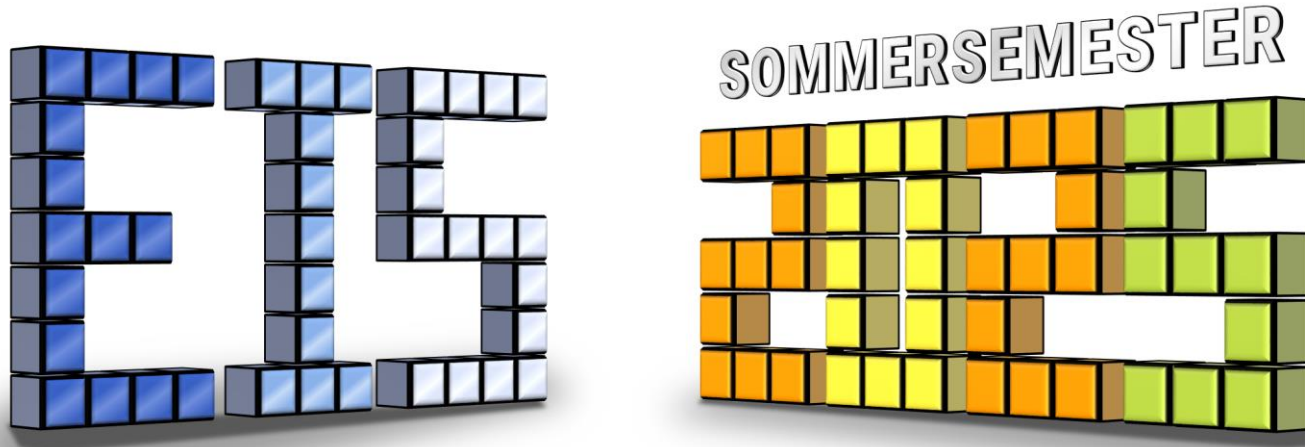


EINFÜHRUNG IN DIE SOFTWAREENTWICKLUNG

Sommersemester 2025



Foliensatz #5

Beispielprojekt

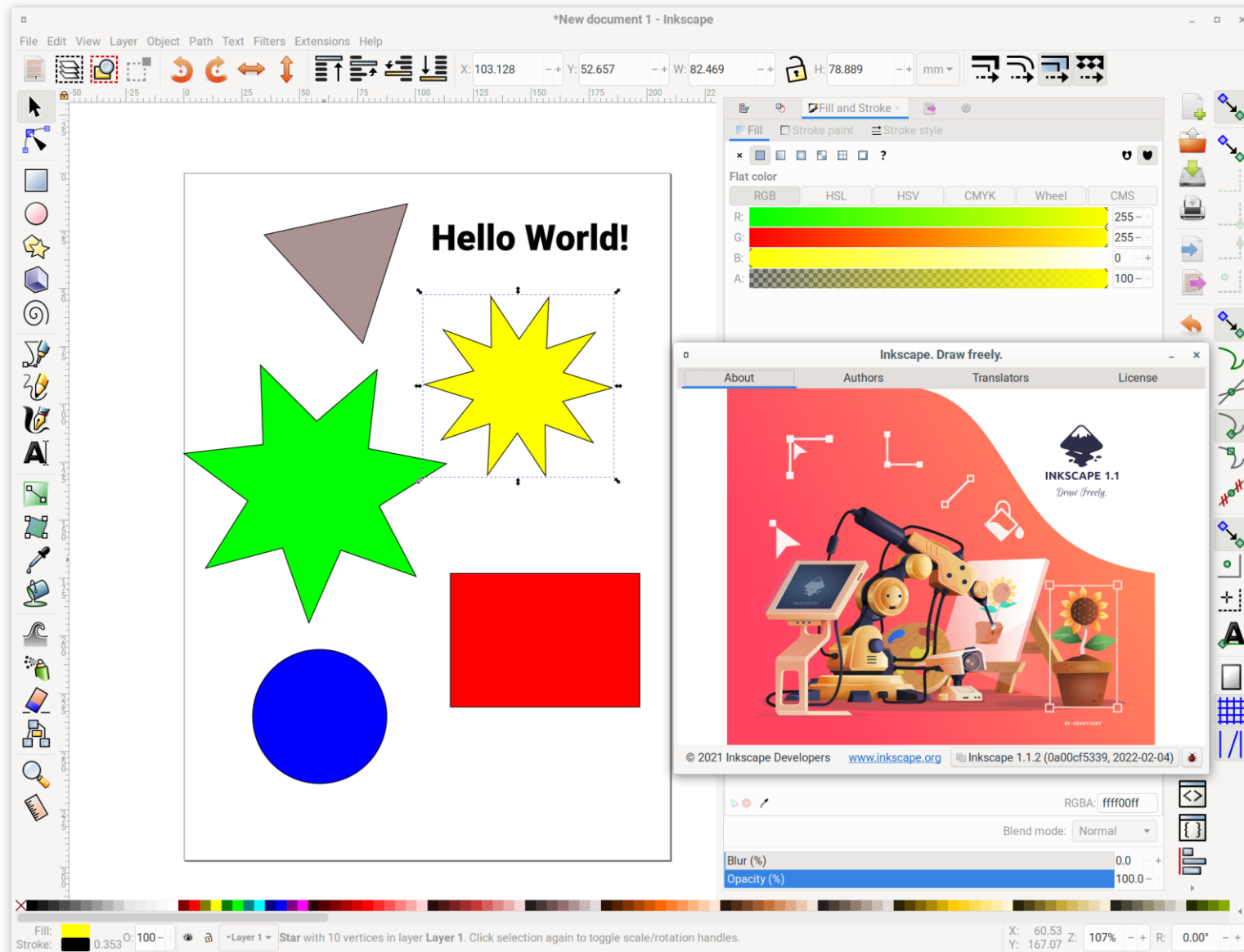
Michael Wand
Institut für Informatik
Michael.Wand@uni-mainz.de



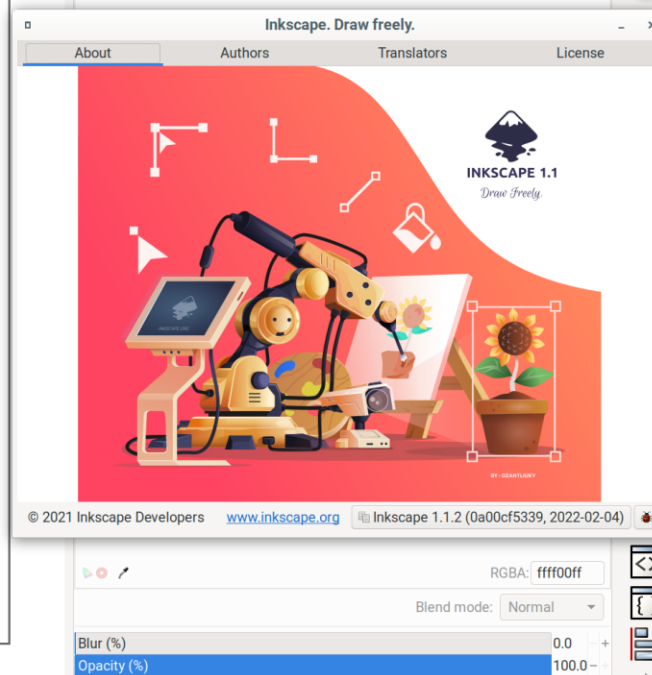
Beispielprojekt

Vektorgraphik-Editor

Zeichenprogramm für Vektorgraphik



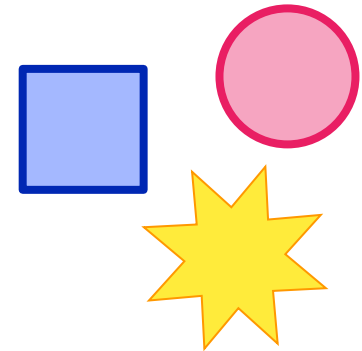
Beispiel:
Inkscape
(open source/GPL)



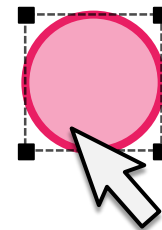
EIS-Version

Aspekte des Projektes

- Vektorgraphik
 - Verschiedene Typen von Vektorobjekten
 - Kreise, Rechtecke, Sterne, etc.
 - Verschiedene Attribute
 - Farben, Umrandungen, Anzahl Ecken der Sterne
 - Ausbau zu GUI-Bibliothek
 - GUI-Library design: Konzeptionell (Vorlesung)
 - In Übungen: GUI mit Qt (fertige GUI-Lib)



- Interaktion
 - Manipulation mit der Maus
 - Technisch durchaus anspruchsvoll



Drei Schritte (Planung)

Schritt 0

- GUI-Programmieren lernen mit Qt (oder AWT)

Schritt 1

- Vektorgraphik Bibliothek prozedural

Schritt 2

- Implementation mit OOP (einfacher)
- GUI-Elemente hinzufügen

Schritt 3

- Funktionale / Datenflussarchitektur

Motivation

Warum dieses Projekt?

- Komplexes Problem
- Führt Design und Anwendung von GUIs mit ein
- Historisch motivierend für OOP
 - Xerox Alto Projekt / Smalltalk / erstes „moderne“ GUI
- Vor- und Nachteile von OOP & FP Pattern gut sichtbar
 - Motivierend für moderne Ansätze
- Relevant bis heute
 - Interaktive Anwendungen, Webframeworks, Games
 - Fortgeschrittenes: Persistenz, automatische GUIs, etc.
- Macht Spaß, weil bunt

„Anforderungsanalyse“

(sehr knapp & informell)

Anforderungen

Vektorgraphik

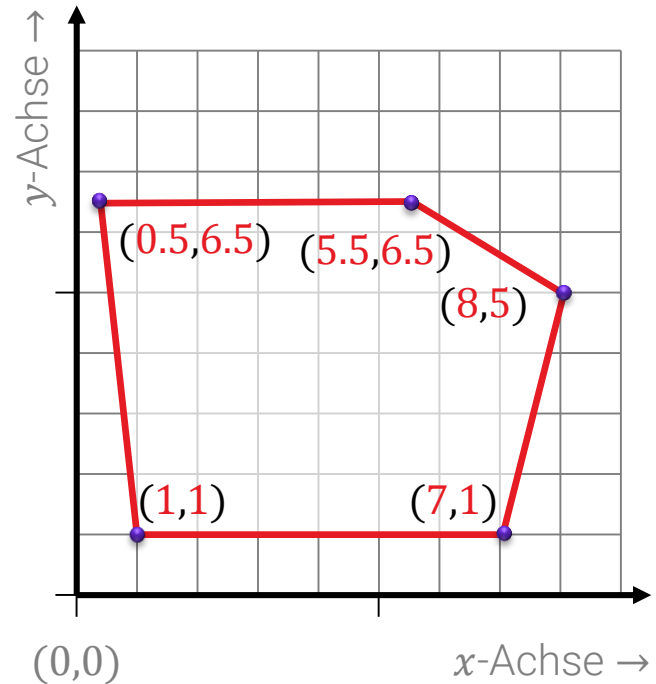
- Geometrische Primitive, z.B.
 - Linien
 - Rechtecke, Kreise
 - Gruppierung („Szenengraph“)
 - (Text, Bitmap-Bilder) *optional (nicht in allen in EIS)*
- Repräsentation als mathematische Beschreibung
 - „Vektorgraphik“ (kein Auflösungslimit)
 - Umwandlung in Rastergraphik bei Bedarf
- Operationen
 - Erzeugen + Transformieren: verschieben (drehen, skalieren)

Einschub: 2D Vektorgraphik

Vektorgraphik

Grundprimitiv: Linien

- Eckpunkte gespeichert als Koordinaten
(x : float, y : float)



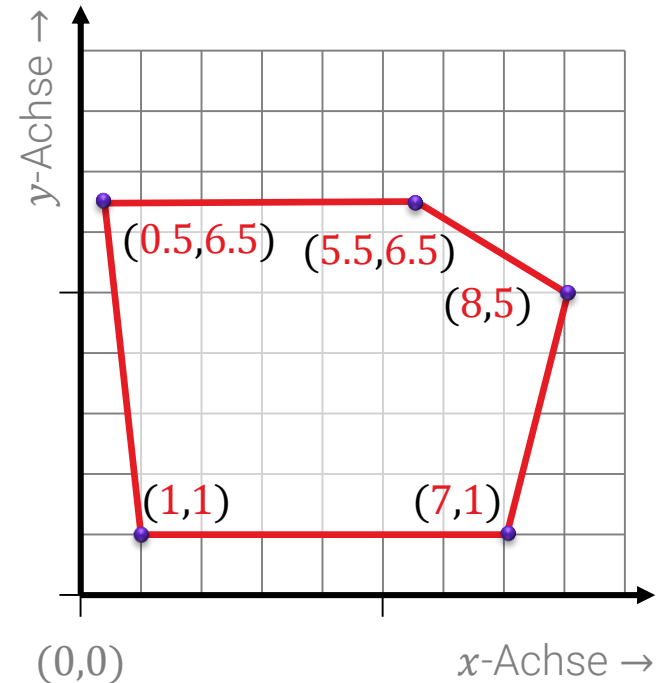
Vektorgraphik

Grundprimitiv: Linien

- Eckpunkte gespeichert als Koordinaten
(x : float, y : float)

Umsetzung:

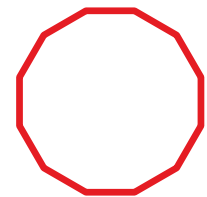
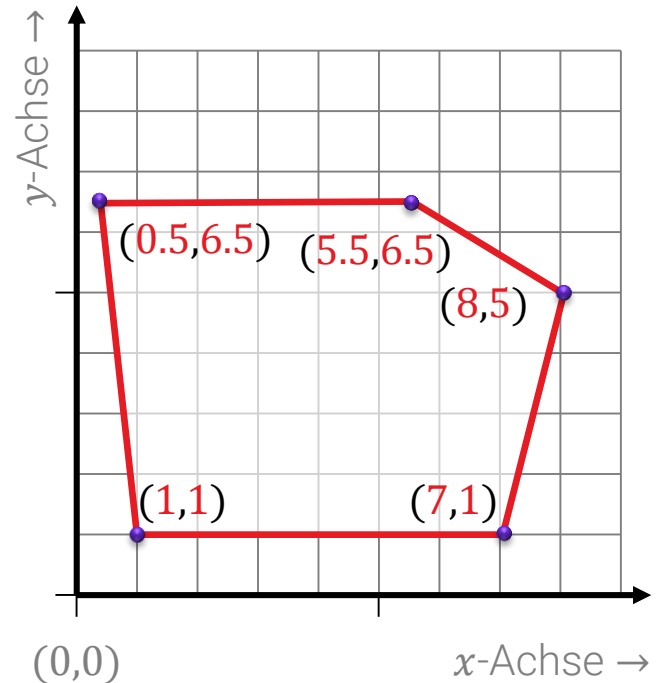
- Vektoren als Tupel in Python
`Tuple[float, float]`
- Vektoren mit NumPy
- Qt Klassen `QVector2D`,
`QTransform2D` (empfohlen)



Vektorgraphik

Grundprimitiv: Linien

- Eckpunkte gespeichert als Koordinaten
(x : float, y : float)
- Polygonzüge
 - Gefüllt
 - Ungefüllt
- Andere Primitive können (notfalls) mit Polygonen angenähert werden
 - Rechtecke
 - Kreise
 - ...



Kreis approx.
durch Polygon

In der Praxis

Wir nutzen eine 2D Graphikbibliothek

- In den Übungen:
 - **QPainter** (Qt for Python)
 - **Graphics** (Java/Scala AWT)
- Vorgefertigte Kommandos für
 - Zeichnen von Linien, Polygonen (gefüllt und ungefüllt)
 - Zeichnen von Kreisen, Ellipsen, Rechtecken, etc.
 - Zeichnen von Text („Font-Rendering“)
 - Zeichnen von Bitmap-Bildern in Zeichenfläche
- Zeichenflächen: Anwenden von QPainter auf...
 - **QImage**: Offline-Bild (Java: **BufferedImage**)
 - **QWidget**: Online, direkt sichtbar (Ü. 02; Java: **Window**)

„Viewport Transformation“

Welt
(0,0) →

Welt
(*viewLeft*,
viewTop) \cong Pixel
(0,0) → $\text{pixelWidth} \cong \text{viewWidth}$

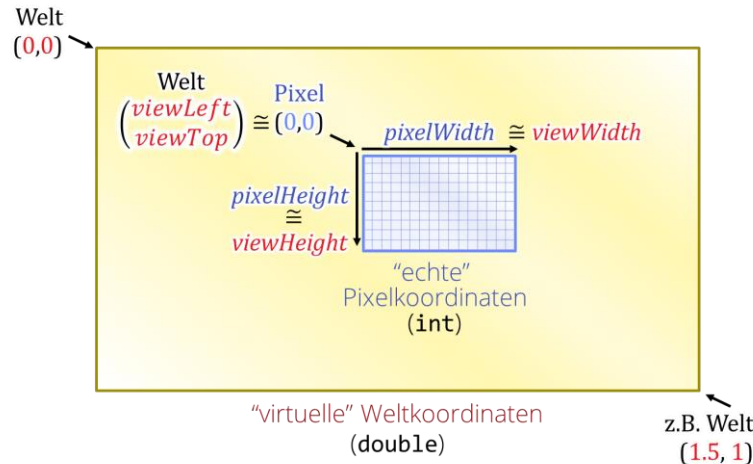
$\text{pixelHeight} \cong \text{viewHeight}$

“echte”
Pixelkoordinaten
(int)

“virtuelle” Weltkoordinaten
(double)

↖ z.B. Welt
(1.5, 1)

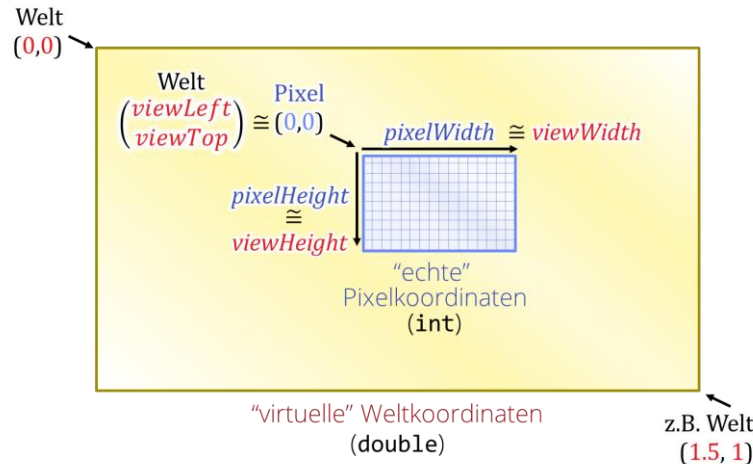
„Viewport Transformation“



Berechnung Viewport-Transformation

- Lege Weltkoordinatensystem fest, z.B.
 - Notation: Ausgabefenster b_{pix} Pixel breit, h_{pix} Pixel hoch
 - Unten links = $(0,0)$
 - Oben links = $(0,1)$
 - Unten rechts = $(0, b_{pix}/h_{pix})$ (also unverzerrt)

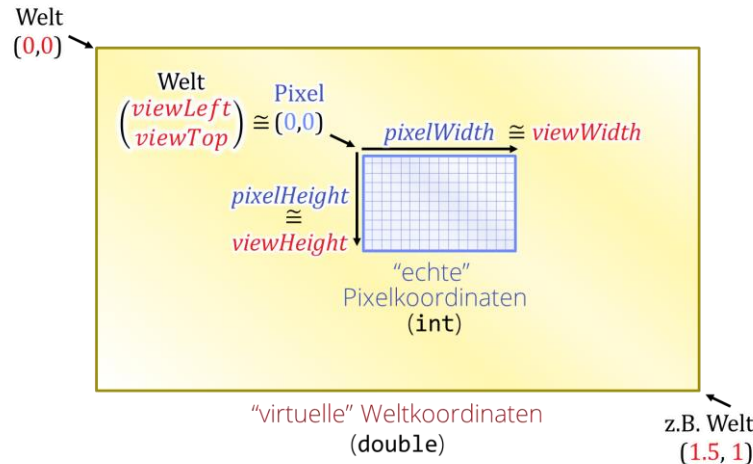
„Viewport Transformation“



Berechnung Viewport-Transformation

- Weltkoordinaten: y -Achse $(0,1)$, x -Achse $(0, b_{pix}/h_{pix})$
- Multiplikation mit Pixeln: Koordinaten $\times h_{pix}$
- Spiegeln: Pixelkoordinaten starten links oben
 - y -Werte $\rightarrow h_{pix} - y$

„Viewport Transformation“



Ausschnitt verschieben

- Verschieben
 - Werte auf x,y addieren
- Zoomen / Vergrößern/Verkleinern
 - Koordinaten mit Faktor multiplizieren
 - Achtung: Mittelpunkt ist Ursprung (0,0)
 - Allgemein: Ursprung abziehen, zoomen, wieder addieren

Weitere Anforderungen...

Orthogonale Anforderungen

Für alle Objekte

- Editieren mit der Maus
 - Erzeugen, Löschen
 - Transformieren
- Editieren per Tastatur
 - Eigenschaften numerisch im UI setzen
 - (Beispiel für Metaprogrammierung)
- Speichern und Laden

Anwendungsprogramm

„Klassische“ Anwendung

- Dokumentenorientiert
 - Bez. Speichern / Laden
 - (Evtl. Multi-Dokument GUI)
- Drop-Down-Menüs für wesentliche Funktionen
- Toolbars & Shortcuts für schnellen Zugriff
 - Qt liefert das ohne viel extra Aufwand
- „Inspector“ für numerisches Editieren

Weitere Anforderungen

Erweiterbarkeit

- Neue Primitive
 - Text, Sterne, Polygone o.ä.
- Neue Werkzeuge zum Editieren
 - z.B. „Align-Left“, Replicate along Curve, etc.
- Dynamisches Laden von Plugins
 - Probleme bei Fehlerbehandlung, Laden/Speichern, Darstellung absehbar – wir werden Lösungen dafür sehen
- Grundlegende / Unvorhergesehene Erweiterungen
 - Z.B. Mehrere Seiten/Folien / Präsentationsmodus
 - Unerwartete Erweiterungen hier schwierig zu planen!

Grobe Struktur

(Hier startet der Entwurf...)

Komponenten

Bestandteile / wesentliche Module

- Vektorgraphik-Bibliothek
 - Zeichenprimitive anlegen
 - Transformieren
 - Umwandlung in Rastergraphik
 - Szenengrapharchitektur
 - Persistenz (Laden + Speichern)
- Editor
 - Operationen und Transformationen interaktiv durchführen
- Anwendungsrahmen
 - Fenster, Menus etc.

Wie packen wir es an?

Bestandteile

- Vektorgraphik-Bibliothek
 - prozedurales Design
 - OOP-Design
 - Dynamische Metaprogrammierung
 - funktionales Design
- Editor
 - Interaktion strukturieren
 - Erweiterung zu allgemeinen GUIs
 - Außerhalb „Spezifikation“, aber passt hier am besten :-)
- Anwendungsrahmen
 - Knapp / in den Übungen

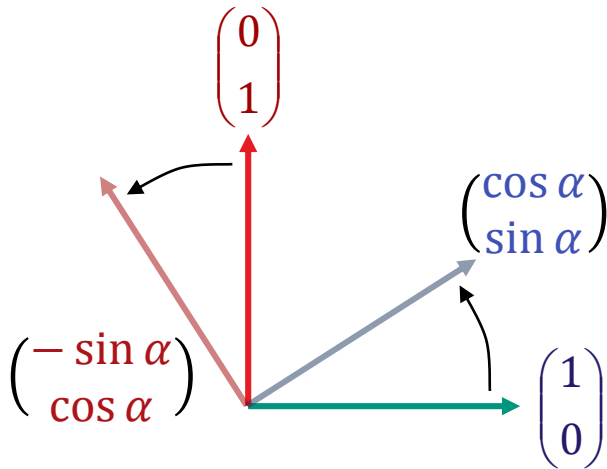
Addendum:

Vektorrechnung & Lineare Abbildungen

(Was kann QTransform2D,
und was passiert hinter den Kulissen?)

Koordinatensystem wechseln

Beispiel: Rotationsmatrix



$$\mathbf{M}_{rot} = \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix}$$

Matrizen

Matrix-Vektor-Produkt

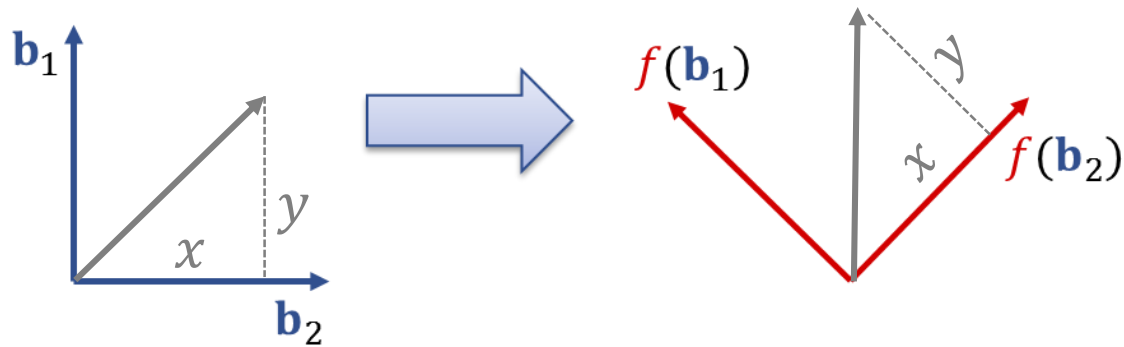
$$\mathbf{y}(\boldsymbol{\lambda}) = \begin{bmatrix} | & & | \\ \mathbf{x}_1 & \cdots & \mathbf{x}_n \\ | & & | \end{bmatrix} \cdot \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_n \end{bmatrix}$$

Konstruktion

- Abbildung *n-dim Vektoren* \rightarrow *n-dim Vektoren*
 - $\boldsymbol{\lambda} \in \mathbb{R}^n$
 - $\mathbf{x}_i \in \mathbb{R}^n \Rightarrow \mathbf{y} \in \mathbb{R}^n$
- Spalten der Matrix \mathbf{X}
= Bilder der Einheitsvektoren (Achsen)

Matrix-Vektor Produkt

Ausrechnen

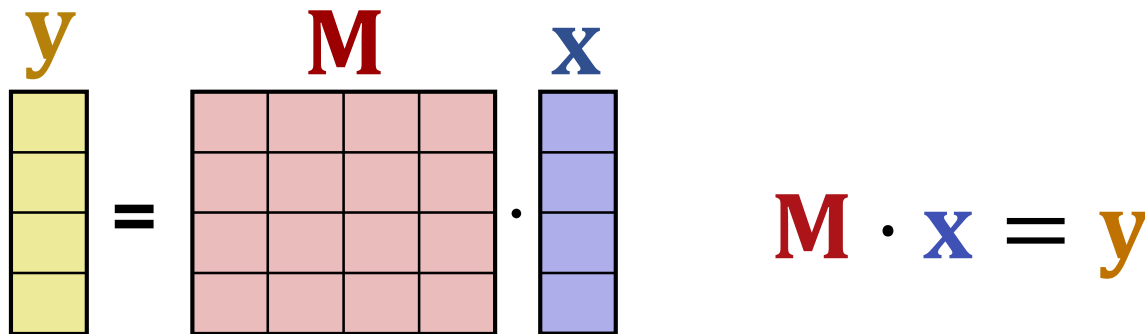


$$f(x, y) = x \cdot f(b_1) + y \cdot f(b_2)$$

Algebra

Algebraisch

- Vektor-Matrix Produkt:



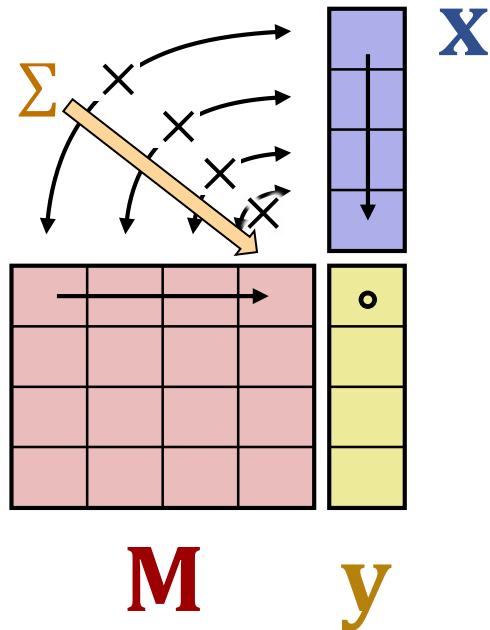
The diagram illustrates the vector-matrix product $\mathbf{M} \cdot \mathbf{x} = \mathbf{y}$. On the left, a yellow vertical vector \mathbf{y} with 4 cells is shown. To its right is an equals sign. Next is a red 4x4 matrix \mathbf{M} , represented as a grid of 16 pink squares. To the right of the matrix is a blue vertical vector \mathbf{x} with 4 cells. A dot operator \cdot is placed between the matrix and the vector. To the right of this entire expression is another equals sign, followed by the yellow vector \mathbf{y} . To the right of the diagram, the equation $\mathbf{M} \cdot \mathbf{x} = \mathbf{y}$ is written in red, blue, and yellow text respectively.

$$\mathbf{y} = \mathbf{M} \cdot \mathbf{x}$$
$$\mathbf{M} \cdot \mathbf{x} = \mathbf{y}$$

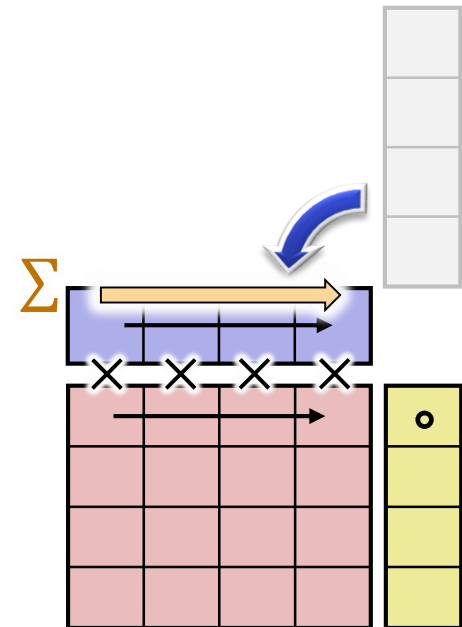
Allg. Matrix Produkt (Notation)

Algebraische Regel:

- Vektor-Matrix Produkt:



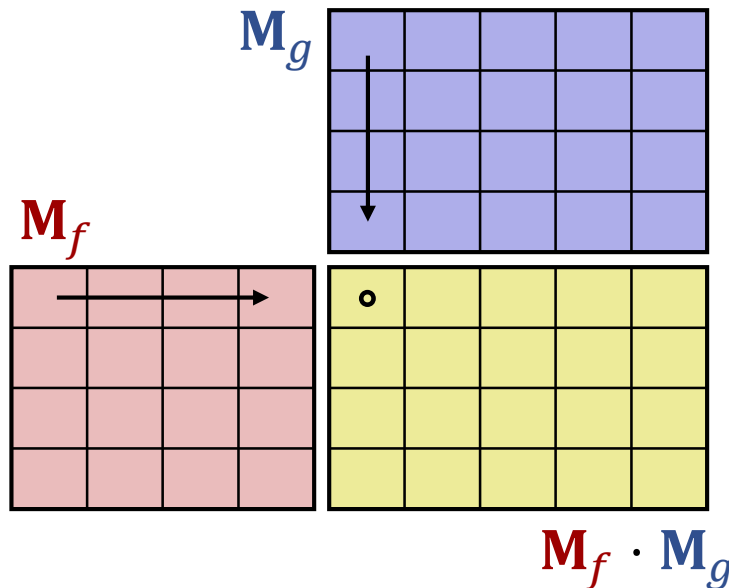
$$\mathbf{M} \cdot \mathbf{x} = \mathbf{y}$$



Matrix Multiplikation

Hintereinanderausführung von Matrizen

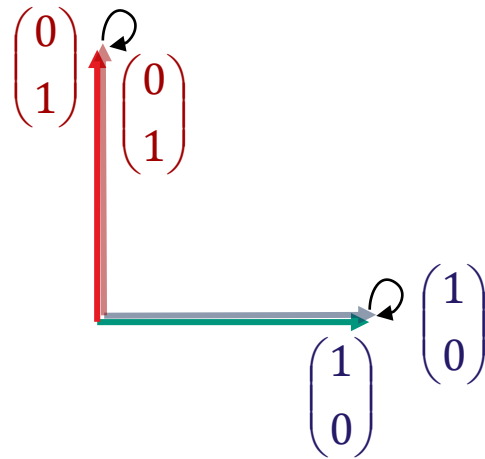
- $f(g) = f \circ g = \mathbf{M}_f \cdot \mathbf{M}_g$
- Berechnung



Der (i, j) -te Eintrag ist das Skalarprodukt von Zeile i von \mathbf{M}_f mit Spalte j von \mathbf{M}_g

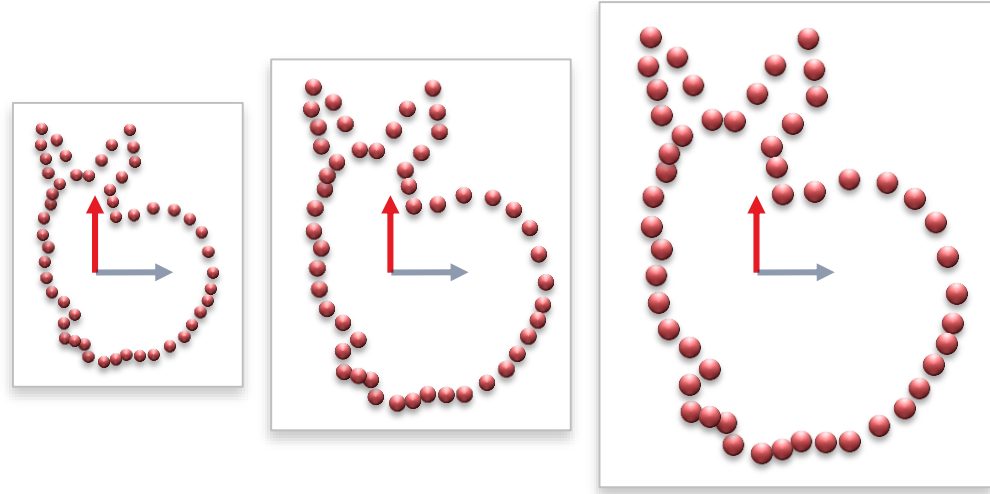
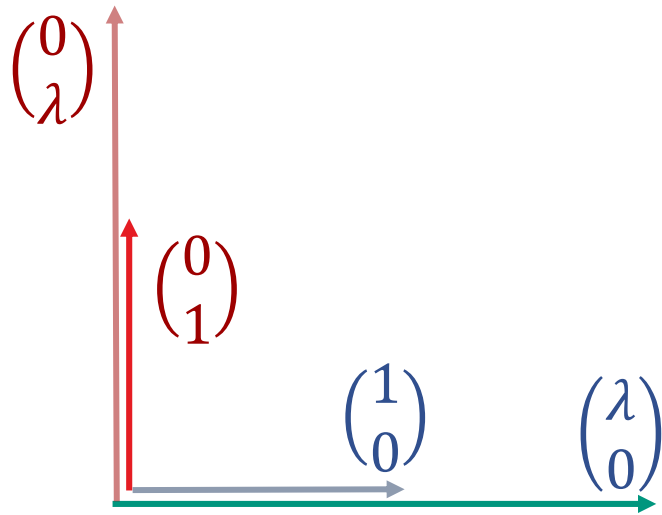
Identitätstransformation

Einheitsmatrix (nix passiert)



$$\mathbf{M}_{identity} = \mathbf{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Skalierung (Zentrum = Ursprung)

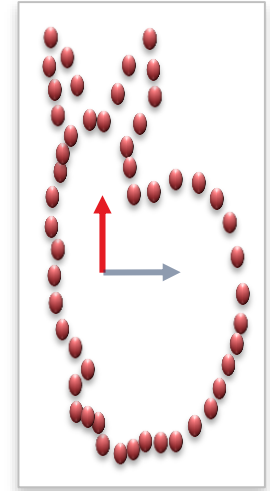
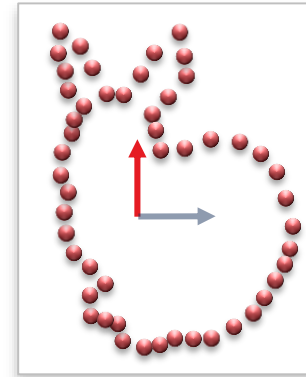
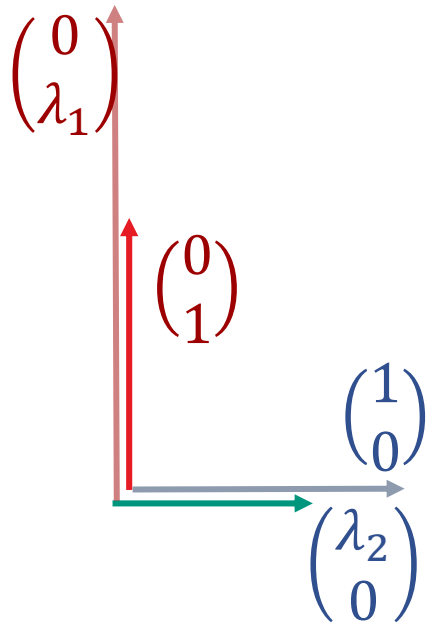


Matrix

$$\mathbf{S}_\lambda: \mathbb{R}^n \rightarrow \mathbb{R}^n,$$

$$\mathbf{S}_\lambda = \begin{bmatrix} \lambda & 0 & \cdots & 0 \\ 0 & \lambda & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda \end{bmatrix}$$

Nicht-uniforme Skalierung

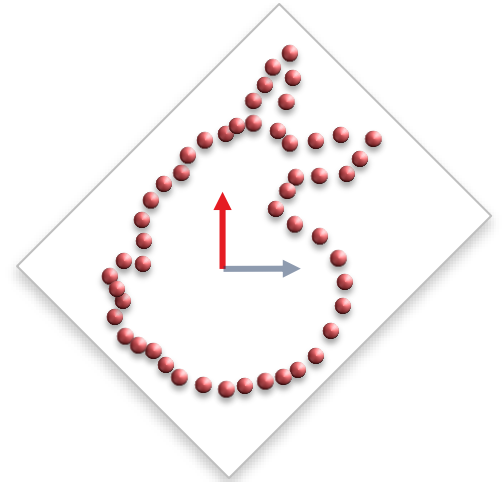
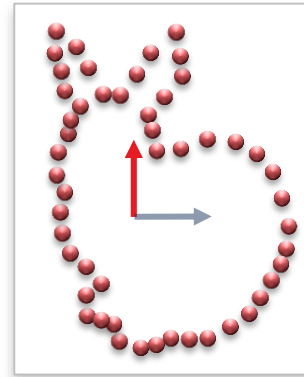
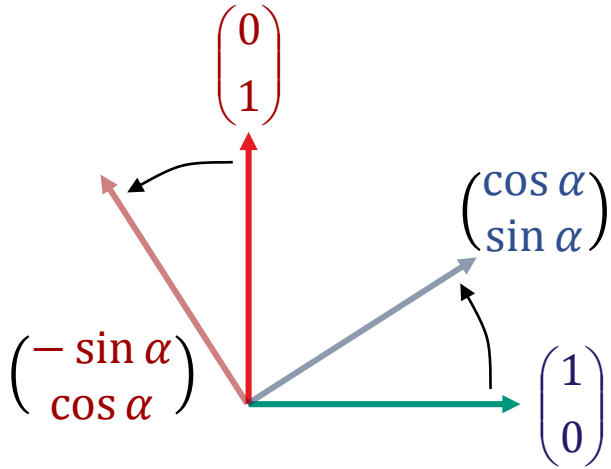


Matrix

$$S_{\lambda}: \mathbb{R}^n \rightarrow \mathbb{R}^n,$$

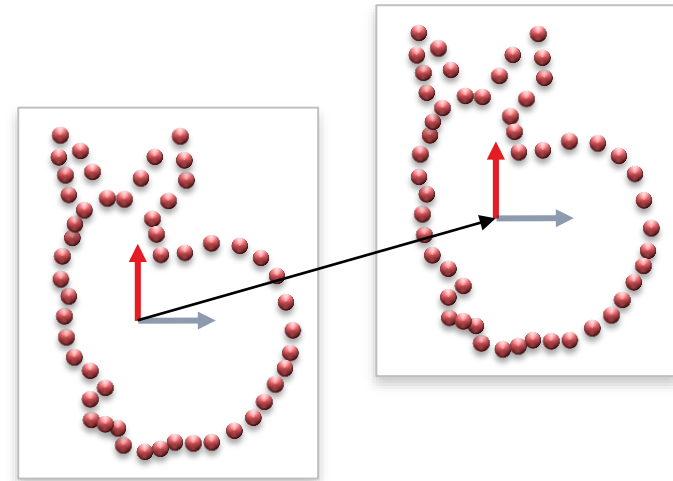
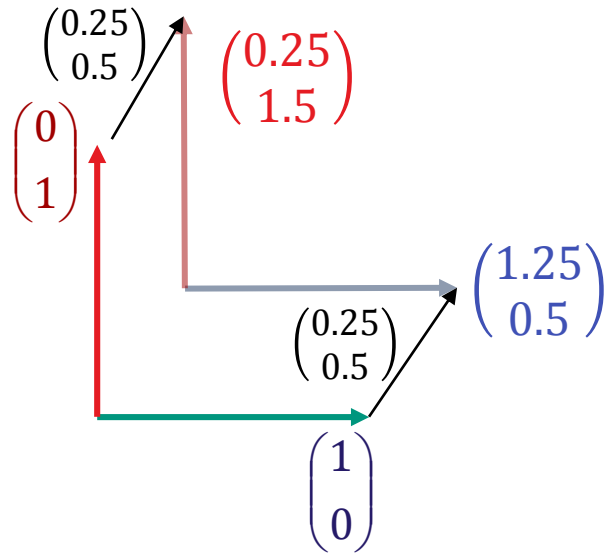
$$S_{\lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_3 \end{bmatrix}$$

Rotation



$$\mathbf{M}_{rot} = \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix}$$

Verschiebung



Verschiebung

- Addieren auf x- und y-Achse
- Nicht als Matrix darstellbar

Transformation:
Allgemeines Rezept

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

(**QTransform** erledigt das automatisch)