University of Glasgow | School of Computing Science

Algorithm Animator

Andrei-Mihai Nicolae

School of Computing Science
Sir Alwyn Williams Building
University of Glasgow
G12 8QQ

Level 4 Project — March 22, 2017

Abstract

Understanding algorithms is both very common and hard for developers in general, regardless of their level of expertise. Even the fundamental ones, such as Dijkstra's algorithm for finding the shortest path between two nodes in a graph, are quite complicated to grasp. Many studies show that visualizing an algorithm and its steps make understanding it much easier. In this report, we will present an Algorithm Animator built specifically for solving this problem in a modern, responsive and efficient manner. Among others, we will also show why certain design decisions (e.g. making it a native desktop app instead of a basic jar, using material design for the user interface), the implementation choices and the evaluation results make this tool a viable option for software engineers when it comes to learning different kinds of algorithms.

Education Use Consent

I hereby give my permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic format. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Name:		Signature:	

Contents

Chapter 1

Introduction

1.1 Aims

1.2 Motivation

1.3 Contributions

1.4 Report Content

The rest of the report will analyze the background of animators and why they were proven useful, as well as cover all the steps in gathering requirements, designing, implementing, testing and evaluating the tool.

- Chapter 2 covers work related to the purpose of algorithm animators and why they are useful
- Chapter 3 goes into how the problem was analyzed and what requirements were gathered through project meetings and discussions with Algorithmics students.
- Chapter 5 explains the design decisions behind the tool and illustrates various lessons learned and problems faced along the way.
- Chapter 6 goes into the implementation details of the animator.
- Chapter 7 show how extensive unit, integration and other types of testing (e.g. smoke, end-to-end) were undergone and why they were essential to the development of the application.
- Chapter 8 details the overall results of the project.

Chapter 2

Background

2.1 Related Work

Chapter 3

Requirements

3.1 Problem Analysis

3.2 Requirements Gathering

3.3 Functional Requirements

3.4 Non-Functional Requirements

Chapter 4

Planning

4.1 Agile

4.1.1 Kanban Board

4.1.2 Issues & Bug Tracking

Chapter 5

Design

5.1 Architecture

5.1.1 EDA (Event-driven Architecture)

5.2 Native Desktop App vs. Jar

5.3 Electron

5.4 Vis.js

5.5 Material Design

5.6 Compromises

Chapter 6

Implementation

6.1 Project Structure

6.2 JavaScript and Multi-Threading

6.3 JS Animation Engine

6.4 Extra Features

6.5 Lessons Learned

6.6 Issues Faced

Chapter 7

Testing

7.1 Unit Testing

7.2 Integration Testing

7.3 Prototype Evaluation

7.4 Results

Chapter 8

Conclusions

8.1 Open Source

8.2 Project Roadmap

8.3 Final Thoughts

Bibliography