Übungen zu Computergrundlagen WS 2012/2013

Übungsblatt 9: NumPy und matplotlib

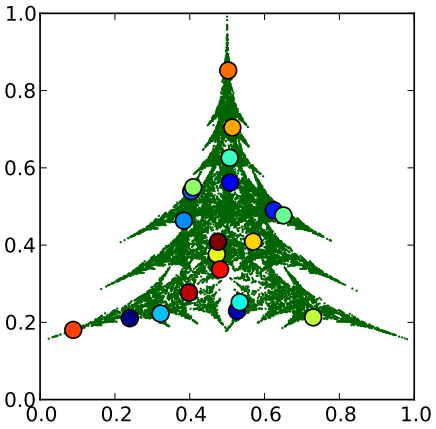
14. 12. 2012

Allgemeine Hinweise

- Abgabetermin für die Lösungen ist **Freitag, 4. 1. 2013, 13:30**, also im neuen Jahr!
- Die Abgabe ist diesmal nur ein Python-Programm, das Du im Anhang einer Email an Deinen Tutor schickst.
- Vergiss nicht, Deinen Namen als Kommentar an den Anfang des Programms zu schreiben.
- Bitte denke daran, Dein Programm ausreichend zu kommentieren!

Ein Fraktalgenerator

Ziel dieses Übungsblatts ist es, den nebenstehenden fraktalen Weihnachtsbaum zu erzeugen. Dazu betrachten wir einen Bildbereich $B = [0,1]^2$, der unseren zukünftigen Weihnachtsbaum enthalten soll. Diesen gestalten wir selbstähnlich, d.h. , wir definieren mehrere Teilbereiche von B, die durch affine Transformationen aus B selber hervorgehen. Affine Transformationen sind dabei solche Abbildungen, die das Quadrat B in ein beliebiges Parallelogramm abbilden. Der Weihnachtsbaum wird also aus mehreren verzerrten Kopien seiner selbst erstellt. So ist zum Beispiel die obere Spitze unseres Weihnachtsbaums wieder ein vollständiger Baum, und die sechs Äste sind Kopien des Baumes, deren Stamm Richtung Mittellinie zeigt.



Um die Transformationen eindeutig zu definieren, genügt es, die Bilder dreier Punkte festzulegen. Der Einfachheit halber nehmen wir die drei Ecken $(0;0)$, $(1;0)$ und $(0;1)$. Dann sind die Transformationen T_i, $i = 1, \ldots, 7$, die unseren selbstähnlichen Weihnachtsbaum beschreiben, gegeben durch

	$T_i(0;0)$	$T_i(1;0)$	$T_i(0;1)$
T_1	(0,6; 0,2)	(0,6; 0,5)	(0,0; 0,0)
T_2	(0,4; 0,2)	(0,4; 0,5)	(1,0; 0,0)
T_3	(0,6; 0,35)	(0,6; 0,65)	(0,15; 0,13)

	$T_i(0;0)$	$T_i(1;0)$	$T_i(0;1)$
T_4	(0,4; 0,35)	(0,4; 0,65)	(0,85; 0,13)
T_5	(0,6; 0,45)	(0,6; 0,75)	(0,2; 0,25)
T_6	(0,4; 0,45)	(0,4; 0,75)	(0,8; 0,25)
T_7	(0,4; 0,5)	(0,6; 0,5),	(0,4; 1)

Wie wird daraus ein Weihnachtsbaum? Dazu bestimmen wir die *Fixpunktmenge* des Systems, also die Menge aller Punkte, die unter jeder der affinen Transformationen T_i auf einen Teilbereich von sich selber abgebildet wird. Das ist ein fraktales Gebilde und analytisch nicht zu bestimmen. Allerdings kann man es numerisch annähern, in dem man auf einen beliebigen Startwert wiederholt zufällige Transformationen T_i anwendet. Nach einigen Anwendungen liegt die Folge der erzeugten Punkte liegt dann dicht an der Fixpunktmenge, unserem Weihnachtsbaum.

Durch geeignete Wahl der Transformationen lassen sich beliebige fraktale Strukturen erzeugen, so zum Beispiel das bekannte Sierpiński-Dreieck oder fraktale Farn. Eine etwas komplexere Variante dieser Methode wird in gerenderten Filmen benutzt, um die Vegetation zu erzeugen.

Aufgabe 9.1: Fraktalgenerator in NumPy (6 Punkte)

9.1.1: Schreibe eine Funktion `calculate_transform(a, b, c)`, die für gegebene (zweidimensionale) Bildvektoren $\vec{a} = T(0;0)$, $\vec{b} = T(1;0)$ und $\vec{c} = T(0;1)$ die affine Transformation T zurückliefert. Wie man sich leicht überlegt, gilt $T(\vec{x}) = A\vec{x} + \vec{a}$, wobei A die Matrix mit den Spalten $\vec{b} - \vec{a}$ und $\vec{c} - \vec{a}$ ist. Daher ist es sinnvoll, die Transformation als Tupel A, $\vec{a}$ zurückzuliefern. Benutze für die Matrix A und den Vektor $\vec{a}$ NumPy-Arrays geeigneter Form. (1 Punkt)

Hinweis: Beachte, dass Matrizen in NumPy *zeilen*weise gegeben werden, hier aber *Spalten* vorgegeben sind.

9.1.2: Schreibe eine Funktion `apply_transform(T, p)`, die die affine Transformation $T = (A, \vec{a})$ auf den Punkt $\vec{p}$ anwendet, also $T(\vec{p}) = A\vec{p}+\vec{a}$ berechnet. Teste `apply_transform` und `calculate_transform`, in dem Du überprüfst, ob für die Transformationen T_1 und T_2 von der Vorderseite die Punkte $(0;0)$, $(1;0)$, und $(0;1)$ tatsächlich auf $\vec{a}, \vec{b}, \vec{c}$ abgebildet werden. (1 Punkt)

Hinweis: Benutze die Vektor-Matrix-Multiplikation und Vektoraddition von NumPy.

9.1.3: In der Datei `~arnolda/computergrundlagen/09/tree.py` sind die 7 Weihnachtsbaumtransformationen von der Vorderseite bereits als Python-Liste von Bildvektoren $\vec{a}, \vec{b}, \vec{c}$ gegeben. Benutze `calculate_transform`, um diese in eine Liste von Transformationen $A, \vec{a}$ zu verwandeln. (1 Punkt)

9.1.4: Schreibe eine Funktion `calculate_attractor(trafos, N)`, die eine Liste von N zweidimensionalen Punkten aus einer Umgebung des Attraktors der Transformationen aus der Liste `trafos` erzeugt. Dazu startest Du bei $\vec{p}_0 = (0;0)$, und wendest zunächst 100-mal zufällig gewählte Transformationen aus der Liste `trafos` auf $\vec{p}_i$ an. Es gilt also $\vec{p}_{i+1} = T_k(\vec{p}_i)$ mit k Zufallszahl aus 1,...,K, wobei K die Länge der Liste `trafos` ist. Wende dann weitere N-mal die Transformation an, und gebe die dann erzeugten Punkte $\vec{p}_i$, $i = 100, \ldots, N + 99$ als Liste zurück. (3 Punkte)

Hinweis: Du kannst Dich an der Random Walk-Funktion vom letzten Übungsblatt orientieren, bei der ja ebenfalls ein Punkt zufällig verändert wurde. Beachte aber, dass NumPy-Arrays durch `p.copy()` kopiert werden! Nimm nicht an, dass die Liste `trafos` sieben Elemente hat, sondern benutze `len`.

Aufgabe 9.2: Darstellung mittels `matplotlib` (4 Punkte)

9.2.1: Benutze nun `calculate_attractor`, um Dir eine Liste von 10.000 Punkten des Attraktors zu erzeugen. Zeichne diese mittels `matplotlib` als grüne Punkte in einen Graphen ein. Die Achsen sollen dabei unseren gesamten Bildbereich $[0, 1]^2$ darstellen. (3 Punkte)

Hinweis: Deine Funktion `calculate_attractor` wird wahrscheinlich eine Liste von zweidimensionalen Punkten zurückgeben. `matplotlib` benötigt jedoch zwei Listen von x- bzw. y-Koordinaten. Schreibe daher eine Schleife, die die Koordinaten der Punkte aus der Liste auf zwei Listen verteilt.

9.2.2: Um den Weihnachtsbaum zu dekorieren, benutzen wir nun die Funktion `scatter` der `matplotlib`, die erlaubt, jeden Punkt einer Liste anders einzufärben. Dazu zeichne 10 der Attraktorpunkte, und wähle zum Beispiel einen fortlaufendenen Index als Farbcode. (1 Punkt)

Extraaufgabe

Wer Lust hat, kann gerne mal mit den Transformationen experimentieren, um andere Attraktoren zu erzeugen. Eine suggestive Einfärbung ist dabei natürlich wichtig! Um zu verstehen, was da passiert, ist es hilfreich, sich die Bildbereiche der Transformationen darzustellen, in dem man die Bilder der Punkte $(0;0)$, $(1;0)$, $(1;1)$ und $(1;0)$ als Parallelogramm einzeichnet.