Abschlussbericht AlgoVis

Valentin Kuhn

Einleitung

AlgoVis beschäftigt sich mit der Animation und Visualisierung von Algorithmen und Datenstrukturen. Dazu wird das Animal-Framework verwendet, welches AnimalScript verarbeitet und ein java-Framework zur Generierung dieser Scripte bereitstellt. Im Rahmen des Praktikums habe ich mich mit der Darstellung von FindG-Set und FindS-Set beschäftigt, zwei Machine Learning Algorithmen, die es vorher in Animal noch nicht gab. Diese fügen sich gut in das bereits vorhandene Repertoire der Machine Learning Algorithmen in Animal ein, da sie jeweilige Erweiterungen der vorhandenen Find-G bzw. Find-S Algorithmen darstellen und elementare Bestandteile des ebenfalls schon implementierten Candidate Elimination Algorithmus sind.

In diesem Bericht befasse ich mich zuerst mit dem Lerneffekt und der Organisation des Praktikums. Anschließend nenne ich einige Stärken und Schwächen von Animal, die mir während der Bearbeitung aufgefallen sind.

Ablauf

Im Praktikum habe ich gelernt, mit dem zuvor unbekannten Animal-Framework zu arbeiten und insbesondere die Java-API zu nutzen. Diese Java-API bietet, trotz einiger Fehler, eine mehr oder weniger komfortable Möglichkeit Generatoren für beliebige Eingabedaten zu implementieren.

Um meine Algorithmen visualisieren zu können, musste ich sie mir zuerst selbst wieder beibringen und ähnliche Algorithmen (insbesondere Find-G und Find-S) in Animal anschauen, um Ideen für meine eigene Visualisierung zu sammeln. So habe ich, praktisch nebenbei, wieder mehrere Algorithmen gelernt, obwohl das vielleicht nicht unbedingt das Kernziel des Praktikums war.

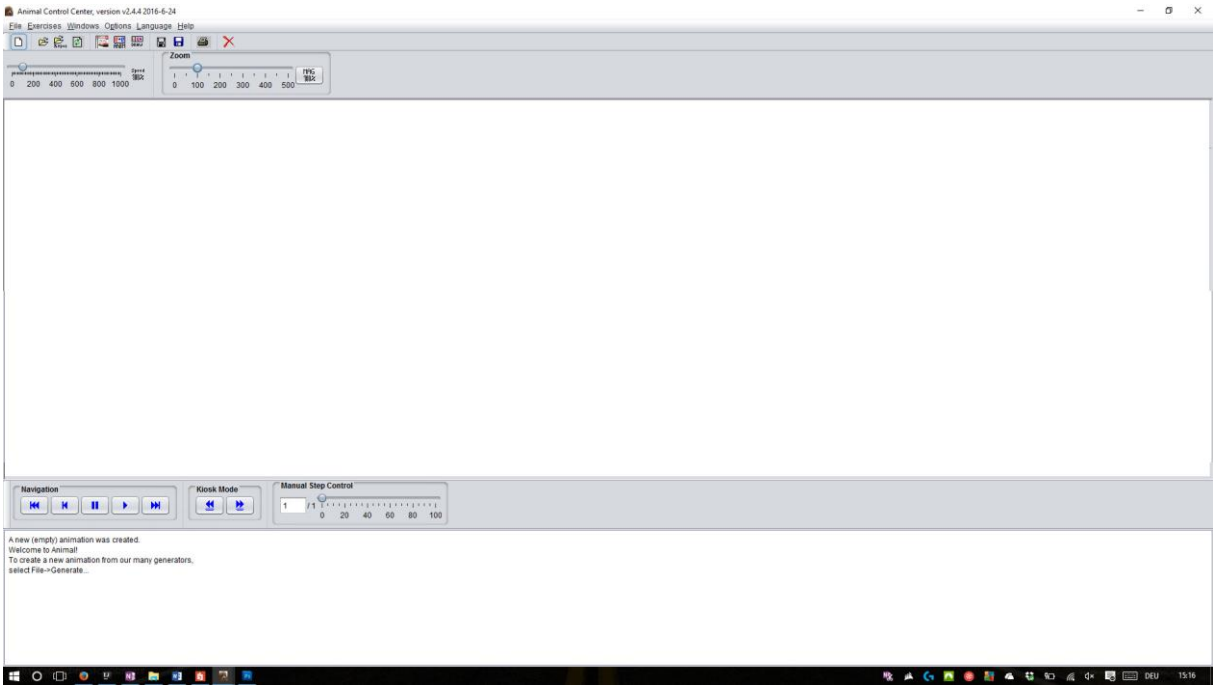
Die Arbeitsblätter haben mir einen relativ einfachen Einstieg in Animal geboten. Für kommende Jahre würde ich mir jedoch wünschen, dass diese Arbeitsblätter besonders für Testat-Gruppen offensiver beworben werden, da ich am Anfang dachte, diese seien nur für Online-Gruppen relevant. Vor der ersten Besprechung habe ich die ersten zwei Arbeitsblätter dann trotzdem bearbeitet, und so auch einen Einstieg gefunden. Anschließend habe ich anhand des dritten Arbeitsblatts einen Generator für FindG-Set erstellt, wobei mir das „APIExample“ Template sehr geholfen hat. Die Integration meines Generators in Animal wurde durch das Generieren des Generators zwar einfacher, stellte sich dann aber doch als anstrengender heraus als erwartet, was vor allem daran lag, dass ich die zusätzlich benötigten Libraries nicht in mein Projekt importiert hatte. Um diese Hürde in Zukunft zu senken, könnte zum 4. Arbeitsblatt ein Hinweis auf diese zusätzlichen Libraries hinzugefügt werden, oder Animal könnte die Libraries direkt als Maven Dependencies beinhalten (diese werden beispielsweise von IntelliJ IDEA direkt als Modul-Dependencies hinzugefügt).

Da ich alleine gearbeitet habe, habe ich leider keine weiteren Teamwork-Erfahrungen gesammelt. Dies wird durch die Zusammenarbeit in anderen Fächern und bei der Arbeit jedoch ausreichend kompensiert. Sehr entgegenkommend war die Unterteilung in Testat- und Online-Gruppen und die resultierende Flexibilität, sodass jeder das Praktikum bearbeiten konnte, wann und wie er wollte. Die gestaffelten Deadlines gegen Ende des Semesters (1. Testat, 1. Generator, 2. Generator & Abschlussbericht) erzeugten trotz dieser Freiheit genügend Zeitdruck, um auch irgendwann fertig zu werden.

Animal

Das Animal-Framework ermöglicht mit relativ wenig Aufwand das Erstellen von Visualisierungen. Leider gibt es dabei jedoch immer wieder Fehler und viele kleinere Dinge, die verbessert werden könnten.

Die Aufteilung von Animal in mehrere Fenster macht es beispielsweise sehr unübersichtlich, ein Ein-Fenster-Modus, der direkten Zugriff auf die Kontrollcenter-Optionen und die Animation bietet, würde hier weniger verwirren. Weiterhin scheint der Speed-Regler keinerlei Auswirkungen auf die Animation zu haben. Ein Ein-Fenster-Modus könnte beispielsweise fongendermaßen aussehen:



Im Framework würde ich mir vor allem Erweiterungen für die Darstellungen von Texten wünschen. Ein einfacher Text kann leider nur einzeilig sein, als Abhilfe kann auf SourceCode umgestiegen werden, welcher mit entsprechenden SourceCodeProperties auch als „normaler“ Text formatiert werden kann. Dem SourceCode fehlt jedoch die Möglichkeit, den Text zu ersetzen oder zu löschen, es können nur Zeilen hinzugefügt werden. So muss für jede Textänderung der alte SourceCode versteckt werden und ein neuer an die gleiche Position gesetzt werden. Außerdem gibt es keinen automatischen Zeilenumbruch, um Textfelder (oder SourceCode) in einer bestimmten Größe zu erstellen.

Eine weitere Kleinigkeit, die mir aufgefallen ist, ist ein Konflikt der Background- und Highlight-Funktion. Wenn man diese auf die Zellen einer StringMatrix anwendet, wird nur der jeweils letzte Wert verwendet. Setzt man Beispielsweise den Hintergrund einer Zelle, die zuvor highlighted wurde, und entfernt diesen Hintergrund anschließend, wird die Zelle weiß angezeigt, anstatt wieder zur Hightlight-Farbe zurückzukehren.