Visualisierung von Algorithmen und Datenstrukturen

TECHNISCHE UNIVERSITÄT DARMSTADT

Abschlussbericht
Gruppe 33: Benedikt Lins (1799381) und Stefan Thaut (1800351)
Fachbereich 20 - Informatik
5. September 2018

1 Lernerfolge

In diesem Praktikum sollten verschiedene Datenstrukturen und Algorithmen visualisiert werden. Dazu sollte das in Java geschriebene Framework *Animal* genutzt werden. Dieses Framework bietet verschiedene grafische Primitive an, um sowohl den Zustand einer Datenstruktur oder Algorithmus als auch die verschiedenen Vorgänge während einer Operation der Datenstruktur oder der Algorithmen visualisieren zu können.

Neben einer Auffrischung in objekt-orientierter Programmierung sowie der Funktionsweise verschiedener Datenstrukturen und Algorithmen - in unserem Fall die Dial-Implementierung der beschränkten monotonen Prioritätswarteschlange und dem Edmonds-Karp Algorithmus - haben wir den Umgang mit Animal gelernt und damit auch, was in einem Algorithmus alles visualisiert werden muss. Zwar sollten selbst kleinere Veränderungen des Zustands hervorgehoben werden, um das Verständnis des Betrachters zu fördern. Dabei muss jedoch auch auf ein Maß an Übersichtlichkeit geachtet werden, damit der Betrachter die Möglichkeit hat, das Gesehene schnell zu verstehen.
Denn niemandem ist geholfen, wenn die Visualisierung alle Vorgänge das Algorithmus gesamtheitlich und detailliert darstellt, es aber für den einfachen Betrachter schwierig ist, die Vorgänge zu erfassen und zu verstehen.

2 Bewertung von Animal

Die Idee von Animal ist, verschiedene Algorithmen und Datenstrukturen verständlich zu visualisieren, um das Erlernen und Verstehen dieser einfacher zu gestalten. Dabei setzt Animal auf dynamische Animationen, welche einen gewählten Algorithmus auf einem konkreten Beispiel ausführen. Dies hat den Vorteil, dass der Lernende nicht direkt mit der mathematischen Definition des Algorithmus konfrontiert wird und zuerst eine Intuition für die Funktionsweise erwirbt. Dies ersetzt leider nicht vollständig das Studium der konkreten Definition, vereinfacht aber deren Verständis deutlich.
Das Animal-Framework bietet eine ausreichende Menge an Visualisierungsmöglichkeiten für die grundlegende Darstellung von Datenstrukturen und Algorithmen an. Verbesserungsmöglichkeiten sehen wir daher nur punktuell.

2.1 Algorithmen-Bugfixing

Während der Bearbeitung des Praktikums haben wir verschiedene Algorithmen ausgewählt und die dazugehörige Animation betrachtet. Dabei ist uns aufgefallen, dass in der aktuellen Version von Animal einige Generatoren existieren, welche nicht funktionieren. Auch halten sich anscheinend nicht alle implementierten Generatoren an die Vorgaben, welche wir im Rahmen des Praktikums erhalten haben. Zum Beispiel existieren keine Einleitungsfolien oder Texte und Formen überlagern sich, sodass diese nur schwer bis gar nicht zu lesen sind. Darüber hinaus existieren Generatoren, die nicht dynamisch gestaltet sind, sodass der Betrachter auf der einen Seite keine Parameter verändern kann oder die Animation selbst keine Statusveränderungen anzeigt. Es wäre hilfreich, wenn solche verbesserungsfähigen Generatoren nicht in Animal integriert werden, damit die allgemein hohe Qualität von Animal nicht gesenkt wird und zukünftige Entwickler von weiteren Generatoren sich an diesen kein negatives Beispiel nehmen.

2.2 Namensgebung der Generatoren

Die Namen der Generatoren können vom jeweiligen Entwickler selbst gewählt werden. Dies kann dazu führen, dass die Namen uneindeutig sind. Auch kann es dadurch vorkommen, dass zu einem Algorithmus oder einer Datenstruktur zwei oder mehr Generatoren existieren, was für den Betrachter verwirrend sein kann. Wenn zwei Algorithmen den selben

Namen aufweisen, dann sind meist die Sprachen und/oder die Programmiersprachen unterschiedlich. Im Sinne der Übersicht und der einfachen Bedienung wäre es sinnvoll, dass zu jedem konkreten Algorithmus nur ein Generator existiert, welcher dann durch Angabe der gewünschten Sprache und Programmiersprache konkretisiert wird. Eine solche Möglichkeit existiert bereits für die Sprache, weshalb eine Erweiterung für die Programmiersprache nicht allzu aufwendig erscheint. Dabei sollte dann auch versucht werden, die doppelten Generatoren zusammenzufassen. Diese vorgeschlagenen Verbesserungen, würden die Recherche nach dem gewünschten Generator deutlich vereinfachen (vgl. Abschnitt 2.6).

2.3 Dokumentation

Bei einigen Klassen und Methoden, die wir genutzt haben, fehlen teilweise Informationen oder die Dokumentation gänzlich. Ein gutes Beispiel dafür sind die Parameter *direction* und *moveType* der Methoden *moveVia* und *moveTo*. Hier ist zwar beschrieben, dass die beiden Parameter die Richtung, in der das Primitiv bewegt werden soll, bzw. den Typ der Bewegung angeben sollen. Leider sind aber nicht die möglichen Werte angegeben, die der jeweilige String annehmen kann. Dies verzögert die Entwicklung, da zunächst entweder per Trial and Error getestet werden muss, was man einsetzen kann, oder sich andere Generatoren anschauen muss, wie der jeweilige Code aussehen muss.

2.4 Label für die Inhaltsangabe

Während der Überarbeitung der Algorithmen, um Einträge in der Inhaltsangabe einzufügen, sind wir auf die Kuriosität gestoßen, dass man den Titel für den Schritt, den man in der Inhaltsangabe aufführen möchte, in der nextStep()-Methode **nach** dem jeweiligen Schritt angeben muss:

Listing 1: Code-Beispiel für die Angabe des Titels nach dem Code

```
lang.nextStep();

//Code for important Step

lang.nextStep("Title␣for␣important␣Step");
```

Dies ist für den Programmierer eher unintuitiv, da man davon ausgeht, dass das was nach dem Aufruf folgt, unter dem entsprechenden Titel zu finden ist. Wir vermuten, dass dies daran liegt, dass man für den ersten Schritt (i.d.R. die Titelfolie) die nextStep()-Methode nicht aufruft und so keinen Eintrag in der Inhaltsangabe erzeugen könnte.

2.5 Erweiterungsmöglichkeit der Primitive

Sowohl bei der Implementierung unserer Datenstruktur wie auch des Algorithmus sind wir auf Grenzen der grafischen Primitive von Animal gestoßen. Ein Beispiel dafür ist die Achsenorientierung der Ellipse. Diese kann man nur horizontal und vertikal verändern und orientieren. Die rechte Ellipse in Abbildung 1 ist damit nicht darstellbar. Dies stellt im weiteren Verlauf darüber hinaus eine Einschränkung für Kreisbögen und auch für Bewegungen entlang eines Kreisbogens dar.

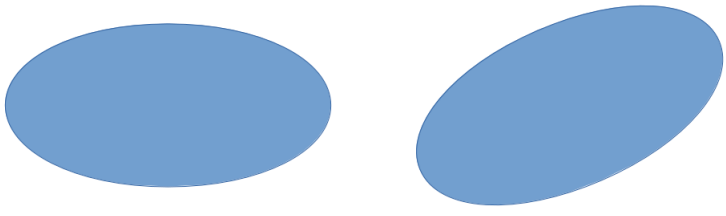

Abbildung 1: Die linke Ellipse ist achsenorientiert und kann von Animal im Gegensatz zur rechten Ellipse dargestellt werden

Insbesondere für Graphalgorithmen wären beliebige Kreisbögen, also solche, die sich aus nicht achsenorientierten Ellipsen ergeben, vorteilhaft. In der momentanen Version von Animal werden bidirektionale Kanten zwischen zwei Knoten als zwei parallel verlaufende Pfeile gezeichnet, was den Graph je nach Größe und Architektur vor allem in Bezug auf die Label unübersichtlich machen kann. Ein Kreisbogen könnte durch das Auseinanderziehen der beiden Kanten und dem dadurch geschaffenen Platz zwischen ihnen der Übersicht dienlich sein.

Eine andere Lösung dieses Problems könnte das Ermöglichen von Tooltips sein, die angezeigt werden, sobald der Betrachter mit dem Zeiger über eine entsprechende Kante fährt und so zusätzliche Informationen über das Element erhält.

Es ist offensichtlich, dass eine solche Erweiterung der Primitive nicht in einer gewissen Volständigkeit der Zeichenmöglichkeiten und insbesondere nicht in einem vertretbaren Aufwand resultiert. Daher wäre es überlegenswert, die Primitive erweiterbar zu gestalten, sodass zukünftige Entwickler eigene Primitive der API hinzufügen können und für die Konsistenz die neuen Primitive von einer abstrakten Oberklasse erben, bzw. eine Schnittstelle implementieren müssen. So kann jeder Entwickler nach eigenem Belieben und nach eigenen Ansprüchen den Aufwand, den er in die Visualisierung steckt, selbst bestimmen.

2.6 Übersicht der Generatoren

Bei der Recherche bereits vorhandener Generatoren ist uns aufgefallen, dass die Suche nach bestimmten Generatoren in der Übersicht in Animal schwierig ist. Auch wenn für alle Generatoren eine Kurzbeschreibung in der Übersicht angezeigt wird, ist diese manchmal nur wenig aussagend oder unverständlich formuliert, sodass man den Generator erst einmal starten muss, um zu erkennen, was der Algorithmus tatsächlich leisten soll. Auch die Namen sind manchmal unglücklich gewählt, sodass man bei manchen Algorithmen fälschlicherweise einen anderen Algorithmus erwartet als den, den man dann letztendlich startet.
Darüber hinaus ist es bei manchen Algorithmen schwer, diese in eine passende Kategorie einzuordnen. Der Algorithmus von Edmonds und Karp beispielsweise kann sowohl in die Kategorie der Graphenalgorithmen als auch in Algorithmen für Netzwerke eingeordnet werden. Hier könnte eine konsistentere Kategorisierung vorgenommen werden.

2.7 Kapselung

Wie in unserem Fall, wird es bestimmt auch andere Entwickler geben, die bereits einen Algorithmus oder eine Datenstruktur implementiert haben, die sie dann mittels Animal visualisieren möchten. Mithilfe der aktuellen Version von Animal ist es jedoch nicht möglich, die Implementierung gleichzeitig sowohl als Datenstruktur bzw. als Algorithmus zu verwenden als auch mittels Animal zu visualisieren. Dies liegt daran, dass der Animalcode innerhalb der Implementierung der Datenstruktur bzw. des Algorithmus liegen muss.
Wünschenswert wäre hier eine Kapselung des Animalcodes von dem Code der Datenstruktur bzw. des Algorithmus, sodass der Code selbst ausführbar und nutzbar bleibt, aber dennoch visualisiert werden kann, ähnlich dem MVC-Muster. Es ist offensichtlich, dass der Code in jedem Fall angepasst werden muss, da der Zustand der Datenstruktur bzw. des Algorithmus zu jeder Zeit abrufbar sein muss. Im schlimmsten Fall muss jede Code-Zeile visualisiert werden. Denkbar wäre dennoch eine Art Listener-Muster, bei dem dann der Code so angepasst werden kann, dass nach wichtigen Code-Teilen, oder auch nach jeder Code-Zeile der Listener, in diesem Fall die Animal-Visualisierung, über den neuen Zustand informiert wird.

3 Bearbeitung des Praktikums

Wir haben uns für eine Bearbeitung als Team entschieden. Die Aufgaben haben wir dabei nicht aufgeteilt, sondern bei teilweise mehrmaligen wöchentlichen Treffen den Entwurf, die Implementierung und das Verfeinern der Generatoren gemeinsam vorgenommen haben.
In unserem Fall war es von Vorteil, dass wir den gewählten Algorithmus und die gewählte Datenstruktur bereits für ein anderes Fach implementiert hatten und so die Grundstruktur schon vorhanden war. Nachdem wir uns auf die Visualisierung geeinigt hatten, konnten wir in die Grundgerüste fast nahtlos die Animal-Codesequenzen einfügen und so schon eine akzeptable Darstellung erreichen, ohne dass wir bis dahin schon viel Detailtiefe hinzufügen mussten.

Während unserer ersten Treffen haben wir größtenteils die detailliertere explizite Darstellungsweise ausgearbeitet und uns darüber hinaus in Animal und die Algoanim-API eingearbeitet. Im weiteren Verlauf haben wir dann den bestehenden Code um die Visualisierung in Animal inkrementell erweitert. In einem weiteren Schritt und nach erfolgten Testaten mit dem betreuenden Tutor haben wir unsere Visualisierungen verfeinert und verschiedene Zusatzfunktionen wie die Übersetzung ins Englische oder die Vergabe von Titeln für wichtige Schritte der Visualisierung hinzugefügt.

4 Bewertung der Organisation

Die Organisation des Praktikums kann anstandslos als einwandfrei bezeichnet werden. Die Aufgabenstellungen sind offen gestellt, sodass man sich die Bearbeitungszeit des Praktikums vollkommen frei einteilen kann. So ist es sowohl möglich, das Praktikum in den ersten Wochen des Semesters zu absolvieren als auch die Prüfungen des Semesters abzuwarten, um sich besser auf diese vorbereiten zu können und das Praktikum in deren Anschluss zu bearbeiten.
Eine weitere Unterstützung der Studenten bietet die Möglichkeit, zwischen Online- und Präsenztestaten zu wählen. So

werden Studenten, die möglicherweise nicht in der näheren Umgebung der Universität wohnen oder nicht sehr flexibel in der Terminauswahl sind, nicht gezwungen, in der Universität erscheinen zu müssen. Auf der anderen Seite wird Studenten, die Hilfe vor Ort benötigen, durch die Testate die Möglichkeit gegeben, ein Gespräch zu führen.

Die Kommunikation über Moodle erfolgte reibungslos. Man erhielt üblicherweise sehr schnell eine, insbesondere hilfreiche, Antwort oder wusste wahlweise, dass der betreuende Tutor für einen gewissen Zeitraum abwesend ist. Der Tutor war darüber hinaus äußerst flexibel, was Termine anbelangt, sodass er selbst weitere Termine ermöglichte, obwohl bereits alle Termine, die in Moodle aufgeführt waren, belegt waren.

Die Praktikumsblätter waren für den Einstieg hilfreich. Für die spätere Entwicklung der eigenen Generatoren wäre aber vielleicht eine Art Wiki hilfreicher, in dem man dann speziell nach den Themen suchen kann, die man gerade benötigt. Hier war die Suche in den Praktikumsblättern eher mühsam. Manche Beispiele der Praktikumsblätter wären noch hilfreicher gewesen, wenn sie konkreter auf die Animal-API eingegangen wären. Ein gutes Beispiel hierfür ist die Funktionsweise des Translators, der in den Blättern anhand von GUI-Komponenten erklärt wird, sodass wir uns den Code anderer Generatoren anschauen mussten, um die genaue Anwendung des Translators in Bezug auf Animal verstehen zu können.

Abschließend kann man zusammenfassen, dass das Praktikum viel Spaß gemacht hat, da es eine breite Sammlung von verschiedenen Themen des Informatikstudiums abdeckt, in die man sich wieder einmal hineintasten konnte. Für die Zukunft kann man sich wünschen, dass Animal und die implementierten Algorithmen auch von anderen Fachgruppen in deren Lehre eingesetzt werden und Animal dadurch noch populärer wird.