\phi)} \\
\cos(\phi) + \frac{\sin(\phi)^2}{\cos(\phi)} &= a
\end{aligned}$$

So the final transformations are:

$$\begin{pmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{pmatrix} = \begin{pmatrix} 1 & \frac{-\sin(\phi)}{\cos(\phi)} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(\phi) + \frac{\sin(\phi)^2}{\cos(\phi)} & 0 \\ 0 & \cos(\phi) \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \frac{\sin(\phi)}{\cos(\phi)} & 1 \end{pmatrix}$$

9.4 Transformations

a)

Rotation matrices are always orthogonal matrices (They don't scale in any direction and they preserve angles). For an orthogoal matrix O it holds that $O^T O = O O^T = I$, so $O^{-1} = O^T$

b)

This 2D transformation rotates 45° counter clockwise around the origin and then translates by 1 in x-direction and by 1 in the y-direction.

9.5 Rasterization

a)

Brute-Force: For each point examine a line in an arbitrary direction and count the number of edge crossings. If the count is even we are outside of the polygon, while we are inside if the count is odd.
Scanline Algorithm: We scan all lines along a specific direction and use the odd-even parity rule to determine whether a point is inside a polygon. For each scan line we store the line-intersections in an Active-Edge-Table

b)

One method to do line rasterization is **Bresenham's algorithm**.

Another method is the **Brute-Force-Algorithm**. For the driving axis of the line for all individual values on this axis in between the start and end point the individual function values of the line are calculated and then rounded to the nearest pixel coordinate on the second axis. The pixel locations are then the current value on the initial axis and the rounded value on the other axis.