Graphische Darstellungen von Kurven & Flächen im Raum mit *Mathematica*

Struktur & Aufgaben

für ein selbständiges Einarbeiten

Ronald Balestra
CH - 8046 Zürich
www.ronaldbalestra.ch

Name:

Vorname:

4. Januar 2020

Das *Ziel* der folgenden Unterlagen ist das selbständige Kennenlernen von Möglichkeiten der graphischen Darstellung von *Kurven & Flächen im Raum mit Mathematica*.
Dazu gehört auch das Kennenlernen und der Umgang mit den von Wolfram angebotenen Hilfen, dem *Wolfram Documentation Center* und natürlich den Hilfen, welche sonst noch im *www* zu finden sind.
Es geht hierbei nicht darum, alle Möglichkeiten der Darstellung kennenzulernen, sondern sich einen Einblick zu verschaffen, was möglich ist und wo ich die Informationen zu einer Umsetzung und gegebenefalls Vertiefung finden kann.

Der Aufbau ist in drei Abschnitte gegliedert:

1. Die graphische Darstellung von Funktionen des Typs $f : \mathbb{R} \to \mathbb{R}$ und der *Show[]*-Befehl.
2. Der *ParametricPlot[]*, für Kurven & Flächen im Raum,
3. *Graphics* und *Primitives*.

und enthält jeweils einen Link zu einer Einstiegsliteratur.

Zielsetzungen:

Die *graphishe Darstellung*

- von Funktionen, mit einem sinnvollen Einsatz von Möglichkeiten, welche Mathematica zur Bearbeitung bietet,
- das Gleiche mit Kurven & Flächen im Raum,
- die graphische Darstellung geometrischer Grundfiguren und
- die gemeinsame Darstellung in einem Koordinatensystem.

Arbeitstechnisch der Umgang mit zu umfangreichen Skripten. Dabei gilt es zu erkennen und zu entscheiden

- was ist wichtig?
- was kann ich überspringen, weglassen?
- wie komme ich zeitlich über die Runden (timemanagement)?
- das Arbeiten mit der *Documentation von Wolfram* und der *Hilfe* in Mathematica.

1 Die graphische Darstellung von Funktionen des Typs $f : \mathbb{R} \to \mathbb{R}$ und der *Show[]*-Befehl

Wir beginnen mit einer Einführung in die graphischen Darstellung von reellwertigen Funktionen und Möglichkeiten der Bearbeitung der Graphik und der Einführung des *Show[ ]* - Befehls.

- Wir verwenden dazu als Grundlage das Skript von Rudolf Schürer *Grafische Darstellung*:

  http://docplayer.org/49133008-Grafische-darstellung-von-funktionen.html

- Beachte:
  - im Skript verwendete Darstellung für Flächen:

    $$f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$$

  - der Abschnitt zur *Genauigkeit* kann wegggelassen werden.
  - eine weiterführende, sehr ausführliche Quelle liefert:

    Michael O. Distler (Vorlage L. Tiator)
    Einführung in Mathemtica (2)

Wichtig in diesem Abschnitt sind:

- die graphische Darstellung mehrerer Funktionen in einem Plot,
- die Anwendung sinnvoller Darstellungs- & Beschriftungsmöglichkeiten,
- die graphische Darstellung mehrere Funktionen mit dem *Show[ ]*- Befehl, sowohl nebeneinander als auch überlagernd.
- Mach dich mit dem *Manipulate*-Befehl vertraut.

2 Der *ParametricPlot[]*

Hierfür braucht es schon grundlegende Kenntnisse zum Thema *Kurven & Flächen im Raum*.

Der *ParametricPlot*, dessen Einführung und die Anwendungen der erarbeiteten Kenntnisse im ersten Kapitel und den Bearbeitungsmöglichkeiten mit diesem *Plot*-Befehl.

- Wir verwenden dazu als Grundlage die Unterlagen von *Wolfram Documentation*

  http://reference.wolfram.com/language/tutorial/ParametricPlots.html

- Vorgehensweise & Ziele:
  - Übertrage deine Erkenntnisse aus dem 1. Abschnitt in dem *ParametricPlot[ ]*
  - Verwende die Hilfen von *Wolfram* und mach dich mit der Navigation & den Hilfsmöglichkeiten auf dieser Seite vertraut.

Aufgaben : Die graphische Darstellung einer Fläche und je einer ausgezeichneten u-, bzw. v-Linie mit dem *Show[ ]* - Befehl, sowohl nebeneinander als auch überlagernd. Ergänze die graphischen Darstellungen jeweils mit einer sinnvollen Beschriftung.

3 *Graphics* und *Primitives*

Das Arbeiten mit *Graphics* und *Primitivs*

- Wir verwenden dazu einen Auszug aus den Unterlagen von *Thomas Unseld*

  Arbeiten mit Listen und der Graphic-Befehl, ab p.23 (siehe Anhang)

- und machen uns weiter vertraut mit der Dokumentation/ dem Documentation Center von Wolfram

  . . . zum Thema *Graphics*

- Wichtig für *dich* in diesem Abschnitt sind:

  –

Der Graphics-Befehl

Die 3 wichtigsten Graphik-Befehle in *Mathematica* sind Plot[ ], ListPlot[ ], die du bereits kennst, und der Graphics-Befehl: Der **Graphics-Befehl** arbeitet mit sogenannten **Graphik-Primitiven wie Linie, Punkt, Kreis, ...** :
Wir speichern sie in den Variablen "punkt", "linie1", ... ab:

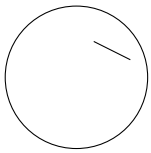
```
punkt = Point[{2, 2}]
linie1 = Line[{{1, 2}, {3, 1}}]

linie2 = Line[{{-1, -2}, {-2, 1}, {-3, 0}}]
dreieck = Polygon[{{-1, 2}, {-2, 6}, {-3, 1}}]
kreis = Circle[{0, 0}, 4]
```

Punkt mit den Koordinaten (2 | 2)
Strecke als **Liste** von 2 Punkten
Streckenzug als **Liste** von 3 Punkten
ausgefülltes Vieleck, Polygon als **Liste**
Kreis mit Mittelpunkt (0 | 0) und r = 4

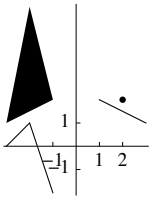
Primitiv nennt man die oben definierten Graphik-Primitiven Point[ ], ..., weil noch keine Darstellungsbedingungen definiert sind. Das geschieht nun mit dem **Graphics-Befehl**, wo man wie beim Plot-Befehl mit Hilfe von Optionen als zweites Argument die Darstellung der Objekte definieren kann. In einer **Liste** im ersten Argument des Graphics-Befehl lege ich fest, welche der oben definierten Objekte ich darstellen will:

```
Graphics[{kreis, linie1}, AspectRatio → Automatic]
```



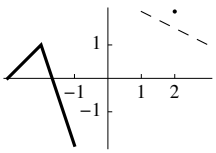
Jetzt das Dreieck, der Punkt, die 2 Linien und die Achsen des Koordinatensystem:

```
Graphics[{linie1, linie2, punkt, dreieck}, Axes → True, Ticks → {{-1, 1, 2}, {-1, 1}}]
```



Wenn man nun nur die 2 Linien (eine davon gestrichelt und die andere dicker) und den Punkt dicker darstellen will, macht man das nicht via Optionen wie beim Plot- oder ListPlot-Befehl, sondern mit Hilfe von **Graphik-Direktiven vor** den Graphik-Primitiven. Das macht auch Sinn, weil man so für jedes Objekt eine eigene Darstellung wählen kann und sie so nicht für den ganzen Plot gelten muss:

```
Graphics[{{Thickness[0.015], linie2}, {Dashing[{0.05}], linie1},
  {PointSize[0.02], punkt}}, Axes → True, Ticks → {{-1, 1, 2}, {-1, 1}}]
```

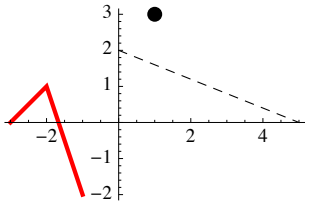


Die Elemente in der Liste der Graphik-Objekte haben also die Form {Direktiven, Primitive}. Weil die verschachtelten Listen im Graphics-Befehl sehr schnell unübersichtlich werden, programmiert man sie meistens vorher. Unten wie man normalerweise vorgeht:

```
pkt = {PointSize[Large], Point[{1, 3}]};
linie1 = {Dashed, Line[{{0, 2}, {5, 0}}]};
linie2 = {Red, Thick, Line[{{-1, -2}, {-2, 1}, {-3, 0}}]};
```

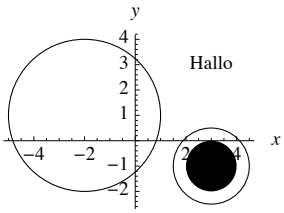
Oben wurde die Primitiven in Variablen abgespeichert. Jetzt werden sie mit dem Graphics-Befehl dargestellt:

```
Graphics[{pkt, linie1, linie2}, Axes -> True]
```

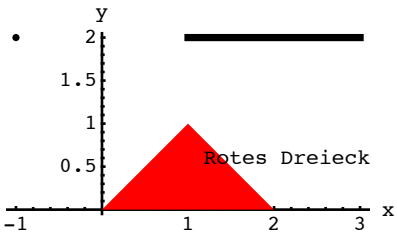


Wie man oben gesehen hat, kann man mehr als eine Direktive pro Graphik-Primitive eingeben.
Jetzt noch 2 weitere neue Graphik-Primitiven, die man oft braucht:

```
txt = Text["Hallo", {3, 3}];
kreis1 = Disk[{3, -1}, 1];
kreise2 = {Circle[{-2, 1}, 3], Circle[{3, -1}, 1.5]};
Graphics[{txt, kreis1, kreise2}, Axes -> True, AspectRatio → Automatic, AxesLabel → {x, y}]
```

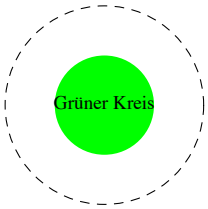


Aufgabe 37:
Versuche das untenstehende Bild herzustellen:

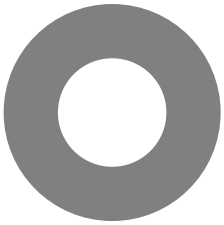


Aufgabe 38:
Versuche die folgenden Bilder von konzentrischen Kreisen herzustellen:
(Tipp: Die Reihenfolge der Graphik-Primitiven im Graphics-Befehl ist nicht unwichtig!)

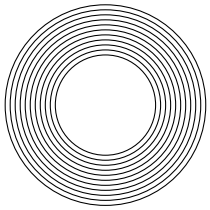
a)



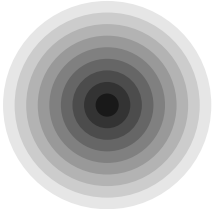
b)



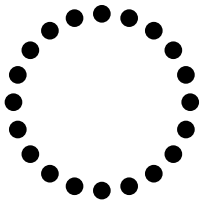
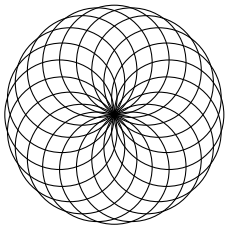
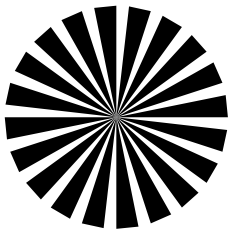
c)



d)



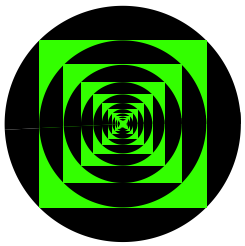
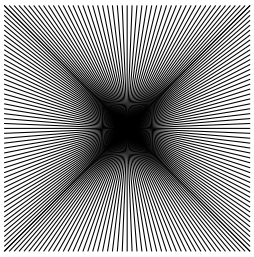
Aufgabe 39:
Versuche die untenstehenden Bilder herzustellen: (Aufgabe c) ist freiwillig!)

a)  b)  c) 

d) Mache aus dem Bild a) einen Film, bei dem sich das Bild um den Mittelpunkt dreht.

Aufgabe 40: (freiwillige Aufgabe!)
Versuche die untenstehenden Bilder herzustellen:

a) b)

Aufgabe 41:
Das Dreieck ABC mit A(1 | 2), B(4 | 7), C(2 | 5) soll eimal an der y-Achse und einmal an der Winkelhalbierenden y = x gespiegelt werden. Lasse *Mathematica* (mit z. B. Replace) die gespiegelten Eckpunkte des Dreiecks berechnen, und stelle die 3 Dreiecke zusammen mit der Spiegelungsgeraden y = x im Koordinatensystem dar.

Aufgabe 42:
Mit Hilfe des Line-Befehls können nun auch Kreise, regelmässige n-Ecke und Funktionen mit dem Graphics-Befehl dargestellt werden, wenn man die Punkt-Liste geschickt mit dem Table-Befehl erzeugt:
a) Zeichne ein regelmässiges 10-Eck.
b) Zeichne einen Kreis.
c) Zeichne die Sinuskurve $y = \sin(x)$ für $-\pi \le x \le 4\pi$.

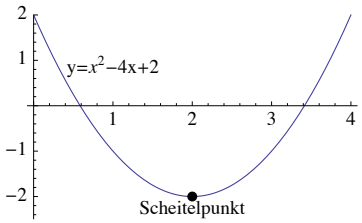
Aufgabe 43:
Mache einen Film eines rotierenden Rads mit Speichen, damit man die Drehung sieht. (Tipp:Manipulate)

Bemerkung:
Graphik-Primitiven kann man auch in die Plot-Befehle einbauen, damit man die Funktionen zum Beispiel anschreiben kann:

```
txt1 = Text["y=x^2-4x+2", {1, 0.9}];
txt2 = Text["Scheitelpunkt", {2, -2.3}];
pkt = {PointSize[0.03], Point[{2, -2}]};
```

Diese Primitiven baut man nun via der Option **Epilog** in den Plot-Befehl ein:

```
Plot[x^2 - 4 x + 2, {x, 0, 4}, Epilog → {txt1, txt2, pkt},
 PlotRange → {-2.5, 2}, ImageSize → 200]
```



Aufgabe 44:

a) `y = sin (x), y = cos (x)` und deren Superposition ("Überlagerung") `y = sin (x) + cos (x)` sollen mit einem Plot-Befehl mit Beschriftung der Funktionen für $-\pi \le x \le 6\pi$ dargestellt werden.

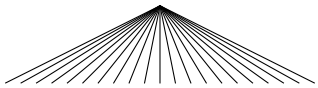
b) Die 2 Funktionen `y = x^3` und `y = 2 x - x^2` schneiden sich in 3 Punkten. Berechne die Schnittpunkte und stelle sie zusammen mit den 2 beschrifteten Graphen dar.

Jetzt noch ein letztes Theoriebeispiel, das einen Graphics-Befehl in einen Manipulate-Befehl einbaut.
Zuerst die Definition der Graphik-Primitiven:

```
pkt = {0, 1};
linien = Line[Table[{{t, 0}, pkt}, {t, -2, 2, 0.2}]];
```

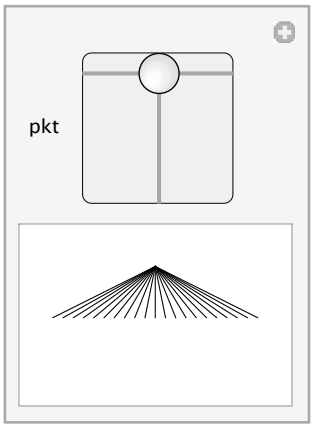
Jetzt werden die Linien graphisch dargestellt:

```
Graphics[linien, PlotRange → {-0.2, 1.2}]
```



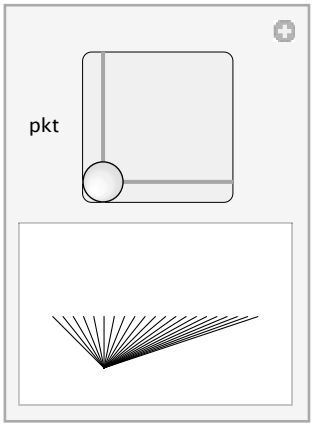
Ich möchte nun die obige Graphik dynamisch machen, und zwar so, dass der Punkt "pkt" an der Spitze (0 | 1) dynamisch mit einem Regler veränderbar wird. Auf den Kopien "Der Plot-Befehl" hatten wir nur Regler für verschiedene Parameter, das heisst für Zahlen. Zum Beispiel war mit `{c, 0, 2, 0.1}` als letztes Argument im Manipulate-Befehl automatisch ein Regler für die Zahl c von 0 bis 2 vorhanden.
Wenn wir nun `{pkt, {-1, -1}, {1, 1}}` im Manipulate setzen, ist automatisch ein zweidimensionaler Regler (x | y) im Rechteck von links unten (-1 | -1) bis rechts oben (1 | 1) definiert:

```
Manipulate[Graphics[linien, PlotRange → {-1.2, 1.2}], {pkt, {-1, -1}, {1, 1}}]
```



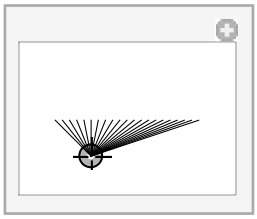
Oben klappt das Dynamische noch nicht. Man kann zwar den Regler verstellen, der Punkt an der Spitze bewegt sich jedoch nicht wie gewünscht! Das liegt daran, dass die Variable "linien" mit dem Punkt (0 | 1) an der Spitze ausserhalb des Manipulate-Befehls definiert wurde und darum immer gleich bleibt.
Nur Befehle innerhalb des Manipulate-Befehls sind dynamisch und können mit dem Regler beeinflusst werden (wenn man den Dynamic-Befehl nicht kennt!).
Wenn wir nicht mit der Variable "linien" arbeiten, sondern alles in den Manipulate-Befehl packen, klappt das Dynamische wie gewünscht:

```
Manipulate[Graphics[{Line[Table[{{t, 0}, pkt}, {t, -2, 2, 0.2}]]}, PlotRange → {-1.2, 1.2}],
 {pkt, {-1, -1}, {1, 1}}]
```



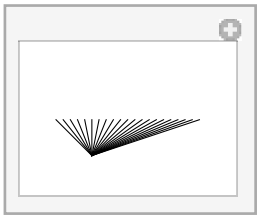
Wenn du oben den Regler mit der Maus bedienst, sollte sich die Spitze in der Graphik dynamisch entsprechend verändern!
Natürlich fragt man sich gleich, wieso man nicht direkt in der Graphik die Spitze mit der Maus verschieben kann. Das ist auch möglich:
Mit `{pkt, {-1, -1}, {1, 1}}` als letztes Argument im Manipulate entsteht automatisch ein zweidimensionaler Regler.
Mit `{pkt, {-1, -1}, {1, 1}, Locator}` jedoch kann man die dynamische Variable "pkt" automatisch mit der Maus in der Graphik verändern:

```
Manipulate[Graphics[{Line[Table[{{t, 0}, pkt}, {t, -2, 2, 0.2}]]}, PlotRange → {-1.2, 1.2}],
 {pkt, {-1, -1}, {1, 1}, Locator}]
```



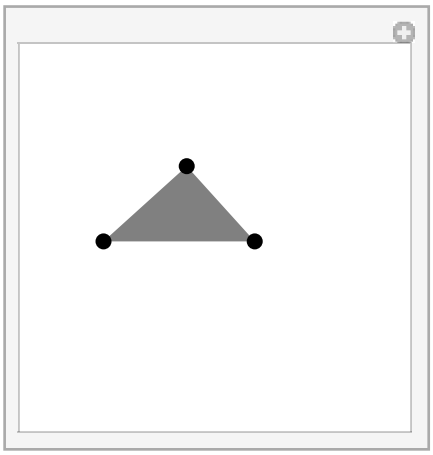
Oben solltest du die Spitze mit der Maus bewegen können!
Wenn man das etwas unschöne Fadenkreuz der dynamischen Variable "pkt" nicht haben will, benutzt man die Option `Appearance → None` des Locators:

```
Manipulate[Graphics[{Line[Table[{{t, 0}, pkt}, {t, -2, 2, 0.2}]]}, PlotRange → {-1.2, 1.2},
   ImageSize → 70], {pkt, {-1, -1}, {1, 1}, Locator, Appearance → None}]
```



Auch oben solltest du die Spitze mit der Maus verändern können.

Aufgabe 45:



a) Versuche die Spitze des linksstehenden Dreiecks dynamisch zu programmieren.
b) Versuche alle 3 Ecken des Dreiecks so zu programmieren, dass sie sich mit der Maus verändern lassen.

Zum Schluss der Kopien eine Aufgabe mit etwas interessanteren Programmen:

Aufgabe 46:
Tippe die unteren Programme ab, und schau, was passiert:

a)
```
Manipulate[Plot[t (x^2 - 1) + (1 - t) (-x^2), {x, -1, 1}, Axes → False,
  Epilog → {Circle[{0, 1}, 3], Disk[{1, 2}, .3], Disk[{-1, 2}, .3], Line[{{0, 0.6}, {0, 1.6}}]},
  PlotRange → {{-3, 3}, {-2, 4}}, AspectRatio -> 1], {t, 0, 1}]
```

b)
```
f[r_] := {Red, Polygon[{{-1, 0}, {r, 0}, {r, 1}, {-1, 1}}]}
txt1 = Text[Style["Auf Wiedersehen!", 30], {0, 0.5}];
Manipulate[Graphics[{txt1, f[c]}, AspectRatio → Automatic,
  PlotRange → {{-1.1, 1.1}, {-0.1, 1.1}}], {c, 1, -1, -0.1}]
```