Abschlussbericht

Christian Richter & Andre Challier

4. Oktober 2017

1 ALLGEMEINES

Bei diesem Praktikum haben wir gelernt Algorithmen mit *AnimalScript* und der *AnimalAPI* zu visualisieren und in *Animal* einzubinden. Innerhalb des Praktikums konnten wir als Onlinegruppe flexibel und eigenständig Arbeiten. Die Aufgabenblätter boten einen guten Einstieg und die Aufgaben hatten einen angemessenen Umfang. Schon die Einführungspräsentation hat einen guten Ausblick auf den Ablauf des Praktikums und die Funktionsweise von *Animal* gegeben.

Während des Praktikums haben wir eine *Multi-Level-Scheduling-Queue* und den *Local Outlier Factor* visualisiert. Bei der Queue war die größte Aufgabe gleichzeitig darzustellen welche Attribute die ankommenden Objekte haben und nach welchen Vorgaben sie in den verschiedenen Queues sortiert und gescheduled werden. Da es unsere erste Arbeit mit *Animal* war haben wir die Attribute schlicht in einer Tabelle dargestellt und die Queues als Arrays mit festen Längen. Dabei haben wir gelernt, dass man zur Visualisierung eines Algorithmus ein noch tieferes Verständnis der Abläufe benötigt als für die Anwendung. Besonders das Einschätzen ob eine Darstellung für einen Betrachter nachvollziehbar ist oder nicht ist uns häufig schwer gefallen. Meist haben wir uns dann dazu entschieden in einem Textfeld einen Hinweistext einzufügen um die Abläufe zu erklären.

2 Ablauf

Eine der ersten Aufgaben des Praktikums ist es, einen Algorithmus zu finden der noch nicht umgesetzt wurde. Hier haben wir die „Liste von Algorithmen“ aus der Wikipedia verwenden müssen um noch neue Algorithmen zu finden. Hier bietet es sich eventuell an, den Studenten eine Liste von Algorithmen anzubieten, die noch nicht umgesetzt wurden.

Bei den ersten Übungsblättern mussten wir leider sehr lange auf die Bewertung warten, sodass wir schon vorher begonnen haben die nächsten Aufgaben zu lösen. Leider hat das dazu geführt, dass sich die Fehler die wir in den ersten Aufgaben gemacht haben in den weiteren Aufgaben fortgesetzt haben. Nachdem wir die Bewertungen dazu bekommen hatten war es ziemlich aufwändig diese Fehler überall zu korrigieren. Hier wäre es praktisch eine grobe Frist für die Bewertungen zu haben, um sich besser organisieren zu können.

Bei den einführenden Übungsaufgaben mit *AnimalScript* haben wir direkt mit einem der Algorithmen die wir uns ausgesucht hatten angefangen. Das führte bei uns leider dazu, dass wir viele Möglichkeiten die *Animal* bietet im ersten Generator nur schwer umsetzen konnten. Zum Einen weil wir das simple Konstrukt aus der ersten AnimalScript-Übung für die weiteren Übungen wiederverwendet hatten und zum anderen weil wir für viele Erweiterungen (Zähler, Translator etc.) sehr große Veränderungen im Code vornehmen mussten um sie überhaupt einbinden zu können.

Hier könnte man den Studenten den Einstieg erleichtern indem man die Algorithmen in den ersten Übungen noch außen vor hält und den Fokus zuerst auf die Möglichkeiten, die *Animal* bietet, legt. Zudem gibt es einige Algorithmen in *Animal* die noch nicht zweisprachig sind oder generell überarbeitet werden könnten. Wir können uns gut vorstellen, dass es für Stundenten ein guter Einstieg in *Animal* wäre einen bereits vorhandenen Algorithmus nach bestimmten Vorgaben zu überarbeiten.

Das arbeiten an den Generatoren selbst hat bei uns gut funktioniert. Zu den meisten Themen liegt genügend Dokumentation vor und man kann sich angenehm von einer zur nächsten Übung arbeiten. Bei manchen Problemen mussten wir den Assistent der Veranstaltung um Hilfe bitten, aber das hat sich in Grenzen gehalten. Das erstellen eines Eclipse-Projekts hat nach einigem hin- und herprobieren auch funktioniert. An dieser Stelle hätten wir uns eine Möglichkeit gewünscht den Generator, an dem wir gerade arbeiten, direkt zu starten, anstatt sich jedes mal durch die Animal-GUI zu klicken.

Ein weiteres Problem war die „Best Practice“. Leider können wir den Source-Code der Generatoren nicht einsehen, daher fällt es bei einigen Problemen schwer einzuschätzen welcher Lösungsweg am tauglichsten ist - gerade wenn einem die Erfahrung mit *Animal* noch gänzlich fehlt. Hier könnte man den Studenten helfen in dem man ein paar Beispiele von Generatoren sammelt die bestimmte Probleme sehr elegant lösen.

3 ANIMAL

Der Anwendung *Animal* sieht man leider an, dass schon einige Zeit an ihr entwickelt wird. Man erreicht die verschiedenen Funktionen zwar schnell und intuitiv, die Optik ist aber leider etwas eingestaubt. *Animal* verwendet ziemlich viele Fenster um die verschiedenen Tools und Informationen zur Verfügung zu stellen. Gerade an Notebooks fällt die Verwaltung der vielen Fenster nicht leicht. Hier würden wir uns einen Einzelfenster-Modus wünschen.

Die Syntax von *AnimalScript* ist gut lesbar und nachvollziehbar. Allerdings waren wir sehr froh darüber nicht zu lange mit *AnimalScript* direkt zu arbeiten, weil die *AnimalAPI* selbst für kleine Projekte viel angenehmer ist.

Die *AnimalAPI* ist gut strukturiert und bietet sehr viele Möglichkeiten seine Algorithmen darzustellen. Die java-doc zu den Primitiva könnte etwas umfangreicher sein, ist aber auch so sehr hilfreich. Die Properties-Klassen sind sehr hilfreich um das Aussehen von Objekte zu vereinheitlichen. Leider ist das Arbeiten mit den *AnimationPropertiesKeys* recht undurchsichtig. Hier würden wir uns *getter* und *setter* für die verfügbaren Attribute wünschen. Ähnlich undurchsichtig sind die *QuestionModel*. Zum beispiel das *TrueFalseQuestionModel* funktioniert nur richtig wenn man die Methode *setPossiblePoints(int)* mit 1 aufruft, während die Methoden zum Hinzufügen verschiedener Antworten zwar implementiert sind, aber offensichtlich keine Funktion bei diesem QuestionModel haben.

Das einbinden eigener Generatoren in *Animal* funktioniert dank detaillierter Beschreibung sehr gut. Leider ist nicht so gut dokumentiert wie ein Generator am besten umgesetzt wird. Beispielsweise fehlen Informationen darüber welche Aufgaben man im Konstruktor machen muss, was in der *generate()* Methode passieren soll und wofür es eine zusätzliche *init()* Methode gibt.

4 VORGEHENSWEISE

Zur Einarbeitung in *Animal* haben wir uns einige male in der Universität getroffen um die ersten Übungsblätter gemeinsam zu bearbeiten. Dabei haben wir erst darüber diskutiert, welche Aspekte des Algorithmus in der Visualisierung besonders wichtig sind und wie wir diese hervorheben wollen. Danach haben wir eine grobe Skizze gezeichnet, wie wir den Platz für die Visualisierung nutzen möchten und welche Primitiva wir dabei verwenden können.

Die Arbeiten an den Generatoren haben wir meist von Zuhause gemacht und die Ergebnisse über ein Repository miteinander geteilt. Dabei haben wir unsere Animationen gegenseitig kontrolliert um einschätzen zu können ob diese nachvollziehbar und selbsterklärend sind.

Bei den ersten Abgaben hatten wir leider - wie oben erwähnt - noch nicht auf die Bewertung gewartet und einfach weiter gearbeitet. Bei den späteren Abgaben haben wir auf die Bewertungen gewartet und dann die verschiedenen Arbeiten zum Ausbessern der Kritikpunkte aufeinander aufgeteilt.

5 Betreuung und Bewertung

Die Betreuung fanden wir sehr gut. Die Einführungsveranstaltung hat den Rahmen des Praktikums gut gezeichnet und uns im Grunde schon alles an die Hand gegeben was für das Praktikum notwendig war. Die wenigen Rückfragen die wir hatten wurden schnell und sehr hilfreich beantwortet. Es ist zudem sehr praktisch, dass man die jeweiligen Übungen mehrmals überarbeiten und erneut Abgeben kann. Die Bewertung der einzelnen Abgaben war immer sehr fair und nachvollziehbar.

Insgesamt hat uns das Praktikum gut gefallen, die Aufgaben waren anspruchsvoll und interessant gestaltet aber durchaus lösbar, die Dokumentation war ausreichend, könnte aber noch verbessert werden und die Betreuung war auch sehr gut.