Kapitel 6

2D-Transformationen

Mit Hilfe von Transformationen ist es möglich, die Position, die Orientierung, die Form und die Größe der grafischen Objekte zu manipulieren. Transformationen eines Objekts werden durch Operationen auf den Definitionspunkten realisiert. Durch 2 Punkte definierte Rechtecke müssen zunächst zu Polygonen gemacht werden.

6.1 Translation

Gradlinige Verschiebung um einen Translationsvektor $T = (t_x, t_y)$, d.h. $P' := P + T$

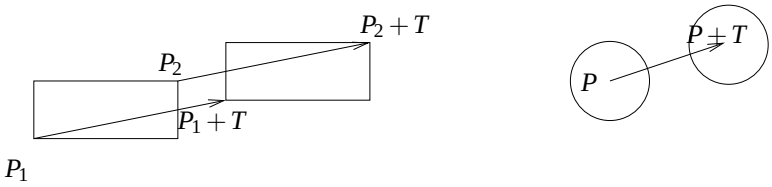


Abbildung 6.1: Translation von Rechtecken und Kreisen

6.2 Skalierung

Vergrößerung bzw. Verkleinerung bzgl. eines Fixpunktes. Zunächst liege der Fixpunkt im Ursprung $(0,0)$.

$$(x', y') \quad := \quad (s_x \cdot x, s_y \cdot y)$$

$s_x = s_y \Rightarrow$ uniforme Skalierung; $s_x \neq s_y \Rightarrow$ Verzerrung

Weder s_x noch s_y dürfen gleich 0 sein. Sonst würde das 2D-Objekt in einer Dimension (oder gar beiden Dimensionen) auf die Ausdehnung 0 zusammengestaucht. Die Objekte sind aber zweidimensional und sollen es im Laufe der Transformationen auch bleiben.

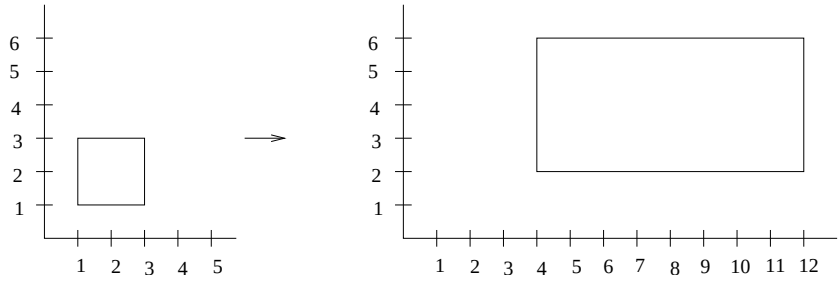


Abbildung 6.2: Skalierung mit $s_x = 4$ und $s_y = 2$

Abbildung 6.2 zeigt ein Beispiel für eine Skalierung bezüglich des Ursprungs mit den Faktoren $s_x = 4$, $s_y = 2$.

Bei Wahl eines beliebigen Fixpunktes (Z_x, Z_y) folgt für den Punkt P:

1. Translation um $(-Z_x, -Z_y)$ liefert P_1.
2. Skalierung mit (s_x, s_y) liefert P_2.
3. Translation um (Z_x, Z_y) liefert $P_3 = P'$.

Die neuen Koordiaten berechnen sich dann wie folgt:

$$\Rightarrow (x', y') := ((x - Z_x) \cdot s_x + Z_x, (y - Z_y) \cdot s_y + Z_y)$$

Vorteilhafter bei mehreren Objekten:

$$\begin{aligned} (x', y') &= (x \cdot s_x + d_x, y \cdot s_y + d_y) \\ &\quad \text{mit } d_x = Z_x \cdot (1 - s_x), d_y = Z_y \cdot (1 - s_y) \end{aligned}$$

Abbildung 6.3 zeigt ein Beispiel für eine Skalierung bezüglich des Punktes $Z = (1,3)$ mit den Faktoren $s_x = 3$, $s_y = 2$.

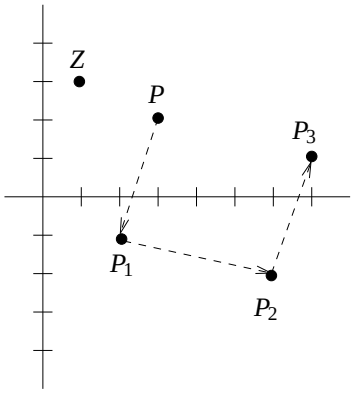
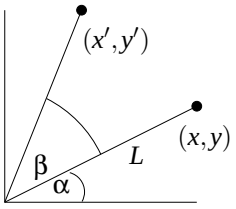


Abbildung 6.3: Skalierung bzgl. Punkt (1,3) mit Faktoren $s_x = 3$ und $s_y = 2$

6.3 Rotation

Drehung des Objekts bzgl. eines Fixpunktes um einen Winkel β. Der Fixpunkt liege im Ursprung.



$L = \sqrt{x^2 + y^2}$
$\sin(\alpha) = y/L$
$\cos(\alpha) = x/L$

Abbildung 6.4: Rotation um den Winkel β bzgl. des Ursprungs

$$
\begin{aligned}
\cos(\alpha+\beta) &= \cos(\beta)\cdot\cos(\alpha) - \sin(\beta)\cdot\sin(\alpha) \\
\sin(\alpha+\beta) &= \cos(\beta)\cdot\sin(\alpha) + \sin(\beta)\cdot\cos(\alpha) \\
\cos(\alpha+\beta) = x'/L &= \cos(\beta)\cdot\cos(\alpha) - \sin(\beta)\cdot\sin(\alpha) \\
\Rightarrow x' &= L\cdot\cos(\beta)\cdot\frac{x}{L} - \sin(\beta)\cdot\frac{y}{L}\cdot L \\
\Rightarrow x' &= x\cdot\cos(\beta) - y\cdot\sin(\beta) \\
\sin(\alpha+\beta) = y'/L &= \cos(\beta)\cdot\sin(\alpha) + \sin(\beta)\cdot\cos(\alpha) \\
\Rightarrow y' &= L\cdot\cos(\beta)\cdot\frac{y}{L} + L\cdot\sin(\beta)\cdot\frac{x}{L} \\
\Rightarrow y' &= x\cdot\sin(\beta) + y\cdot\cos(\beta)
\end{aligned}
$$

Positive Werte für den Drehwinkel β bewirken eine Rotation gegen den Uhrzeigersinn.
Vorsicht: Auf dem Bildschirm ist es andersherum, da die y-Achse nach unten zeigt.
Bei Wahl eines beliebigen Rotationszentrums (R_x, R_y) folgt für den Punkt P

1. Translation um $(-R_x, -R_y)$ liefert P_1
2. Rotation bzgl. Ursprung um Winkel β liefert P_2
3. Translation um (R_x, R_y) liefert $P_3 = P'$

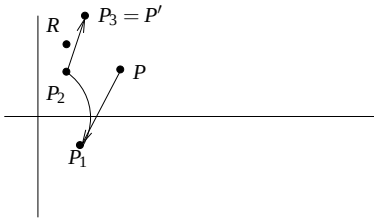


Abbildung 6.5: Rotation des Punktes P um 90 Grad bzgl Punkt R

```
/*****************************************************************************/
/*                                                                           */
/*                     Skalieren und Rotieren eines Polygons                 */
/*                     Obacht: Da die Polygon-Koordinaten ganzzahlig sind,   */
/*                     verstaerken sich die Rundungsfehler nach jeder Transformation */
/*                                                                           */
/*****************************************************************************/

private static final double DEG_TO_RAD = Math.PI/180.0; // Umrechnung Grad in Bogenmass


private void skalier_polygon(           // skaliert ein Polygon
                        Point[] points, // gespeichert im Array points
                        int n,          // mit n Eckpunkten
                        double sx,      // in x-Richtung um den Faktor sx
                        double sy,      // in y-Richtung um den Faktor sy
                        Point  z)       // bezueglich des Punktes z
{
        int i;
        for (i=0; i< n; i++) {

                points[i].x = (int)((points[i].x - z.x ) * sx + z.x + 0.5);
                points[i].y = (int)((points[i].y - z.y ) * sy + z.y + 0.5);

        }
}


private void rotate_polygon(            // rotiert ein Polygon
                        Point[] points, // gespeichert im Array points
                        int   n,        // mit n Eckpunkten
                        int   g,        // um einen Winkel von g Grad
                        Point z)        // bezueglich des Punktes z
{
        int i;
double beta = g * DEG_TO_RAD;   // Winkel im Bogenmass
        Point P = new Point();

        for (i=0; i < n; i++) {

                P.x = (int)((points[i].x - z.x ) * Math.cos( beta ) -
                      (points[i].y - z.y ) * Math.sin( beta ) + z.x + 0.5);

                P.y = (int)((points[i].y - z.y ) * Math.cos( beta ) +
                      (points[i].x - z.x ) * Math.sin( beta ) + z.y + 0.5);

                points[i].x = P.x;
                points[i].y = P.y;

        }
}
```

6.4 Matrixdarstellung

Häufig werden mehrere Transformationen hintereinander auf ein Objekt angewendet. Es entstehen Rundungsfehler, wenn nach jeder Einzel-Transformation die ganzzahligen Koordinaten bestimmt werden. Deshalb sollten mehrere Transformationen zu einer zusammengesetzt werden. Dies ist mit Hilfe von Matrizen, die die Transformationen repräsentieren, möglich. Durch Matrizenmultiplikation ($A \cdot B = C$) der Transformationsmatrix A und dem als einspaltige Matrix interpretierten Koordinatenvektor B wird die eigentliche Transformation durchgeführt. Die einzelnen Elemente c_{ik} der Ergebnismatrix C errechnen sich aus den Elementen der Eingangsmatrizen A und B wie folgt:

$$c_{ik} = \sum_{j=0}^{n} a_{ij} \cdot b_{jk}$$

Eine Skalierung sieht dann z.B. folgendermaßen aus:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} := \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x \cdot s_x \\ y \cdot s_y \end{pmatrix}$$

Eine Rotation hat folgendes Aussehen:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} := \begin{pmatrix} \cos(\beta) & -\sin(\beta) \\ \sin(\beta) & \cos(\beta) \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x \cdot \cos(\beta) - y \cdot \sin(\beta) \\ x \cdot \sin(\beta) + y \cdot \cos(\beta) \end{pmatrix}$$

Die Reihenfolge von Matrix und Vektor kann vertauscht werden. Allerdings müssen dann beide transponiert werden:

$$A \cdot B = (B^T \cdot A^T)^T$$

6.5 Homogene Koordinaten

Um auch die Translation durch eine Matrixmultiplikation ausdrücken zu können, muß das Konzept der *homogenen Koordinaten* eingeführt werden. Dabei wird unserem Objektraum eine Dimension hinzugefügt, es werden aber weiterhin 2D-Objekte repräsentiert und dargestellt.

Ein Punkt $P = (x,y)$ hat die homogenen Koordinaten $(x_h\ y_h\ w)^T$ mit $w \neq 0$ und

$$\begin{aligned} x_h &= x \cdot w \\ y_h &= y \cdot w \end{aligned}$$

Beispiel: Das homogene Koordinatentripel $(6\ 8\ 2)^T$ gehört zum Punkt $P = (3,4)$. Derselbe Punkt hat die homogenen Koordinaten $(3\ 4\ 1)^T$. D.h. jeder Punkt $P = (x,y)$ repräsentiert eine ganze Ursprungsgerade im 3D-Raum.

Der Richtungsvektor $\vec{r}$, der vom Ursprung zum Punkt $R = (x,y)$ führt, hat die homogenen Koordinaten $(x\ y\ 0)^T$.

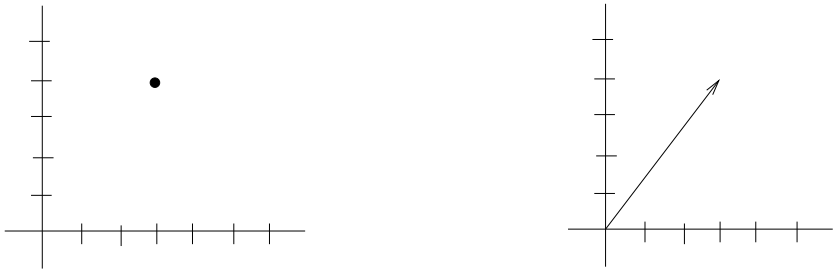


Abbildung 6.6: Punkt $(3,4)$ und Richtungsvektor $(3,4)^T$

Die Transformationen Translation, Skalierung und Rotation werden nun als 3×3-Matrizen realisiert. Zusammengesetzte Transformationen ergeben sich durch Matrix-Multiplikation.

Translation

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} := \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x+t_x \\ y+t_y \\ 1 \end{pmatrix}$$

Skalierung

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} := \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x \cdot s_x \\ y \cdot s_y \\ 1 \end{pmatrix}$$

Rotation

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} := \begin{pmatrix} \cos(\beta) & -\sin(\beta) & 0 \\ \sin(\beta) & \cos(\beta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x \cdot \cos(\beta) - y \cdot \sin(\beta) \\ x \cdot \sin(\beta) + y \cdot \cos(\beta) \\ 1 \end{pmatrix}$$

Für die Auswertung einer zusammengesetzten Transformation

- gilt das Assoziativgesetz, d.h. $A \cdot B \cdot C = (A \cdot B) \cdot C = A \cdot (B \cdot C)$,
- gilt im allgemeinen nicht das Kommutativgesetz, d.h. $A \cdot B \neq B \cdot A$.
  Z.B. ist eine Translation um $(5,8)$ mit anschließender Rotation um 90° verschieden von Rotation um 90° mit anschließender Translation um $(5,8)$.

Beispiel für zusammengesetzte Transformation: Rotation bzgl. $(3,5)$ um 60°

Matrix für Translation um $(-3,-5)$ lautet

$$A = \begin{pmatrix} 1.0000000 & 0.0000000 & -3.0000000 \\ 0.0000000 & 1.0000000 & -5.0000000 \\ 0.0000000 & 0.0000000 & 1.0000000 \end{pmatrix}$$

Matrix für Rotation um 60° lautet

$$B = \begin{pmatrix} 0.5000000 & -0.8660254 & 0.0000000 \\ 0.8660254 & 0.5000000 & 0.0000000 \\ 0.0000000 & 0.0000000 & 1.0000000 \end{pmatrix}$$

Matrix für Translation um $(3,5)$ lautet

$$C = \begin{pmatrix} 1.0000000 & 0.0000000 & 3.0000000 \\ 0.0000000 & 1.0000000 & 5.0000000 \\ 0.0000000 & 0.0000000 & 1.0000000 \end{pmatrix}$$

Matrix für gesamte Transformation lautet

$$D = C \cdot B \cdot A = \begin{pmatrix} 0.5000000 & -0.8660254 & 5.8301270 \\ 0.8660254 & 0.5000000 & -0.0980762 \\ 0.0000000 & 0.0000000 & 1.0000000 \end{pmatrix}$$

6.6 Allgemeine Transformationen

Weitere Transformationen, die sich durch Matrizen darstellen lassen, sind

- Spiegelung an einer beliebigen Geraden,
- Spiegelung an einem Punkt,
- Scherung.

Bei der Scherung in x bleiben die y-Werte konstant, und die x-Werte werden proportional zu den y-Werten horizontal verschoben, d.h. $x' = x + Sch_x \cdot y$.
Die Transformationsmatrix lautet:

$$\begin{pmatrix} 1 & Sch_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{ bzw. } \begin{pmatrix} 1 & 0 & 0 \\ Sch_y & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

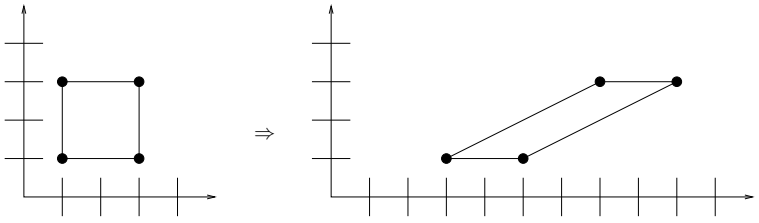


Abbildung 6.7: Scherung mit $Sch_x = 2$

6.7 Raster-Transformationen

Es werden nicht die Definitionspunkte, sondern die Pixel im Frame-Buffer modifiziert:

- Verschieben eines Fensterinhalts (Move),
- Kopieren eines Fensterinhalts (Copy),
- Vergrößern eines Fensterinhalts (Zoom).

Statt Pixelwerte zu setzen, können auch AND-, OR-, XOR-Verknüpfungen verwendet werden. Es gilt $(x \text{ XOR } y) \text{ XOR } y = x$. XOR ist daher geeignet, ein Objekt über den Schirm zu bewegen, ohne den Hintergrund zu zerstören.

```
zeichne Objekt mit XOR
repeat
   loesche Objekt mit XOR
   modifiziere Objekt_Koordinaten
   zeichne Objekt mit XOR
until Objekt am Ziel
```

Allerdings ändert das bewegte Objekt je nach Hintergrundfarbe sein Aussehen.

6.8 Java-Applet zu 2D-Transformationen

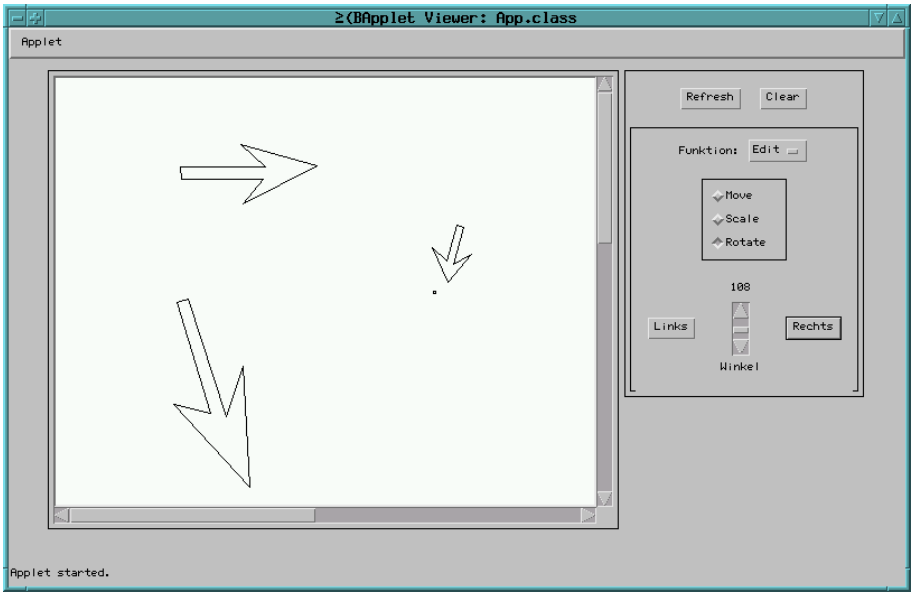


Abbildung 6.8: Screenshot vom 2D-Trafo-Applet