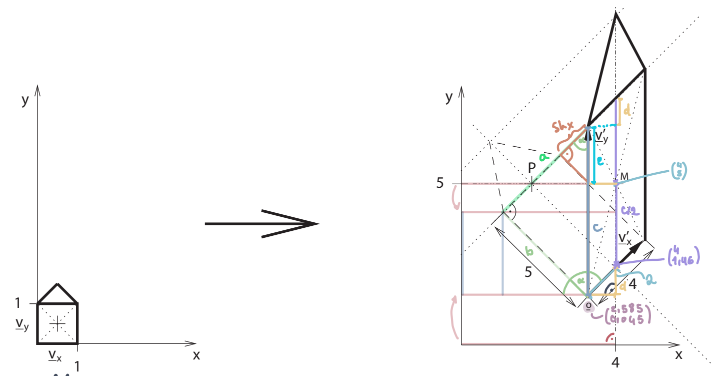
Tina Truong
Tanja Huber

	1	2	3	4	5	Σ

Exercise Nr. 9

(We worked on most exercises together with Stephan Amann and Amelie Schäfer so solutions might be similar.)

9.1



① Get origin and general information

The angle α is given with b being the bisector of c and a parallel line to the Y-Axis.

$\Rightarrow \alpha = 45°$

The triangle spanned by abc is therefore an isoscles and we can calculate c by the pythagorem theorem:

$$
\begin{aligned}
c &= \sqrt{a^2 + b^2} \quad \text{(a = b because of isoscles properties)} \\
&= (5^2 + 5^2)^{1/2} \\
&= 50^{1/2} \\
&= 5\sqrt{2} \quad \text{(length)}
\end{aligned}
$$

Given c and M we can calc. the origin O:

- $c/2 = \frac{5\sqrt{2}}{2}$ (length)
- M is the middle point of the parallelogram H

$\Rightarrow$ from $\binom{4}{5}$ we go half the length of H down

$$\binom{4}{5} - \binom{0}{\frac{c}{2}} = \binom{4}{1{,}46}$$

- to get d we have
  - $\alpha = 45°$, hypetenuse $= 2$

  $\Rightarrow \sin(\alpha) =$ opposite / hypotenuse

  opposite $= \sin(45°) \cdot$ hypotenuse $= \sqrt{2}$

  so $d = \sqrt{2}$
- therefore

$$\binom{4}{1{,}46} - \binom{\sqrt{2}}{\sqrt{2}} = \binom{2{,}585}{0{,}045} = O$$

② Get Shearing factor:

- get e by

$$e = c/2 - d = \frac{5\sqrt{2}}{2} - \sqrt{2} = \frac{3\sqrt{2}}{2}$$

- $Sh_x = \sin(45°) \cdot e = \frac{3}{2}$

So the basic transformation matrices in sequence are:

1. Scaling with $S = \binom{4}{5}$

$$\begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

2. Shearing in X-Axis with $Sh_x = 1{,}5$

$$\begin{pmatrix} 1 & 0 & 0 \\ Sh_x & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 3/2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

3. Rotation with $\Theta = 45°$

$$\begin{pmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \sqrt{2}/2 & -\sqrt{2}/2 & 0 \\ \sqrt{2}/2 & \sqrt{2}/2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

4. Translation with $T = \binom{2{,}585}{0{,}058}$

$$\begin{pmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 2{,}585 \\ 0 & 1 & 0{,}058 \\ 0 & 0 & 1 \end{pmatrix}$$

9.2

a)

$(1):\ a \in A$ and $v, w \in V : (a+v)+w = a+(v+w)$

$$\left(\begin{bmatrix}a_1\\a_2\\a_3\\1\end{bmatrix}+\begin{bmatrix}v_1\\v_2\\v_3\\0\end{bmatrix}\right)+\begin{bmatrix}w_1\\w_2\\w_3\\0\end{bmatrix}=\begin{bmatrix}a_1\\a_2\\a_3\\1\end{bmatrix}+\begin{bmatrix}v_1\\v_2\\v_3\\0\end{bmatrix}+\begin{bmatrix}w_1\\w_2\\w_3\\0\end{bmatrix}=\begin{bmatrix}a_1\\a_2\\a_3\\1\end{bmatrix}+\left(\begin{bmatrix}v_1\\v_2\\v_3\\0\end{bmatrix}+\begin{bmatrix}w_1\\w_2\\w_3\\0\end{bmatrix}\right)$$

Since $\mathbb{R}$ is associative and each line can be seen individually with $a+v+w$.

$(2):$ for each $a, b \in A$ there is a unique $v \in V : a+v = b$

$$\begin{bmatrix}a_1\\a_2\\a_3\\1\end{bmatrix}+\begin{bmatrix}v_1\\v_2\\v_3\\0\end{bmatrix}=\begin{bmatrix}b_1\\b_2\\b_3\\1\end{bmatrix}$$

The vector v is calculable as follows:

$$\begin{bmatrix}v_1\\v_2\\v_3\\0\end{bmatrix}=\begin{bmatrix}b_1-a_1\\b_2-a_2\\b_3-a_3\\1-1\end{bmatrix}=\begin{bmatrix}b_1-a_1\\b_2-a_2\\b_3-a_3\\0\end{bmatrix}$$

Since $\mathbb{R}$ is also closed regarding addition.

b)

A point in the affine space is written as $A = \{(x,y,z,w) \in \mathbb{R} | w = 1\}$.
A vector in the affine space is written as $V = \{(x,y,z,0) \in \mathbb{R}\}$.
So the difference is the number in the last dimension.

9.3

$$\begin{aligned}\begin{bmatrix}cos(\phi) & -sin(\phi) & 0\\ sin(\phi) & cos(\phi) & 0\\ 0 & 0 & 1\end{bmatrix} &= \begin{bmatrix}1 & h_{xy} & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix} * \begin{bmatrix}s_x & 0 & 0\\ 0 & s_y & 0\\ 0 & 0 & 1\end{bmatrix} * \begin{bmatrix}1 & 0 & 0\\ h_{yx} & 1 & 0\\ 0 & 0 & 1\end{bmatrix}\\ &= \begin{bmatrix}s_x & h_{xy} * s_y & 0\\ 0 & s_y & 0\\ 0 & 0 & 1\end{bmatrix} * \begin{bmatrix}1 & 0 & 0\\ h_{yx} & 1 & 0\\ 0 & 0 & 1\end{bmatrix}\\ &= \begin{bmatrix}s_x + h_{yx} * h_{xy} * s_y & h_{xy} * s_y & 0\\ h_{yx} * s_y & s_y & 0\\ 0 & 0 & 1\end{bmatrix}\end{aligned}$$

$$s_y = cos(\phi)$$

$$
\begin{aligned}
h_{xy} * s_y &= -sin(\phi) \\
h_{xy} * cos(\phi) &= -sin(\phi) \\
h_{xy} = -\frac{sin(\phi)}{cos(\phi)} &= -tan(\phi)
\end{aligned}
$$

$$
\begin{aligned}
h_{yx} * s_y &= sin(\phi) \\
h_{yx} * cos(\phi) &= sin(\phi) \\
h_{yx} = \frac{sin(\phi)}{cos(\phi)} &= tan(\phi)
\end{aligned}
$$

$$
\begin{aligned}
s_x + h_{yx} * h_{xy} * s_y &= cos(\phi) \\
s_x &= cos(\phi) - (tan(\phi) * (-tan(\phi)) * cos(\phi)) \\
&= cos(\phi) + tan(\phi) * tan(\phi) * cos(phi) \\
&= cos(\phi) + \frac{sin(\phi)}{cos(\phi)} * \frac{sin(\phi)}{cos(\phi)} * cos(\phi) \\
&= cos(\phi) + \frac{sin(\phi)}{cos(\phi)} * sin(\phi)
\end{aligned}
$$

9.4

a)

$$T = \begin{bmatrix} cos(\phi) & -sin(\phi) \\ sin(\phi) & cos(\phi) \end{bmatrix}$$

Insert $-\phi$ since it's the inverse rotation of the rotation with ϕ:

$$T^{-1} = \begin{bmatrix} cos(-\phi) & -sin(-\phi) \\ sin(-\phi) & cos(-\phi) \end{bmatrix} = \begin{bmatrix} cos(\phi) & sin(\phi) \\ -sin(\phi) & cos(\phi) \end{bmatrix} = T^T$$

With cosine's symmetry:

$$cos(-\phi) = cos(\phi)$$

With sinus's asymmetry:

$$
\begin{aligned}
sin(-\phi) &= -sin(\phi) \\
-sin(-\phi) &= sin(\phi)
\end{aligned}
$$

b)

$$\begin{bmatrix} \color{blue}{0} & \color{blue}{-1} & \color{green}{1} \\ \color{blue}{1} & \color{blue}{0} & \color{green}{1} \\ 0 & 0 & 1 \end{bmatrix}$$

The blue part of the matrix is a swap of the coordinates from (x, y) to $(y, -x)$.
The green part of the matrix marks the origin as $(1, 0)$.

Altogether the matrix is a function which rotates a point around $(1, 0)$ in 4 clockwise steps ($90°$ each), building a square around $(1, 0)$.

9.5

a)

Scanline

Based on the concept of coherence it is an incremental algorithm which uses the odd-even parity rule to determine if a point is inside a polygon. It processes one line at a time rather than one pixel at a time. All of the polygons to be rendered are first sorted by the top at which they first appear, then each scanline (row) of the image is computed using the intersection of a scanline with the polygons on the front of the sorted list, while the sorted list is updated to discard no-longer-visible polygons as the active scan line is advanced down the picture.
`https://en.wikipedia.org/wiki/Scanline_rendering`

Edge-flag

The edge-flag algorithm draws the contour of the polygon (roughly speaking: using the first intersection point of an edge with a pixel), then colors the pixels inside this contour using a bool variable (inside polygon or not), so if pixel is inside polygon then it will be colored, else it will be for example set to the background color.
`https://de.wikipedia.org/wiki/Rasterung_von_Polygonen`

b)

DDA

The Digital Differential Analyzer is an incremental algorithm, which means one variable is simply increased (e.g. $x_{i+1} = x_i + 1$) and the other calculated from the first (e.g. $y_{i+1} = m(x_i + 1) + B$). y_{i+1} then also has to be rounded up or down.
`https://medium.com/@thiagoluiz.nunes/rasterization-algorithms-computer-graphics-b9c3600a7`
`http://groups.csail.mit.edu/graphics/classes/6.837/F02/lectures/6.837-7_Line.`
`pdf`

Bresenham step