ulm university universität
uulm

Institut für Numerische Mathematik

Dr. Andreas F. Borchert

24. Juni 2016
Blatt 8

Digitale Typografie (SS 2016)

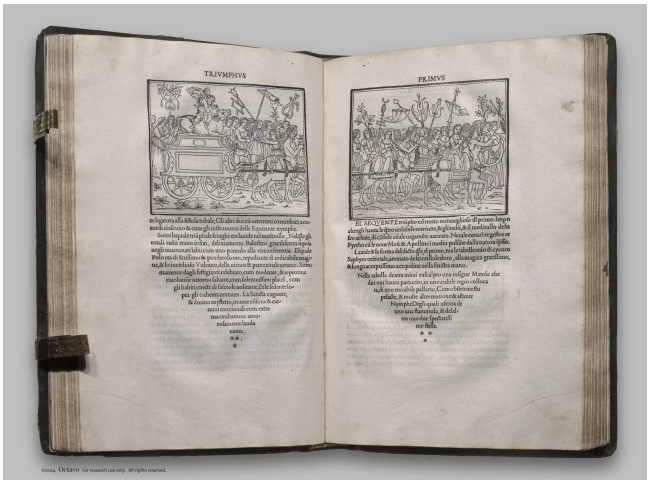
Abgabe bis zum 1. Juli 2016, 10:00 Uhr

Lernziele:

- Entwicklung eines speziell angepassten Algorithmus zur Zerlegung eines Paragraphen in Zeilen

Aufgabe 10: Inszenierende Typografie

Aldus Manutius (1449–1515) war ein venezianischer Publizist und Buchdrucker der Renaissance, der sich insbesondere mit dem Druck des 1499 erschienenen Werks *Hypnerotomachia Poliphili* unsterblich gemacht hat. In vielfacher Sicht wurde die Typografie dieses Werks zu einem der Höhepunkte der damaligen Buchgestaltung. Zu den Besonderheiten gehört der frühe Einsatz der „inszenierenden Typografie“, die, wenn sie gekonnt eingesetzt wird, das Lesevergnügen im besonderen Maße steigert. So ließ beispielsweise Manutius einige der Textabschnitte nach unten hin dreiecksmäßig abschließen. Hier ist eine Doppelseite, die diese Technik gleich zweifach demonstriert:



Aus dem Digitalisat unter `http://www.rarebookroom.org/Control/colhyp/index.html?page=84`

Ziel dieser Aufgabe ist es, einen Zerlegungsalgorithmus für Paragraphen zu entwickeln, der den Abschluss eines Paragraphen als Dreieck gestaltet. Das Problem ist hier, dass selbst bei

dem allgemeinen Knuth'schen Algorithmus mit variablen Zeilenlängen diese fest vorgegeben sein müssen, d.h. wir müssen zu Beginn wissen, wieviel Platz für die erste Zeile vorgesehen ist, für die zweite Zeile usw. Bei einem Dreiecksabschluss müssen wir jedoch mit der Zerlegung hinten bei dem Dreieck beginnen. Nur das, was sich nicht im Dreieck unterbringen lässt, kann dann konventionell zerlegt werden. Entsprechend sind uns die letzten Zeilenlängen bekannt (in Abhängigkeit von der vollen Breite des Paragraphen), aber wir wissen zu Beginn nicht ohne weiteres, wieviel Zeilen wir insgesamt haben bzw. wann das Dreieck beginnt.

Hier ist eine Demonstration mit dem ersten Paragraphen aus der Kurzgeschichte *The Black Cat* von Edgar Allan Poe:

> For the most wild, yet most homely narrative which I am about to pen, I neither expect nor solicit belief. Mad indeed would I be to expect it, in a case where my very senses reject their own evidence. Yet, mad am I not—and very surely do I not dream. But to-morrow I die, and to-day I would unburden my soul. My immediate purpose is to place before the world, plainly, succinctly, and without comment, a series of mere household events. In their consequences, these events have terrified—have tortured—have destroyed me. Yet I will not attempt to expound them. To me, they have presented little but horror—to many they will seem less terrible than baroques. Hereafter, perhaps, some intellect may be found which will reduce my phantasm to the commonplace—some intellect more calm, more logical, and far less excitable than my own, which will perceive, in the circumstances I detail with awe, nothing more than an ordinary succession of very natural causes and effects.

Zu entwickeln ist konkret eine auf der typografischen Vorlesungsbibliothek basierende Java-Klasse *TriangleLineBreaker*, die eine Erweiterung des *BasicLineBreaker* ist. Diese müsste in der Methode *breakParagraph* wie folgt vorgehen:

- Die gesamte horizontale Sequenz des Paragraphen ist in eine Liste einzelner Wörter und der jeweils zugehörigen Sollbruchstellen zu zerlegen. Diese können dann in eine neue Liste eingefügt werden, so dass es möglich wird, sie in umgekehrter Reihenfolge zu durchlaufen. Dies erlaubt es uns, mit der Spitze des Dreiecks zu beginnen. Dabei ist zu beachten, dass die letzte Sollbruchstelle ersetzt werden sollte, da wir dort keine unendlich dehnbare Dehnfuge gebrauchen können. Stattdessen kann eine geeignete Instanz von *Penalty* verwendet werden.

- Dann können wir mit einem umgedrehten *First-Fit*-Algorithmus sukzessive die einzelnen Zeilen des Dreiecks (als jeweilige horizontale Sequenzen) erzeugen, wobei wir mit der unteren Dreiecksspitze beginnen. Sobald eine Zeile fertig ist, sollte sie an die entsprechende Zeilenlänge innerhalb des Dreiecks mit Hilfe des *HorizontalFitter* angepasst werden, um sie dann mit Hilfe zweier umgebender Dehnfugen unendlicher Dehnbarkeit an die normale Paragraphenbreite anzupassen.
- Sobald das Dreieck fertig ist, können Sie den Rest dem *TotalFitLineBreaker* übergeben. Achten Sie dabei darauf, dass die letzte Sollbruchstelle des regulären Paragraphen nicht dehnbar ist.
- Wenn Sie zwei Objekte des Typs *VerticalBox* aufeinanderstapeln möchten (regulär formatierter Paragraph gefolgt von dem Dreieck), dann sollten Sie hierfür die Klasse *VerticalStack* verwenden, die frisch in der Vorlesungsbibliothek aufgenommen wurde. Um diese Klasse verwenden zu können, sollten Sie also das aktualisierte JAR-Archiv herunterladen.

Um Ihre Klasse *TriangleLineBreaker* zu testen, empfiehlt es sich, von der Vorlesungsseite entweder das Testprogramm *TestPar.java* oder *TestHyphenatedPar.java* zu nehmen und unmittelbar nach dem Erzeugen des Objekts *lbf* folgende Zeile einzufügen:

```
lbf.add("triangle", new TriangleLineBreaker.Constructor());
```

Damit dies geht, sollten Sie folgende Unterklasse bei *TriangleLineBreaker* einfügen:

```
public static class Constructor
        implements LineBreakerFactory.Constructor {
    public LineBreaker create() {
        return new TriangleLineBreaker();
    }
} // class Constructor
```

Sie finden ähnliche Konstrukte bei den anderen Klassen, die zeilenbrechende Algorithmen implementieren. Wenn Sie das tun, bietet Ihr Testprogramm Ihnen neben „first“, „best“ und „total“ nun auch noch „triangle“ als Algorithmus an.

Für die schmaleren Zeilen des Dreiecks ist es sehr hilfreich, wenn möglichst viele Trennstellen zur Verfügung stehen. Es lohnt sich deswegen, den Trennungsalgorithmus zu verwenden, den wir in der nächsten Vorlesung näher kennenlernen werden. Sie benötigen hierfür Trennungsregeln. Sie finden diese bei jeder TEX-Distribution, bei uns auf der Thales unter dem Pfadnamen */usr/local/texlive/2015/texmf-dist/tex/generic/hyphen/hyphen.tex*. Kopieren Sie diese Datei, und behalten Sie davon nur die Zeilen, die *innerhalb* der *patterns*-Auflistung gegeben sind, d.h. nur die Zeilen 6 bis 4452.

Wenn Sie mit Ihrer Lösung fertig sind, sollten Sie alle notwendigen Dateien mitsamt den Quellen in ein ausführbares Java-Archiv *table.jar* packen und zusammen mit einer beispielhaften Ausgabe Ihres Programms mit dem Kommando

```
submit typo 10 triangle.jar triangle.ps
```

auf der Thales einreichen.

Viel Erfolg!