Informatik–Biber

Lehrmittel für die informatische Bildung an der Sekundarstufe I

Kommentar für Lehrpersonen

Inhaltsverzeichnis

Auszeichnungssprachen – Wie lassen sich digitale Inhalte beschreiben? 3

Umsetzungsvorschläge (bis 4 Lektionen) 4

Hintergrundwissen zu „Auszeichnungssprachen“ 5

Wozu Auszeichnungssprachen? 5
Wie ist eine Auszeichnungssprache aufgebaut? 6
Ein Beispiel für eine Auszeichnungssprache und deren Aufbau 6
Was ist SVG? 7
Wie sieht die Auszeichnungssprache „SVG“ aus? 8
Hintergrundwissen „Rastergrafiken vs. Vektorgrafiken“ 9
Warum hat das alte Band-Logo eine schlechte Qualität? 9
Was kann die Vektorgrafik, was die Rastergrafik nicht kann? 13
Reflexion Lernfilm / Perspektiven der Lernenden 17
Experiment 1: „Wie lässt sich ein Logo aus farbigen Kreisen bauen?“ 17
Experiment 2: „Wie lässt sich ein Logo verschieben, drehen oder verzerren?“ 17
Experiment 3: „Wie bringe ich Bewegung in ein Logo?“ 18

Umsetzungshilfen 16

Informatik-Biberaufgaben zum Thema Auszeichnungssprachen 19

Mögliche Vertiefungen 20

Weiterführende Literatur 20

Lösungen 21

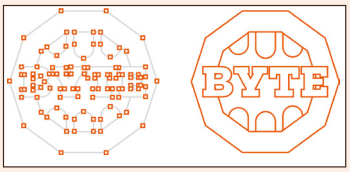
Experiment 1: „Wie lässt sich ein Logo aus farbigen Kreisen bauen?“ 21
Experiment 2: „Wie lässt sich ein Logo verschieben, drehen oder verzerren?“ 23
Experiment 3: „Wie bringe ich Bewegung in ein Logo?“ 24

Auszeichnungssprachen – Wie lassen sich digitale Inhalte beschreiben?

Ausgangskompetenzen

- Schülerinnen und Schüler kennen den Zweck von Auszeichnungssprachen in der Informatik und kennen mit dem SVG-Format Beispiele für den Einsatz einer Auszeichnungssprache.
- Schülerinnen und Schüler können in einem Textfenster oder Editor einfache SVG-Elemente erstellen und im Browser darstellen.
- Schülerinnen und Schüler kennen den Unterschied zwischen Raster- und Vektorgrafiken und können Vorteile von Vektorgrafiken benennen.

Umsetzungsvorschläge (bis 4 Lektionen)

Lernenden-Perspektiven	Anschauen des Lernfilms "Auszeichnungssprachen – Wie lassen sich digitale Inhalte beschreiben?" Thema: Auszeichnungssprachen (Markup Languages) Hast du Lust, die Möglichkeiten von Auszeichnungssprachen kennenzulernen und selber ein Logo zu animieren?
Experiment 1	Wie lässt sich ein Logo aus farbigen Kreisen bauen? Bezug Lernfilm: Vektorgrafiken und SVG 02:10 
Experiment 2	Wie lässt sich ein Logo verschieben, drehen oder verzerren? Bezug Lernfilm: geometrische Formen 02:15 
Experiment 3	Wie bringe ich Bewegung in ein Logo? Bezug Lernfilm: Animationen 02:30 
Biberaufgaben	Verschiedene Aufgaben zum Thema Auszeichnungssprachen (Markup Languages) selbstständig lösen.
Mögliche Vertiefung	Einführung in HTML: http://clab2.phbern.ch/script/html/html.htm Digitale Bildformate: http://www.swisseduc.ch/informatik/anwendungen/grafikformate/index.html

Hintergrundwissen zu „Auszeichnungssprachen“

Es ist Ihnen bestimmt schon einmal aufgefallen, dass dieselbe Webseite je nach Endgerät anders dargestellt wird. Haben Sie sich schon einmal überlegt, woher eine Webseite weiss, ob Sie diese auf einem Computer oder einem Smartphone ansehen? Wer befiehlt der Webseite, wie sie sich darzustellen hat?
Richtig, auf der Webseite wurden zwei unterschiedliche Eigenschaften zur Darstellung festgelegt, eine für Computerbrowser und eine für Smartphones.
Und warum lassen sich einige Bilder beliebig vergrössern, während andere schnell einmal unscharf werden? Dies hat nicht zwingend mit grossen Bildern zu tun, das Bild wurde möglicherweise in einem falschen Format gespeichert.
Oder was passiert eigentlich, wenn Sie in einem Textverarbeitungsprogramm wie Word einen Text schreiben und ein markiertes Wort mit der Schaltfläche „F“ markieren? Wenn Sie diesen Text in einem anderen Programm öffnen, woher weiss dieses, dass hier ein Wort „fett“ markiert wurde?
Richtig, beim Wort wurde die Eigenschaft „fett“ hinterlegt und mitgeschickt. Das Wort wird aber nur fett angezeigt, wenn diese Eigenschaft „fett“ bekannt ist und verstanden wird.
Missverständnisse können vorkommen, daher passiert es manchmal, dass Formatierungen auf Windows- und Mac-Geräten nicht gleich aussehen. Kennen Sie das?

All diese unterschiedlichen Probleme haben einen gemeinsamen Nenner: Die gewählten Eigenschaften, die bei einer Information mitgeliefert und dargestellt werden, sind Elemente von sogenannten „Auszeichnungssprachen“.

Wozu Auszeichnungssprachen?

Es ist, als würde man eine neue Landessprache lernen: Programmieren am Computer bedeutet, dass man sich mit Wortschatz, Grammatik und Rechtschreibung einer Programmiersprache auseinandersetzen muss. Dabei ist eine gute Struktur das A und O beim Schreiben eines Textes. Einer, der es wissen muss und täglich Blog-Beiträge im Internet schreibt, vergleicht die Struktur eines Textes mit einem Skelett: „Sie ist nicht für jeden sichtbar, aber ohne sie würde dein Artikel in sich zusammenfallen“[1].
Für die Struktur von Daten am Computer und im Internet sind „Auszeichnungssprachen“ sogenannte „Markup-Languages“, zuständig.
Im Internet begegnen uns jederzeit Auszeichnungssprachen, ohne dass wir diese bewusst wahrnehmen. Eine der bekanntesten ist HTML (Hyper Text Markup Language). Würde diese Auszeichnungssprache fehlen, gäbe es keine Webseiten, erst durch HTML wurden sie ermöglicht. Es gäbe kein Gross und Klein, keine Aufzählungen, keine Bilder und keine Hyperlinks.
Auszeichnungssprachen sind unter anderem auch dazu da, anderen Geräten anzugeben, wie sie die Daten weiterverarbeiten und darstellen müssen: z.B. die Darstellung eines Dokuments für den Drucker oder die Darstellung einer Webseite auf dem Smartphone oder die Sprachausgabe eines Textes für Menschen mit einer Seh-Behinderung.

[1] Zitat von Bamidele Onibalusi, gefunden bei http://www.toushenne.de/newsreader/schreiben-mit-system-warum-die-blogartikel-struktur-so-wichtig-ist.html (Juli 2014)

„Ursprünglich dienten die Auszeichnungssprachen im Text als Anweisungen für die Setzer im Drucksatz“[2]. Dabei wurden die Strukturmerkmale (wie z.B. Überschriften, Hervorhebungen im Text, Fussnoten, Kursivschrift u.a.) handschriftlich auf ein Manuskript eingetragen.

Wie ist eine Auszeichnungssprache aufgebaut?

Auszeichnungssprachen bestehen aus Elementen, Attributen (= charakteristische Eigenschaft) und Textinhalten. Durch die Auszeichnung eines Textes werden den Textinhalten Eigenschaften zugewiesen, die beschreiben, wie der Inhalt dargestellt wird.
Bei jeder dieser Auszeichnung muss definiert werden, wo eine Textauszeichnung beginnt und wo diese endet. Meist sind dies Startmarkierungen (start tag, z.B. in HTML <ul>) und Endmarkierungen (end tag, z.B. in HTML), sog. „Tags“. Die Markierung legt dann die Eigenschaft des umschlossenen Textes fest. Je nach Auszeichnungssprache können diese Markierungen unterschiedlich aussehen.

Beispiel für …	Darstellungsbeispiel	HTML	LaTeX	Wikitext
Überschrift	**Überschrift**	Überschrift	\section{Überschrift}	= Überschrift =
Aufzählung	▪ Punkt 1 ▪ Punkt 2 ▪ Punkt 3	 - Punkt 1 - Punkt 2 - Punkt 3 	\begin{itemize} \item Punkt 1 \item Punkt 2 \item Punkt 3 \end{itemize}	* Punkt 1 * Punkt 2 * Punkt 3
fetten Text	**fett**	fett	\textbf{fett}	'''fett'''
kursiven Text	*kursiv*	kursiv	\textit{kursiv}	''kursiv''

Vergleichstabelle unterschiedlicher Markup-Markierungen[3]

Die Markierungen werden nach festgelegten Regeln gebildet und sollten korrekt ineinander geschachtelt werden (d.h. sie müssen „wohlgeformt“ sein). Zum Beispiel im Bild von oben: Die HTML-Markierung <li>... funktioniert nur innerhalb der Markierung <ul>.... Die Markierungen ermöglichen somit, dass die Dokumentinhalte strukturiert werden.[4]

Ein Beispiel für eine Auszeichnungssprache und deren Aufbau

Folgender Satz

Der Kreis ist rot

besteht in der Auszeichnungssprache HTML aus dem Element <p> für einen Text-Absatz, dem Attribut „style=color:“ für die rote Schriftfarbe und dem Textinhalt „Der Kreis ist rot“. Im HTML-Code sieht das so aus:

[2] Quelle: http://de.wikipedia.org/wiki/Auszeichnungssprache (Juli 2014)
[3] Quelle: http://de.wikipedia.org/wiki/Auszeichnungssprache
[4] Quelle: http://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/technologien-methoden/Sprache/Auszeichnungssprache

```
<p style="color:#red">Der Kreis ist rot</p>
```

Hinweis:
Um den Aufbau einer Auszeichnungssprache zu erklären, eignen sich die beiden Sprachen HTML und SVG besonders gut. Wir gehen in diesem Modul auf die Markup-Language „SVG“ näher ein. Damit lässt sich aufzeigen, dass es nicht nur bekannte Formatierungen wie fett, kursiv, Auflistungen und Überschriften gibt, die man darstellen kann. Sondern auch geometrische Formen lassen sich mit Textinhalten und Zahlen beschreiben. Das Format SVG hat zudem grosse Vorteile für die Logogestaltung und im Bereich der Animationen. Die im Lernvideo vorgestellten Vorteile von SVG werden hier in einem separaten Kapitel nähergebracht.

Was ist SVG?

Das bevorzugte Vektorgrafikformat SVG heisst ausgeschrieben: Scalable Vector Graphics. Es weist daher auf die wichtigste Eigenschaft von Vektorgrafiken hin: Die verlustfreie Skalierung (= Grösse anpassen).
SVG ist ein vom World Wide Web Consortium (W3C) empfohlener Standard zur Beschreibung von zweidimensionalen Vektorgrafiken[5]. Die Datei-Endung ist .svg

Alle wichtigen modernen Webbrowser können SVGs in HTML-Seiten darstellen. In SVG gibt es Tags für Rechteck, Kreis, Linie, Polylinie, Texte und Animationen. SVG wird hauptsächlich in der Kartografie eingesetzt oder zur Visualisierung von Graphen und Illustrationen wie Logos und Icons.

Da es sich bei SVG um eine Auszeichnungssprache handelt, können SVG-Bilder in jedem beliebigen Texteditor[6] erstellt und bearbeitet werden. Es ist auch möglich, diese mit einem Zeichenprogramm wie Inkscape zu erstellen. Um aber zu verstehen, wie die Auszeichnungssprachen funktionieren, lohnt es sich, einen Blick auf die im Texteditor erstellte Variante zu werfen. Texteditoren sind wichtig für das Programmieren, da nur Texteditoren den Programmiercode richtig darstellen und speichern. Hier reicht es nicht, ein Textverarbeitungsprogramm wie zum Beispiel Word zu benutzen.

In diesem Modul können Schülerinnen und Schüler in einigen Experimenten herausfinden, wie das funktioniert. Die Bearbeitung von SVG wäre auch mit speziellen Programmen, sog. SVG-Editoren[7] möglich. Die Kenntnisse über Markup-Language sind aber besser in einem Texteditor oder in der Biber-eigenen Lernumgebung erfahrbar: http://mgje.github.io/draw/

5 mehr zu Vektor- und Rastergrafiken siehe Kapitel „Hintergrundwissen zu ‚Raster- und Vektorgrafik‘ “
6 Beispiele für Texteditoren: Alle Betriebssysteme: www.sublimetext.com / Windows: www.notepad-plus-plus.org / Mac: www.barebones.com/products/textwrangler
7 Beispiele dazu finden Sie im Bereich „Mögliche Vertiefungen“.

Wie sieht die Auszeichnungssprache „SVG“ aus?

Jede Auszeichnungssprache besitzt ein Wurzelelement, im Fall von SVG ist das <svg>, etwas genauer:

```
<svg height="420" width="620" xmlns="http://www.w3.org/2000/svg" version="1.1">
```

Dieser Code bedeutet, dass man eine Vektorgrafik (svg) mit der Höhe 420 Pixel und der Breite 620 Pixel erstellt, wobei sich alle Elemente innerhalb dieser SVG-Markierung (<svg>...</svg>) auf den Namensraum

```
xmlns="http://www.w3.org/2000/svg"
```

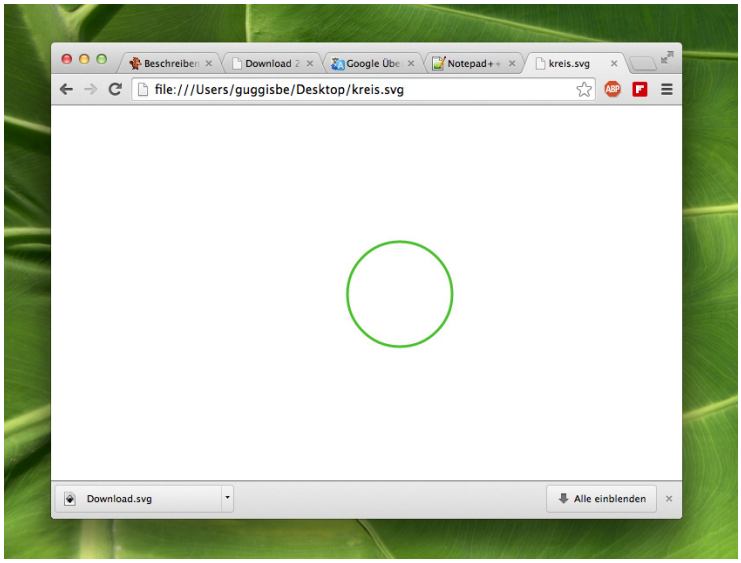
beziehen. Ein Namensraum bestimmt, auf welchen Wortschatz man sich bezieht, weil mehrere Auszeichnungssprachen in einem Dokument vorkommen und sich mischen können. Da muss klar sein, was eine Bezeichnung genau bedeutet[8].

Ausserdem soll es eine SVG-Grafik der Version 1.1 sein.

Eine Grafik mit genau einem Kreis sieht zum Beispiel so aus:

```
<svg height="420" width="620" xmlns="http://www.w3.org/2000/svg" version="1.1">
    <circle cx="400" cy="216" r="60" fill="none" stroke="#4cbf2f"
    stroke-width="3"></circle>
</svg>
```

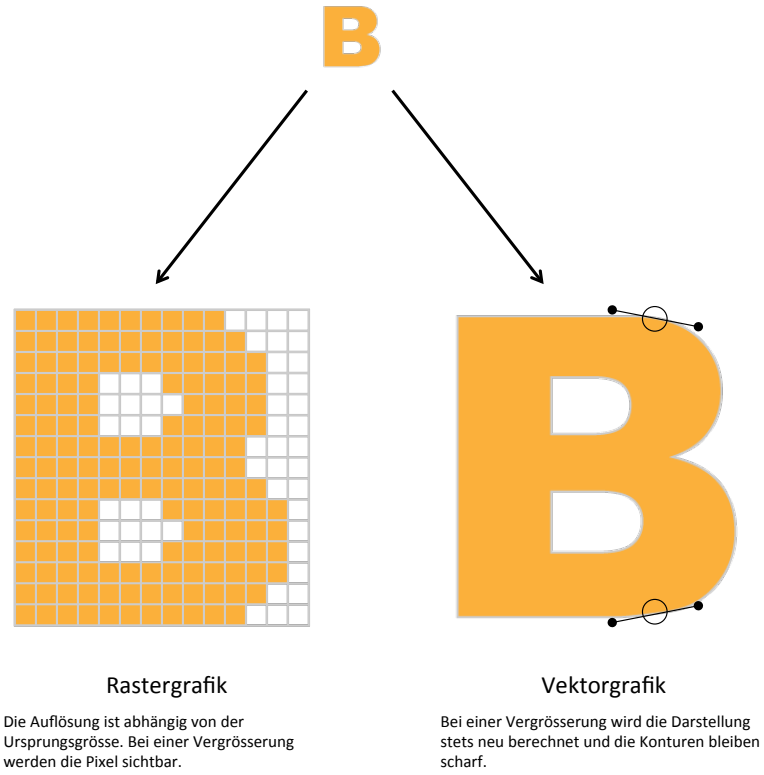
Kopiert man diese Zeilen in einen Texteditor und speichert diese Datei mit der Endung .svg ab, dann erscheint in jedem Browser ein grüner Kreis. Um die Datei im Browser zu öffnen, kann man einfach die Datei auf eine leere Webseite oder das Browser-Icon ziehen.



[8] Zum Beispiel: Ein <p>-Element bedeutet in HTML ein Absatz, in der XML-Sprache könnte <p> für eine Person stehen.

Hintergrundwissen „Rastergrafiken vs. Vektorgrafiken“

Digitale Bilder können auf zwei Arten dargestellt werden: Als Rastergrafik oder als Vektorgrafik. Die Vergrösserung zeigt den Unterschied auf.



Rastergrafik

Die Auflösung ist abhängig von der Ursprungsgrösse. Bei einer Vergrösserung werden die Pixel sichtbar.

Vektorgrafik

Bei einer Vergrösserung wird die Darstellung stets neu berechnet und die Konturen bleiben scharf.

Bei **Rastergrafiken**[9], wie z.B. dem alten Band-Logo unserer vier Freunde, werden die Informationen in winzigen Bildpunkten gespeichert, in sogenannten „Pixeln“. Zoomt man nah in eine Rastergrafik, sieht man die einzelnen Pixel. Jedem Bildpunkt (im Raster entspricht das jedem Kästchen) ist eine Farbe zugeordnet. Rastergrafiken eignen sich für komplexere Bilder wie Fotos, haben aber den Nachteil, dass sie sich nicht beliebig vergrössern lassen.

Bei **Vektorgrafiken**, wie dem neuen Band-Logo, werden die Informationen als geometrische Elemente (wie Linien, Kurven und Flächen) gespeichert, in sogenannten „Vektoren“. Die Bildinformationen werden in Text umgewandelt, welcher je nach Programm grafisch dargestellt wird. Um das Bild eines Kreises zu speichern, benötigt die Vektorgrafik z.B. nur zwei Werte: Die Lage des Kreismittelpunktes und den Kreisradius. Der Vorteil der Vektorgrafik ist, dass sie sich verlustfrei vergrössern lässt. Vektorgrafiken eignen sich hauptsächlich für Logos und Illustrationen.

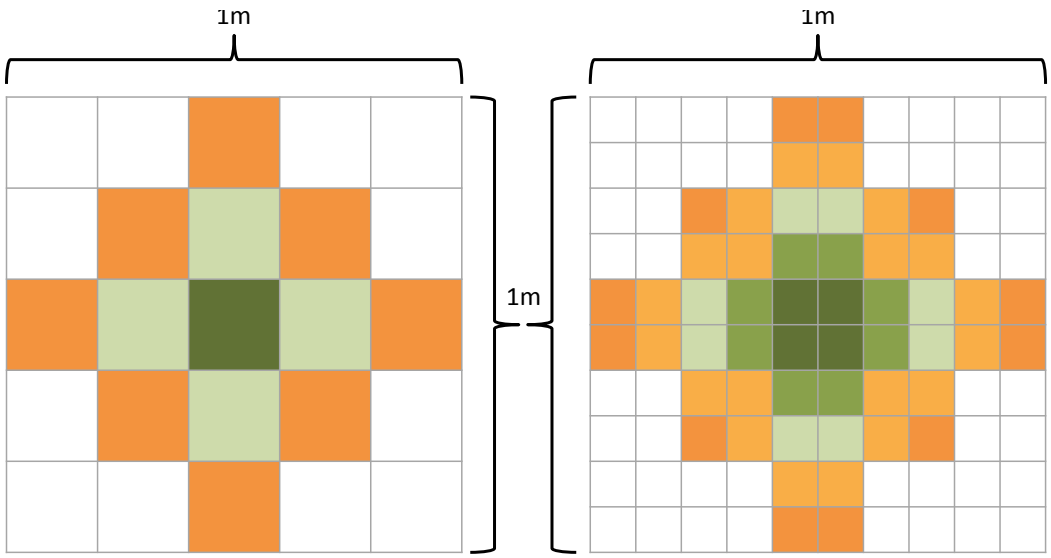
Warum hat das alte Band-Logo eine schlechte Qualität?

Dies liegt daran, dass es als Rastergrafik abgespeichert ist und sich nicht beliebig vergrössern lässt. Wird z.B. das Biber-B im Bild oben als Rastergrafik gespeichert (in einem Bildfor-

[9] siehe auch Biber-Modul „Musik: Komprimieren“, Arbeitsblatt 04

mat wie z.B. jpg, png, tiff), muss die Datei für jeden Bildpunkt einzeln einen Farbwert speichern. Hier spielt die **Pixeldichte** eine wichtige Rolle: Die Pixeldichte klärt die Frage, wie viele Pixel auf eine bestimmte Länge passen (sie wird in ppi = „pixel per inch“ angegeben).

Eine gute Vorstellung für die Pixeldichte bietet die Metapher des „geknüpften Teppichs“: Stellen wir uns vor, wir haben zweimal dieselbe Grösse eines Teppichs, sagen wir 1 x 1m. Wenn der Teppich mit grober Wolle geknüpft wird, passen auf dieselbe Länge viel weniger Knoten. Das geknüpfte Bild wird sehr ungenau. Wenn der Teppich mit feiner Seide geknüpft wird, passen viel mehr Knoten auf diese Länge, da jeder Knoten schmaler ist. Das geknüpfte Bild ist sehr viel genauer und enthält mehr Details, da wir nun viel mehr Farbnuancen knüpfen können: Wo wir mit der dicken Wolle z.B. nur einen orangen Knoten machen konnten, haben mit Seide auf einmal vier Farbnuancen von orangen Knoten Platz.



links: Geknüpfter Teppich mit dicker Wolle (ungenau)
rechts: Geknüpfter Teppich mit feiner Seide (detaillierter)

Grundsätzlich gilt: Eine hohe Pixeldichte enthält viele Pixel, eine kleine Pixeldichte enthält wenige Pixel.

Wenige Pixel heisst aber auch, dass das Bild nur in einer kleinen Anzeigegrösse (Auflösung) qualitativ gut dargestellt wird, da weniger Informationen gespeichert wurden. Je mehr Pixel auf einem Display angezeigt werden, desto schärfer wird das Bild. Diese Information ist zum Beispiel auch im Hinblick auf die Entwicklung von Retina-Displays und hochauflösenden Smartphone-Bildschirmen relevant: Ein Foto, das auf einem Gerät mit kleiner Pixeldichte gut aussieht, kann auf einem Gerät mit grosser Pixeldichte, wie z.B. Retina, unscharf aussehen, obwohl die Anzeigefläche in Zentimeter genau gleich gross ist bei beiden Geräten.

Exkurs Auflösung:
Bei der Bildqualität spielt auch die Auflösung eines Bildes eine Rolle. Sie gibt an, auf was für einer Ausgabegrösse das Bild dargestellt wird. Es macht dabei einen Unterschied, ob es sich z.B. um Ausdrucke handelt oder um Darstellungen an einem Monitor oder an einem Tablet mit Retina-Display. Jedes Ausgabegerät hat eine eigene Auflösung. Die Anzahl Pixel der Breite, multipliziert mit der Anzahl Pixel der Höhe, definiert die Auflösung eines Bildes.

Wenn wir z.B. ein Foto mit einer Höhe von 1600 Pixeln und einer Breite von 1'800 Pixeln haben, werden in der Datei 1'600 x 1'800 = 2'880'000 Bildpunkte (das sind 2,8 Megapixel) gespeichert, wobei jeder dieser Bildpunkte einen Farbwert hat (siehe Exkurs Farbtiefe). Haben wir zu Hause eine Kamera mit der Auflösung 2,8 Megapixel, kann sie Fotos machen, die 1'600x1'800 Pixel gross sind.

Je mehr Pixel auf einem Display derselben Grösse sind, desto schärfer ist das Bild. Beispiel: Ein 15 Zoll-Laptop kann unterschiedliche Auflösungen haben, obwohl die Fläche des Displays genau gleich gross ist. Dasjenige mit der grösseren Auflösung stellt Inhalte jedoch schärfer dar.
Die Auflösung gibt keine Informationen darüber, wie gross eine Fläche in Zentimetern ist. Denn hier wiederum spielt die Pixeldichte[11] eine wichtige Rolle. Denn diese gibt das Verhältnis zwischen Anzeigegrösse und Pixelanzahl an.

Hier zeigt sich bereits eine Problematik bei Rastergrafiken auf einer Webseite: Sie sind nicht beliebig vergrösserbar und können je nach Anzeigegrösse eines Geräts besser oder schlechter aussehen.

Das alte Band-Logo wurde mit wenigen Pixel-Informationen gespeichert. Daher ist seine Qualität auf einem grossen Bildschirm schlecht.

Es stellt sich daher die Frage:

Warum kann man das alte Band-Logo nicht mit mehr Pixeln speichern? Bzw. wie „schwer“ ist ein Bild?

Jedes Bild benötigt eine bestimmte Menge an Speicherplatz, man kann sich dies auch als „Gewicht“ eines Bildes vorstellen. Je schwerer ein Bild im Internet ist, d.h. je mehr Speicherplatz es benötigt, desto länger braucht die Internetverbindung, um das Bild herunterzuladen und anzuzeigen.

Daher sind auch die Ladezeiten der Fotos der Band-Webseite endlos: Die Bilder wurden mit vielen Pixel-Informationen gespeichert (d.h. die Qualität auf einem grossen Bildschirm ist gut), aber dadurch hat sich auch der Speicherbedarf erhöht.

[10] in Online-Rechner für die Pixeldichte inkl. Formelerklärung: http://www.werbeverdienste.de/rechner/ppi-pixeldichte-rechner.html

Der Speicherbedarf eines Bildes hängt mit der Pixeldichte und der Anzeigegrösse (Auflösung) zusammen. Zudem spielt auch die Fabrtiefe eine sehr wichtige Rolle dabei.

Exkurs Farbtiefe[12]:
Ein Bild kann unterschiedliche Farbabstufungen haben. Je mehr Farbabstufungen vorhanden sind, desto mehr Farben hat das Bild. Aber auch: desto grösser wird der Speicherbedarf. Die Farbtiefe gibt den Wert an, wie viel Bits an Informationen pro Pixel gespeichert werden.

Bei einer Farbtiefe von 1 Bit sind nur zwei Zustände möglich: Schwarz oder Weiss.
Wenn wir ein Bild mit 1600 x 1800 Pixeln haben, so ist dieses Bild 2.8 Megapixel * 1 = 2,88 Megabit gross. Da normalerweise von „Megabyte“ gesprochen wird, können wir diese Angabe umwandeln. Da 8 Bit = 1 Byte ist, ergibt 2,88 Megabit : 8 = 0,36 Megabyte.

Bei einer Farbtiefe von 8 Bit (= 1 Byte) kann jeder Pixel 28 = 256 verschiedene Graustufen darstellen. Diese Farbtiefe wird in der Regel für Schwarzweiss-Fotos verwendet.
Unser Bild mit 1600 x 1800 Pixeln benötigt somit 2,8 Megapixel * 8 : 8 = 2,8 Megabyte Speicherplatz.

Bei einer Farbtiefe von 24 Bit (= 3 Byte) können ebenfalls 256 Abstufungen dargestellt werden, aber nicht nur schwarz und weiss, sondern für die drei Farbkanäle Rot, Grün und Blau (RGB-Modus). Dadurch ergeben sich insgesamt 256^3 = 16,7 Millionen verschiedene Farbabstufungen. Diese Farbtiefe wird in der Regel für Farbfotos verwendet.
Unser Bild mit 1600 x 1800 Pixeln benötigt somit 2,8 Megapixel * 24 : 8 = 8,4 Megabyte Speicherplatz.

Die Formel zur Berechnung des Speicherplatzes eines Fotos ist:
Breite des Bildes [in Pixel] x Höhe des Bildes [in Pixel] * Farbtiefe [in Bit]

Rechnet man dieses Resultat durch 8, erhält man Anzahl Byte anstelle von Bit.
Und rechnet man es nachher durch 1‘000‘000, erhält man das Resultat in Megabyte.

Die Fotos der Band-Webseite werden nur sehr langsam geladen. Dies deutet daraufhin, dass die Anzahl Bildpunkte eines Bildes zu gross ist und die Datenleitung zu langsam, um die Bilder flüssig zu laden. Je mehr Pixel eine Bilddatei enthält, desto grösser ist in der Regel die Dateigrösse. Ist zudem noch die Farbtiefe gross (z.B. für Farbfotos), wird der Speicherbedarf sehr viel grösser als mit einem Schwarzweiss-Bild.

Eine Lösung für das Problem – so könnte man denken – ist es, den Speicherbedarf zu ver-

[11] Quelle: http://www.copyshop-tips.de/scan07.php

ringern. Es gibt zwei Möglichkeiten, damit ein Foto nicht so viel Speicherplatz benötigt:

1. Man kann es verkleinern, man sagt dem auch „komprimieren“[12]. Dies ist zum Beispiel mit dem Bildformat JPEG möglich.
2. Man kann die Pixeldichte eines Bildes verringern.
3. Man kann die Auflösung eines Bildes verringern.

Alle drei Anpassungen führen zu einem Qualitätsverlust und dies wiederum zu schlechten Bildern.

Wann ein Bild auf dem Display scharf aussieht, hängt somit von Auflösung, Displaygrösse und dem Ausgangsmaterial ab[13].

Was kann die Vektorgrafik, was die Rastergrafik nicht kann?

Der Grafiker und Animator im Lernfilm zeigt, warum die Vektorgrafik für ein Logo der Rastergrafik vorzuziehen ist:

- Vektorgrafiken sind beliebig skalierbar (vergrösserbar) und können daher auf beliebigen Ausgabegrössen (Smartphone, Monitor, grosser Fernseher, Beamer, Poster, Flyer, ...) ohne Qualitätsverlust dargestellt werden.
- Vektorgrafiken benötigen viel weniger Speicherplatz und haben daher wenig Einfluss auf die Ladezeit einer Webseite. Es gibt daher keine Auflösung wie bei einer Rastergrafik, sondern die Frage ist nur, wie viele und was für Objekte diese Vektorgrafiken enthalten.

Grafische Elemente einer Vektorgrafik können erstellt, transformiert und animiert werden. Dazu benötigt man nur die entsprechenden Befehle, die in den Arbeitsblättern näher betrachtet werden.

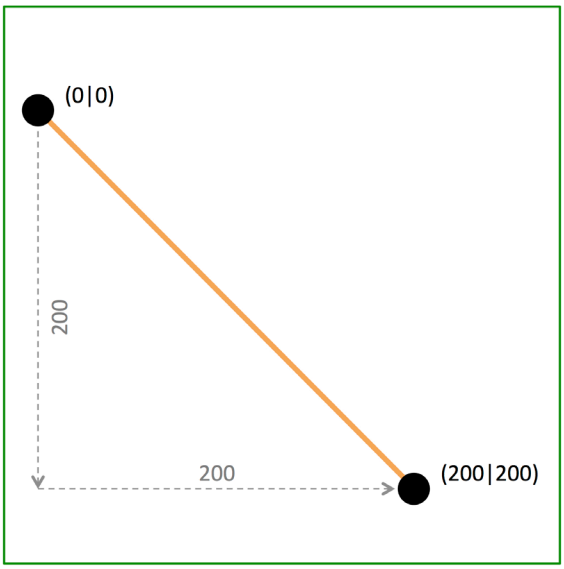
Da Vektorgrafiken nicht aus Pixeln bestehen, sieht man bei Vergrösserung einer solchen keine „Kästchen“ oder ähnliche Formen. Wird eine Vektorgrafik vergrössert, so „wachsen“ die geometrischen Elemente mit.

[12] siehe Biber-Modul „Musik: Komprimieren“

[13] Eine ausführliche Erklärung dieser drei Punkte findet sich hier: http://www.deathmetalmods.de/was-macht-ein-display-scharf/

Beispiel 1:

Bei einer Linie ist der Anfangs- und Endpunkt bestimmt. Diese Punkte werden durch eine mathematische Berechnung mit einer Linie verbunden. Wird die Vektorgrafik vergrössert, werden Anfangs- und Endpunkte neu positioniert, und die Linie wird dazwischen neu berechnet. Da die Linie bei jeder Vergrösserung oder Verkleinerung neu gezeichnet wird, bleibt die Linie (auch wenn man nah heranzoomt) scharf.




```
<svg height="210" width="210">
    <line x1="0" y1="0" x2="200" y2="200" stroke="#F8A65F" stroke-width="3.5" />
</svg>
```

Grösse der SVG-Leinwand

Anfangspunkt

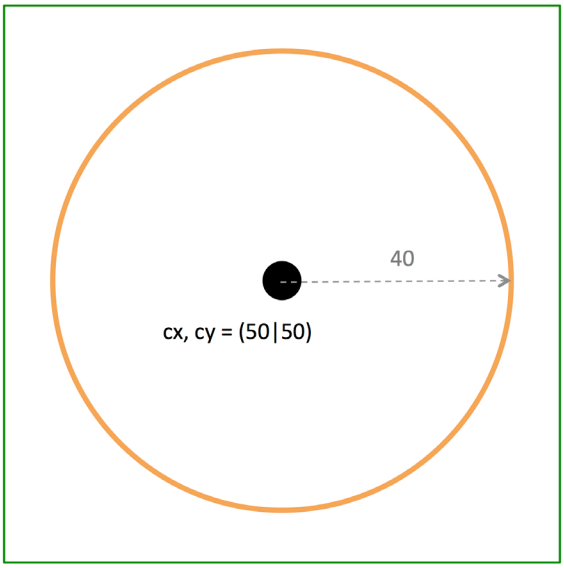
Endpunkt

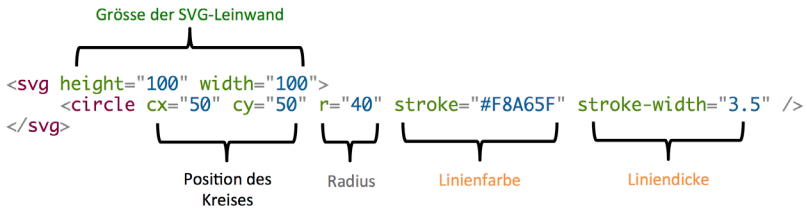
Linienfarbe

Liniendicke

Beispiel 2:

Bei einem Kreis ist der Kreismittelpunkt und der Kreisradius wichtig: Sie legen fest, wie gross der Kreis ist und an welcher Position er sich befindet. Vergrössert man den Kreis, wird die Kreislinie neu berechnet. Es wird daher nicht eine bestehende Linie auseinandergezogen (wie z.B. bei einem Gummiband), sondern sie wird angepasst.

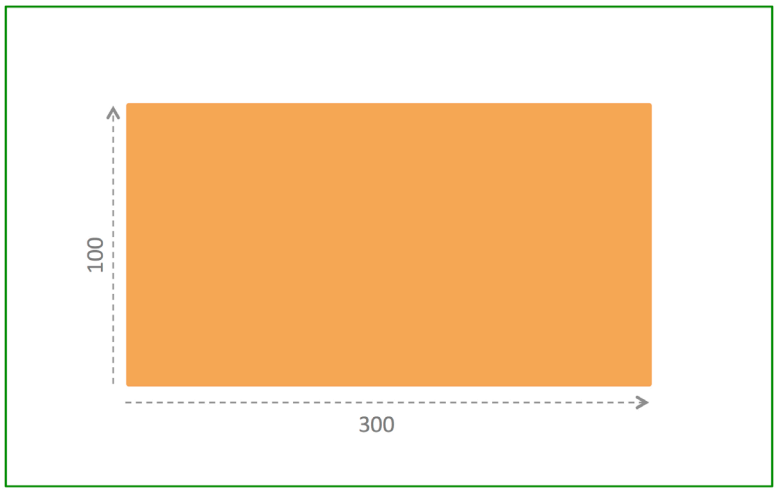


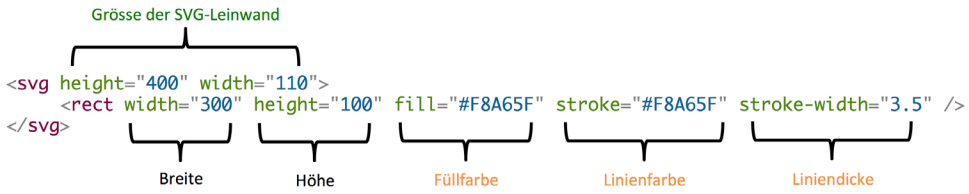


Umsetzungshilfen

Beispiel 3:

Bei einem Rechteck wird mit Breite und Höhe die Grösse festgelegt. Zusätzlich wird die Fläche mit einer Farbe ausgefüllt. Wird das Rechteck vergrössert, werden die Seitenlängen angepasst. Auch hier wird die Form neu berechnet.





Reflexion Lernfilm / Perspektiven der Lernenden

Im Plenum wird der Lernfilm „Auszeichnungssprachen – Wie lassen sich digitale Inhalte beschreiben?“ angeschaut.
Zur Reflexion des Lernfilms können folgende Fragen in der Klasse aufgenommen werden:

- Was sind die Unterschiede zwischen einer Rastergrafik und einer Vektorgrafik?
- Warum soll das Bildformat SVG für das Logo verwendet werden?
- Wie sieht eine Auszeichnungssprache wie z.B. SVG aus?

Diese und ähnliche Fragen werden in den Experimenten thematisiert. Die Experimente lassen sich in einem eigenen Texteditor umsetzen oder über die Lernumgebung http://mgje.github.io/draw

Experiment 1: „Wie lässt sich ein Logo aus farbigen Kreisen bauen?“

Die Schülerinnen und Schüler lernen in einem ersten Schritt, mit dem Element Kreis den SVG-Code aufzubauen und leicht zu verändern. Die Veränderung besteht darin, mehr Kreise zu zeichnen oder die Eigenschaften des Kreises (Radius, Linienfarbe, ...) zu verändern.

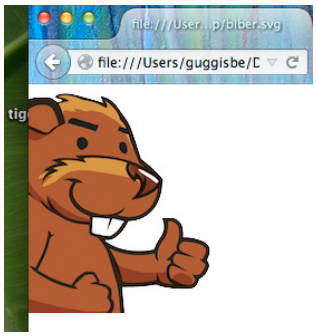
Neben Kreisen lassen sich auch gerade und krumme Linien zeichnen. Wie das geht, zeigt
http://mgje.github.io/draw/#Vieleck
http://mgje.github.io/draw/#Kurve2

Weitere Informationen dazu findet man in den SVG-Spezifikationen zum Element path:
http://www.w3schools.com/svg/default.asp

Experiment 2: „Wie lässt sich ein Logo verschieben, drehen oder verzerren?“

Hier lernen Schülerinnen und Schüler, wie sie ein Bild im Internet transformieren können.
Neben geometrischen Figuren können auch ganze Bilder (Bitmaps) innerhalb von SVG dargestellt werden. Dazu muss das Bild irgendwo auf dem Internet liegen.
Ein Bild des Informatik-Bibers erhält man wie folgt:

```
<svg height="420" width="620" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1">
    <image x="0" y="0" width="166" height="205"
    xlink:href="http://mgje.github.io/draw/images/biber.png"></image>
</svg>
```



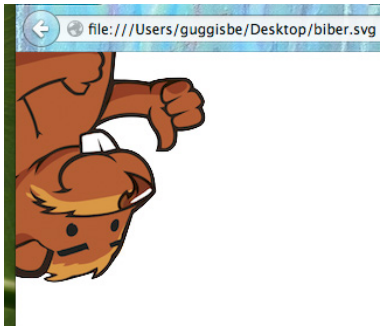
Alle SVG-Elemente lassen sich transformieren, verändern. Dazu benutzt man eine Transformations-

matrix. Wie das mathematisch funktioniert, findet man im Internet. Die Schülerinnen und Schüler können es sich womöglich von der Lehrperson erklären lassen.
Was eine Veränderung bewirkt, kann durch eigenes Experimentieren rasch herausgefunden werden. Zum Beispiel stellt die folgende Transformation den Biber auf den Kopf:

```
transform="matrix(1,0,0,-1,0,205)"
```

Diese Transformation kann einfach als Attribut im ‹Image›-Element wirken:

```
<svg height="420" width="620" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1">
    <image x="0" y="0" width="166" height="205"
    xlink:href="http://mgje.github.io/draw/images/biber.png"
    transform="matrix(1,0,0,-1,0,205)">
    </image>
</svg>
```



Mit

```
transform="matrix(1,0,0.4,1,0,0)"
```

lässt sich der Biber verzerren.

Auch drehen ist möglich, siehe http://mgje.github.io/draw/#Transformation

Experiment 3: „Wie bringe ich Bewegung in ein Logo?“

Das Animieren von SVG ist interessant, aber nicht ganz einfach. Hierzu wird ein neues Attribut verwendet, das „animate“-Attribut. In einer kleinen Übung können die Schülerinnen und Schüler ein Bild mit diesem Animations-Attribut horizontal bewegen.
Möchte man mehr Animationen kennenlernen, bietet sich dieses kleine Tutorial als Einstieg an:

Informatik-Biberaufgaben zum Thema Auszeichnungssprachen

Die Biberaufgaben stellen konkrete Fragen zu Themen rund um Auszeichnungssprachen. Sie wurden den drei Schwierigkeitsstufen einfach, mittel, schwierig zugeordnet und liegen in den entsprechenden Verzeichnissen.

Weitere Informationen zum Informatik-Biber-Wettbewerb finden Sie auf der Webseite *http://www.informatik-biber.ch/*

Mögliche Vertiefungen

Online SVG-Editor mit SVG-Code:
http://svg-edit.googlecode.com/svn-history/r1771/trunk/editor/svg-editor.html

Online 3D-Editor mit HTML- und CSS-Code zum Erstellen von 3D-Objekten:
http://trivid.com

Einführung in HTML:
http://clab2.phbern.ch/script/html/html.htm

Unterrichtsmaterial zu „Digitale Bildformate“:
http://www.swisseduc.ch/informatik/anwendungen/grafikformate/index.html

Ausführliches SVG-Tutorial:
- http://svg.tutorial.aptico.de/start.php
- http://www.mediaevent.de/tutorial/svg-animate.html (Animationen)

Farbtabelle Hex-Werte:
http://www.hexfarben.de

Weiterführende Literatur

Erlenkötter, H. (2003). XML: Extensible Markup Language von Anfang an, rororo Verlag ISBN-13: 978-3499612091

Spona H. (2001). SVG-Webgrafiken mit XML, vmi-Buch Verlag. ISBN-13: 978-3826671814

SVG Dokumentation und Beispiele:
- https://developer.mozilla.org/de/docs/Web/SVG
- http://docs.webplatform.org/wiki/svg
- http://de.wikibooks.org/wiki/SVG
- http://www.mediaevent.de/tutorial/svg.html

Tutorial zu INKSCAPE: http://www.inkscape.org/de/lernen/tutorials/

Unterrichtsmaterial für den Informatikunterricht:
- http://www.swisseduc.ch/informatik/
- http://www.educ.ethz.ch/unt/um/inf

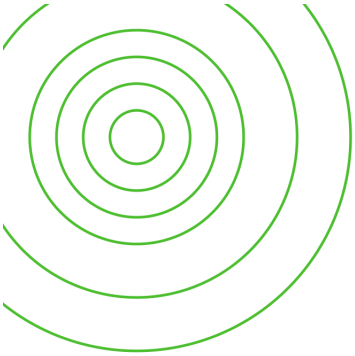
Lösungen

Mögliche Lösungen

Experiment 1: „Wie lässt sich ein Logo aus farbigen Kreisen bauen?“

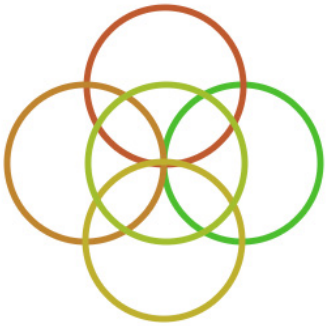
Aufgabe 1)

```
<svg height="800" width="800" xmlns="http://www.w3.org/2000/svg" version="1.1">
    <circle cx="250" cy="250" r="200" fill="none" stroke="#4cbf2f"
    stroke-width="5" style=""></circle>
    <circle cx="250" cy="250" r="400" fill="none" stroke="#4cbf2f"
    stroke-width="5" style=""></circle>
    <circle cx="250" cy="250" r="300" fill="none" stroke="#4cbf2f" stroke-width="5"
    style=""></circle>
    <circle cx="250" cy="250" r="100" fill="none" stroke="#4cbf2f" stroke-width="5"
    style=""></circle>
    <circle cx="250" cy="250" r="150" fill="none" stroke="#4cbf2f" stroke-width="5"
    style=""></circle>
    <circle cx="250" cy="250" r="50" fill="none" stroke="#4cbf2f" stroke-width="5"
    style=""></circle>
</svg>
```



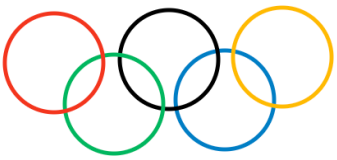
Aufgabe 2)

```
<svg height="420" width="620" xmlns="http://www.w3.org/2000/svg" version="1.1">
    <circle cx="550" cy="245" r="60" fill="none" stroke="#4cbf2f"
    stroke-width="4.912280701754386" style=""></circle>
    <circle cx="491" cy="186" r="60" fill="none" stroke="#bf5a2f"
    stroke-width="4.912280701754386" style=""></circle>
    <circle cx="430" cy="245" r="60" fill="none" stroke="#bf852f"
    stroke-width="4.912280701754386" style=""></circle>
    <circle cx="490" cy="304" r="60" fill="none" stroke="#bfb02f"
    stroke-width="4.912280701754386" style=""></circle>
    <circle cx="492" cy="245" r="60" fill="none" stroke="#a2bf2f"
    stroke-width="4.912280701754386" style=""></circle>
</svg>
```



Aufgabe 3)

```
<svg height="420" width="620" xmlns="http://www.w3.org/2000/svg" version="1.1">
    <circle cx="401" cy="232" r="60" fill="none" stroke="#007FC6"
    stroke-width="4.912280701754386" style=""></circle>
    <circle cx="471" cy="180" r="60" fill="none" stroke="#FFBA00"
    stroke-width="4.912280701754386" style=""></circle>
    <circle cx="333" cy="183" r="60" fill="none" stroke="#000000"
    stroke-width="4.912280701754386" style=""></circle>
    <circle cx="266" cy="237" r="60" fill="none" stroke="#00B760"
    stroke-width="4.912280701754386" sty-le=""></circle>
    <circle cx="193" cy="187" r="60" fill="none" stroke="#F4361D"
    stroke-width="4.912280701754386" style=""></circle>
</svg>
```



Experiment 2: „Wie lässt sich ein Logo verschieben, drehen oder verzerren?“

Aufgabe 1)

```
<svg height="420" width="620" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1">
     <image x="180" y="110" width="166" height="205"
     xlink:href="http://mgje.github.io/draw/images/biber.png"
     opacity="0.3"></image>
     <image x="180" y="110" width="166" height="205"
     xlink:href="http://mgje.github.io/draw/images/biber.png"
     transform="matrix(1,0,0,1,190,0)"></image>
</svg>
```

```
<svg height="420" width="620" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1">
     <image x="180" y="110" width="166" height="205"
     xlink:href="http://mgje.github.io/draw/images/biber.png"
     opacity="0.3"></image>
     <image x="180" y="110" width="166" height="205"
     xlink:href="http://mgje.github.io/draw/images/biber.png"
     transform="matrix(-1,0,0,1,676,0)"
     stroke-width="1.2280701754385965" style=""></image>
</svg>
```

Aufgabe 2)

```
<svg height="420" width="620" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1">
     <image x="180" y="110" width="166" height="205"
     xlink:href="http://mgje.github.io/draw/images/biber.png"
     opacity="0.3"></image>
     <image x="180" y="110" width="166" height="205"
     xlink:href="http://mgje.github.io/draw/images/biber.png"
     transform="matrix(1,0,0.5033,1,0,0)"></image>
</svg>
```

Aufgabe 3)

```
<svg height="420" width="620" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1">
     <image x="180" y="110" width="166" height="205"
     xlink:href="http://mgje.github.io/draw/images/biber.png"
     opacity="0.3"></image>
     <image x="180" y="110" width="166" height="205"
     xlink:href="http://mgje.github.io/draw/images/biber.png"
     transform="matrix(-0.1045,0.9945,-0.9945,-0.1045,501.8269,-26.847)"></image>
</svg>
```

Experiment 3: „Wie bringe ich Bewegung in ein Logo?“

Aufgabe 1)

```
<svg height="420" width="620" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1">
    <image x="70" y="130" width="166" height="205" preserveAspectRatio="none"
    xlink:href="http://mgje.github.io/draw/images/biber.png">
        <animate attribute-Type="XML" begin="click" attributeName="x" from="70"
            to="630" dur="1.3s" fill="freeze"></animate>
    </image>
</svg>
```

Aufgabe 2)

Individuelle Lösungen

Aufgabe 3)

```
    <image x="70" y="130" width="166" height="205" preserveAspectRatio="none"
    xlink:href="http://mgje.github.io/draw/images/biber.png">
        <animate attribute-Type="XML" begin="click" attributeName="width"
        from="166" to="398.4" dur="3.3s" fill="freeze"></animate>
        <animateTransform begin="click" attributeType="XML"
        attributeName="transform" type="rotate" from="0 153 232" to="359.9 153 232"
        repeatCount="indefinite" additive="sum" dur="3.3s" fill="freeze">
        </animateTransform>
    </image>
</svg>
```

Impressum

Herausgeber	SVIA, Schweizerischer Verein für Informatik in der Ausbildung
Partner	Hasler Stiftung ICT Berufsbildung SWITCH
Konzeption/Umsetzung	LerNetz AG
Autorenschaft	Martin Guggisberg, PH FHNW Yvonne Seiler