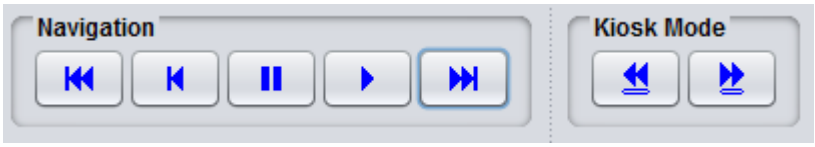
Abschlussbericht von Yadullah Duman

Ich hatte sehr viel Spaß am Praktikum und würde es gerne wieder machen, wenn es möglich wäre. Vom Ablauf her, war ich positiv beeindruckt! Alle relevanten Informationen waren in Moodle zu finden und waren ausführlich erklärt, was ich sehr begrüße, da ich neben dem Studium auch sehr viel arbeiten muss. Wenn man fragen hatte, wurden sie schnell beantwortet. Sei es im Forum, oder durch persönliche Mitteilungen des Dozenten. Die aktive Hilfsbereitschaft des Dozenten, war sehr bemerkbar und enorm hilfreich, was man in den zukünftigen Veranstaltungen so beibehalten kann. Des weiteren war die, meiner Meinung nach, faire Zeitverteilung der Aufgaben sehr begrüßenswert. Da ich in diesem Semester neben dem Praktikum noch fünf weitere Module belege, konnte ich zeitlich sehr flexibel planen. Im Großen und Ganzen bin ich mit dem Ablauf des Praktikums sehr zufrieden gewesen. Was mich nur minimal gestört hat war die AlgoAnim API, auf die ich aber später eingehen werde. Im Praktikum habe ich gelernt, wie man sich eigenständig in ein Java-Projekt einarbeitet. In meinen bisherigen Projekten/Praktika, war immer mindestens ein weiterer Kommilitone mit dabei. Diesmal war das nicht der Fall und somit lag die ganze Verantwortung und Organisation bei mir, was mir im Prinzip sehr gefallen hat. Darüber hinaus habe ich gelernt, dass man sorgfältig planen sollte bevor man anfängt zu arbeiten, worauf ich später auch nochmal näher eingehen werde. Vor allem aber, habe ich durch das Praktikum zwei weitere Algorithmen kennengelernt, auf die ich vielleicht in Zukunft nie gestoßen wäre. Es war schön sie zu studieren und im Internet darüber zu recherchieren und sich verschiedene Implementierungen des Problems anzuschauen. Man hatte auf jeden Fall sehr viel Input und konnte dies in Ideen umwandeln und später umsetzen. Auch gelernt habe ich (erst im Nachhinein), dass man mehr als nur „Sortier- und Suchalgorithmen“ visualisieren kann. Dass man die Möglichkeit hat auch Inhalte zu visualisieren, die nicht auf Informatik beschränkt sind, ist meines Erachtens nach eine sehr gute Herangehensweise, da man ANIMAL mit sehr vielen unterschiedlichen Visualisierungen füttern kann, die dem ein anderen Student_in beim studieren helfen kann.

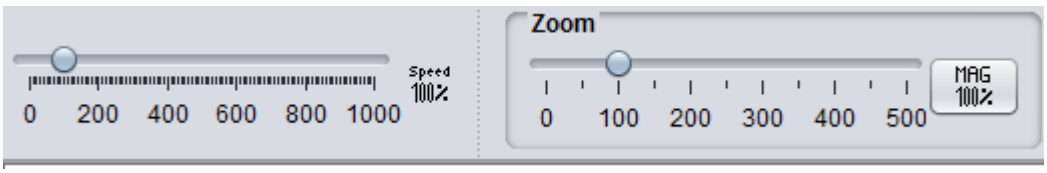
Das Programm ANIMAL kannte ich schon ein wenig im letzten Sommersemester. Damals habe ich Grundlagen der Informatik 2 (GdI2) belegt, wo Algorithmen und Datenstrukturen eine wesentliche Rolle spielen. Um ehrlich zu sein, fiel mir der Anfang sehr schwer und mir war nicht klar was zum Beispiel „Generatoren“ sein sollten. Ich wollte einfach einen Algorithmus starten und sehen, was da geschieht. Da mir das ganze am Ende zu aufwändig war, und ich mich als ziemlich fauler Mensch beschreiben würde, habe ich es „links liegen lassen“. Meine Kritik zu ANIMAL ist deshalb folgender: Ich finde die Idee dahinter einfach genial! Dass man die Möglichkeit hat, „den trockenen Stoff“ in Aktion zu sehen, mit einer Erklärung nebenbei und vielen Beispielen ist etwas, was sich wohl jeder Student_in wünscht. Die Umsetzung dieser Idee, ist schon mal ziemlich gut (sei es die Animation selbst, Beschreibung, Source Code, Inhaltsverzeichnis, Änderung von Parametern etc.), aber meines Erachtens nach, könnte man es noch viel einfacher und interessanter gestalten, damit mein „faules Ich“ vom letzten Sommersemester vielleicht nicht so

schnell das Handtuch geworfen hätte. Nun zu den Punkten, was man an ANIMAL, AnimalScript und/oder der AlgoAnim API verbessern sollte: Fangen wir an mit ANIMAL. Wie schon bereits erwähnt, finde ich, dass ANIMAL etwas „simpler“ gestaltet werden sollte. Der Benutzer sollte nicht groß darüber nachdenken, was bspw. „Generate...“ heißen könnte. Im Endeffekt will man ganz am Anfang erst mal animierte Algorithmen sehen. Das heißt im Endeffekt, möchte ich als Benutzer, eine simple UI, mit der ich sofort beim ersten Start klarkomme. Zurzeit ist es eher anders. Ich starte ANIMAL, und direkt taucht ein Fenster „Empty Animal Animation“ auf. Ich klicke es natürlich weg, weil ich was animiertes sehen will und keine „leere Animation“ und deshalb stelle ich mir schon die Frage, weshalb dieses Fenster bei jedem Start immer auftauchen muss? Ich wollte animierte Algorithmen sehen, deshalb suche ich weiter. Ich nehme an, dass der „ANIM-Button“ es sein könnte, aber der startet auch nur eine „Empty Animation“. Ich suche ich weiter. Nächste Vermutung: es könnte in „Exercises“ liegen. Ich fahre mit der Maus erst mal darüber um Informationen zu bekommen, stattdessen bekomme ich die Nachricht „Invalid Message Key exercises.toolTipText“, was gefixt werden sollte. Ich ignoriere mal diese Fehlermeldung und gehe zu „Open an Exercise“ und diese erwartet Daten von mir, aber ich will immer noch etwas animiertes sehen, deshalb suche ich weiter. Ich gehe zu „File“ und finde „Examples“! Doch nach einigen Klicks, finde ich immer noch keine Animation. Schließlich gehe ich einfach mal zu „Generate...“ und bin endlich glücklich. Genau dieses Szenario hatte ich im letzten Sommersemester (das war Versuch 2, nach dem gescheiterten Versuch 1). Ich musste sehr viel hin und her klicken, bis ich das wollte, wonach ich gesucht habe. Ein Vorschlag wäre den Block „Generate...“ bis „Demo animation“ in einen separaten Tab auszulagern, weil es für mich nichts in „File“ zu suchen hat, da ich immer davon ausgehe, dass in File Aktionen, wie speichern, drucken, erstellen, löschen usw. existieren. „Generate...“ sollte umbenannt werden zu etwas wie „Animate…“, damit einem wirklich klar ist, dass man hier die Animationen findet.
Was man auch noch verbessern kann, sind die zwei folgenden Leisten:

Leiste 1



Leiste 2



Was mich beim ersten Start bei Leiste 1 irritiert hat, war, dass ich dachte, dass der Button rechts von „Pause“, die Animation direkt startet, da es wie ein „Play“-Button aussieht. Wenn man diese Darstellung jedoch eher für „gehe eine Folie weiter“ benutzt, dann frage ich mich warum der Button für „gehe eine Folie zurück“ (links vom Pause-Button) einen „Strich“ hat. Ich finde, dass diese Buttons ähnlich aussehen sollten. Darüber hinaus sollte man den „Kiosk Mode“ umbenennen zu „Animation Mode“, weil ich am Anfang gar nicht wusste, was „Kiosk“ in diesem Kontext bedeutet. Das Einzige, was ich an Leiste 2 verbessern würde, wäre die Skalierungen für „Speed“ und „Zoom“ zu verkleinern, da diese zu groß sind. Ich finde 1000% Speed und 500-faches Zoom, sind schon extreme Werte.

Das war so ziemlich alles, was ich in ANIMAL verbessern würde. Es geht dabei schlicht mehr um die „Einfachheit“ des Programms. Zu AnimalScript kann ich leider wenig sagen, da ich mich Praktikum kaum damit befasst habe, weil ich ausschließlich mit der AlgoAnim-API gearbeitet habe, wozu ich etwas mehr sagen kann.

Ein großer Pluspunkt ist, dass die API sehr einfach ist und man sie somit schnell versteht. Wenn man seinen Variablen und Methoden bei der Implementierung sinnvolle Namen gibt, dann lässt sich der Code sogar wie ein Stück Text lesen. Nichts desto trotz finde ich, dass „Clean Coding“ mit der API sehr schwer ist und das finde ich ziemlich schade. Es gibt teilweise Methoden die mehr als drei Parameter erhalten und manche Parameter sind (fast) immer „null“ (Beispiel: Der Parameter „Timing offset“). Das erschwert einem die „Code Comprehensibility“[1]. Außerdem wäre es schön, wenn es eine andere Möglichkeit gäbe, als immer wieder „Language.nextStep()“ zu wiederholen. Das gleiche gilt für „SourceCode.highlight()“ und „SourceCode.unhighlight()“ und „SourceCode.addCodeLine()“. Mir fällt zurzeit dafür noch kein Verbesserungsvorschlag ein, aber beim Programmieren habe ich gemerkt, dass diese Methoden den Code aufblähen und sich ständig wiederholen. Des weiteren konnte ich im Forum ab und zu noch beobachten, dass einige Kommilitonen Probleme mit Bugs hatten, wie zum Beispiel, dass Grids nicht gezeichnet/markiert werden o.ä. Da ich zum Glück von solchen Bugs nicht betroffen war, wäre es trotzdem nett, dass man diese zu einem gegebenen Zeitpunkt fixen kann, damit zukünftige Studierende nicht denken, es wäre ein Fehler ihrerseits, obwohl der Code korrekt ist, aber die API intern den Code falsch interpretiert.

Die Übungsblätter für das Praktikum waren verständlich, auch alleine lösbar, und sehr hilfreich während dem Praktikum (vor allem Übungsblatt 4 und 5). In diesen wurden wichtige Themen, wie der Wizard-Generator, die Erstellung eines Generators und das anschließende Einbinden in das Framework ausführlich erklärt. Das kann man für zukünftige Veranstaltungen gerne beibehalten. Schließlich noch ein paar kurze Worte darüber, wie ich im Praktikum vorangegangen bin.

Da ich, wie schon erwähnt, alleine arbeiten sollte, wollte ich was neues ausprobieren, wozu ich noch nie gekommen bin. Ich habe mir ein Kanban-Board[2] angelegt und nur damit alles organisiert und geplant. Mein Board war unterteilt in vier Spalten: Backlog, ToDo, Doing, Done. Im Backlog

1 Siehe auch „Clean Code: A Handbook of Agile Software Craftsmanship“ von Robert C. Martin

2 Siehe https://de.wikipedia.org/wiki/Kanban_(Softwareentwicklung)

habe ich alles mögliche gesammelt, was ich im Hinterkopf hatte. Alles, wo ich wusste, „das muss gemacht werden“ kam in den Backlog. In „ToDo“ kamen alle „Issues“ rein, die ich für die aktuelle Woche machen wollte. Hier habe ich mir eine realistische Menge an Issues genommen, bei der ich zuversichtlich war, dass ich sie in diesem Zeitraum erledigen könnte. In „Doing“ dann die Issues, die ich aktuell mache und wenn diese fertig wurden, kamen sie in „Done“. Ich war von dieser Methodik positiv überrascht, da ich auf einmal extrem produktiv wurde und alles im Überblick hatte. Darüber bin ich echt froh, dass ich das in diesem Praktikum „entdecken“ durfte, weil ich ungefähr das gleiche Prinzip nun auch für mein Studium generell benutze, und eine Leistungssteigerung merke.