UNIVERSITÄT BASEL

University of Basel

Dozent
Prof. Dr. Thomas Vetter
Departement
Mathematik und Informatik
Spiegelgasse 1
CH – 4051 Basel

Assistenten
Marcel Lüthi

Tutoren
Pascal Mafli
Loris Sauter
Linard Schwendener
Clemens Büchner
Florian Spiess
Jonathan Aellen
Lukas Stöckli

Erweiterte Grundlagen der Programmierung (45398-01)

Blatt 11 [8 Punkte]

Vorbesprechung	03. - 07. Dez
Abgabe	10. - 14. Dez

Wir empfehlen Ihnen, dass Sie im Buch "Sprechen Sie Java" das Kapitel 15 lesen, bevor Sie beginnen die Übungen zu lösen.

Aufgabe 1 - Der Schwarm [4 Punkte]

Auf diesem Blatt werden Sie eine Schwarmsimulation implementieren. Nutzen Sie dazu den Code, den Sie im Zip-Archiv für diese Übung (im Verzeichnis `boids`) finden. Lesen Sie dazu bitte
`http://www.red3d.com/cwr/boids/`.

Die Grundidee ist, eine Menge von Individuen zu simulieren, die folgenden einfachen Regeln folgen:

(a) Cohesion: Versuche, dorthin zu gehen wo deine Nachbarn sind

(b) Alignment: Passe deine Geschwindigkeit und Richtung deinen Nachbarn an

(c) Separation: Vermeide Zusammenstösse mit deinen Nachbarn

(d) Target: Versuche, zu einem (oder mehreren) Zielen zu gelangen

Mit der richtigen Wahl von Parametern kann diese kleine Regelbasis zu komplexem, Schwarmartigen verhalten führen.

Der Code von der Übungswebseite enthält das Framework für solch eine Simulation. Verschieden, durch `/** IMPLEMENT THIS **/` markierte Teile des Framework sind noch nicht implementiert. Lesen und Verstehen Sie den Code, und Ergänzen Sie die fehlenden Teile. Spielen Sie dann mit der Simulation.

Aufgabe 2 - Faltung [4 Punkte]

In dieser Aufgabe soll eine kleine Bildbearbeitungssoftware geschrieben werden. Nutzen Sie dazu den Code, den Sie im Zip-Archiv für diese Übung finden.

Die Software soll verschiedene 2D-Faltungskerne auf ein Bild anwenden können. Eine Faltungskern ist im Rahmen dieser Übung die Information die in einem der `.txt`-Files steht.

Stellen Sie sich den Faltungskern als Zahlengitter vor. Das Gitter hat in jeder Dimension eine ungerade Anzahl von Zellen. Zusätzlich zum Gitter besteht ein Faltungskern noch aus zwei Zahlen, einer Normierungskonstante (`norm`) und einer Zuwachskonstante (`gain`).

Implementieren Sie die Klasse `FaltungsKern` welche ein `.txt`-File einlesen kann. Folgende Information ist darin gespeichert:

- `size` - die Höhe und Breite des Kerns
- `norm` - Zähler und Nenner des Bruchs für die Normierungskonstante
- `gain` - die Zuwachskonstante
- `data` - der eigentliche Kern

Wenn Sie die Zahlen (`data`) einlesen und im 2D-Array ablegen, teilen Sie diese gleich durch die Normierungskonstante.

Implementieren Sie die Methode `falten`. Eine Faltung wird als atomare Operation angewendet. Wie beim Game-Of-Life aus den Übungen wird das Ausgangsbild während dem Falten nicht verändert. Jedes Teilergebniss wird in einem temporären Bild zwischengespeichert und am Ende wird das originale durch das temporäre Bild überschrieben.

Ein Faltungskern wird an jeder Stelle auf das Bild angewendet um das gefaltete Bild zu berechnen. Schauen Sie sich die unten folgende Abbildung an welche die Anwendung eines Kerns auf eine Stelle im Bild illustriert. Ihre Aufgabe ist es nun die Formeln zu verstehen und zu implementieren. Dabei soll die Faltung für jeden Farbkanal getrennt berechnet werden.

Soll der Kern auf eine Stelle (x,y) im Bild angewendet werden, wird der Mittelpunkt des Kerns auf diese Stelle "verschoben". Die aufeinander liegenden Zellen des Bildes und des Kerns werden miteinander multipliziert. Die Summe über alle Ergebnisse der Multiplikationen wird zur Konstanten `gain` addiert. Die so gewonnene Zahl wird an der Stelle (x,y) in dem temporären Bild gespeichert.

Wenn der Kern am Rand des Bildes angewendet wird, so dass "unter" einer Zelle des Kerns kein Pixel des Bildes zu liegen kommt, wird der nächste Pixel im Bild verwendet. Man kann sich auch vorstellen, dass nach aussen die Randpixel des Bildes wiederholt werden.

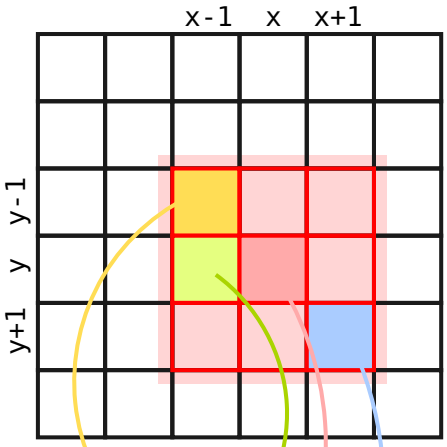
Schreiben Sie ein Programm `BasicImageProcessing` welches die Faltungskerne verwendet um ein Bild zu falten. Sie dürfen dazu die Ihnen bekannte Klasse `ImageWindow` verwenden.

Anforderung an das UserInterface des Programms sind:

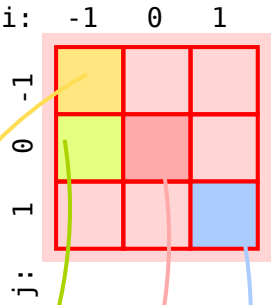
- Der Benutzer kann ein Bild laden.
- Der Benutzer kann einen Faltungskern auswählen.
- Das Programm gibt dem Benutzer Feadback über auftretende Fehler und behandelt diese so, dass es zu keinem Absturz kommt. (Beispiele: Bild nicht gefunden, Kern nicht vorhanden, Kern zuerst ausgewählt und noch kein Bild geladen)

Dabei spielt es keine Rolle ob der Benutzer über Menüs und Dialoge die Aktionen auswählt oder ob der Benutzer das Programm über die Kommandozeile steuert.

Bild: img



Faltungskern: f



Gain: gain

$$
\begin{aligned}
t[x][y] = & img[x-1][y-1] * f[-1][-1] + \dots \\
& img[x-1][y] * f[-1][0] + \dots \\
& \dots \\
& img[x][y] * f[0][0] + \dots \\
& \dots \\
& img[x+1][y+1] * f[1][1] + gain
\end{aligned}
$$

$$
t[x][y] = gain + \sum_{i=-d}^{d} \sum_{j=-d}^{d} img[x+i, y+j] * f[i,j]
$$

Target: t