University of Sheffield

User-Defined Data Structures and Algorithms Visualization Teaching Tool for Beginners

Hongyu Wang

Supervisor: Dr Mike Stannett

A report submitted in fulfilment of the requirements
for the degree of BSc in Artificial Intelligence and Computer Science

in the

Department of Computer Science

May 10, 2023

Declaration

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination.

Name: Hongyu Wang

Signature: Hongyu Wang

Date: October29, 2022

Abstract

It is well known that data structures plus algorithms equal programs[1]. The standard way of using computers to solve problems is to abstract an appropriate data model from specific issues, design an algorithm with the model, and then write a program in one developing language to solve the problem. This means that data structures and algorithms are crucial for computer science learners. Most university students in the UK come from various backgrounds. Although many have entry-level programming skills, some students may not have a solid theoretical computer science foundation in high school. As a result, they might find it difficult when faced with abstract algorithms and data structures in some modules. Therefore, a convenient and efficient learning tool or teaching tool is necessary. This project aims to create an interactive program visualization system for students and teachers to facilitate beginners learning algorithms and data structures.

The following chapters will report the work done so far, including an introduction and discussion of the survey results and literature research, mathematical formulas, and flow diagrams with particular language-developed demos, which will be used to demonstrate the solutions.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr Mike Stannett, for his invaluable guidance and advice on functional programming and his constructive feedback on other aspects of my dissertation. I am also grateful to Professor Heidi Christensen for her participation in my presentation and for serving as my second marker. Their expertise and support have significantly contributed to the completion of my dissertation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

First-year undergraduate students at the University of Sheffield's Department of Computer Science study basic data structures and algorithms and must pass an advanced module in their second year. Although many students have basic programming skills, the advanced module can be challenging for beginners. Traditional lectures often use pseudo-code or programming languages to explain data structures and algorithms, which may not be intuitive for students. Some educators now use graphical views to represent data structures, making it easier for students to understand. However, these approaches have limitations: students tend to be passive, the views are often static or inconsistent, and they may find it difficult to apply their knowledge in practice. Developing educational tools that combine graphical views and real programming environments through software engineering could help address these issues and enhance students' learning experiences.

1.2 Aims and Objectives

This project aims to create a visualization tool for algorithms and data structures at TUOS's Computer Science Department. It is designed to bridge the gap between programming and theory for beginners, enhancing teaching efficiency. The tool will supplement traditional lectures, not replace them, and improve students' understanding through practical programming interactions and graphical representations.

The project seeks to support beginner learning without replacing existing solutions. Understanding and evaluating current solutions and their impact on beginners is crucial, as it will influence the software design. Technical challenges include visualizing various information types, such as pseudo-code, data structures, and their logical relationships. The tool should also display dynamic behaviour, like animations, to facilitate algorithm understanding. The primary challenges will be designing and developing dynamic visualization functions and managing the status and relationships among different information types.

1.3 Overview of the Report

The report contains 7 chapters, covering the literature survey, aims and requirements, design, implementation, testing, outcomes, and conclusion.

Chapter 2

Literature Survey

This chapter will give an overview and discuss the work done in visualization or animation systems.

2.1 What is the Data Structures and Algorithms Visualization?

The IT industry's popularity has driven programming education development, increasing interest in Program Visualization (PV) tools over Data Structures and Algorithms Visualization (AV) tools. See Figure2.1. The use of AV tools in education dates back to the Swan visualization system of Virginia Tech in 1996[6], which will be discussed later in this chapter. Both visualization systems use graphics and animations to illustrate processes, often in educational settings, to help programming beginners. The differences lie in the visualization content and methods. Program Visualization (PV) focuses on graphically representing a program's execution process[11], which means larger programs can be divided into several "black boxes", with visualization focusing on these black boxes' inputs and outputs. In contrast, the process can be appropriately "faked" See Figure2.1. Data Structures and Algorithms Visualization (AV) is more specific, mainly concentrating on graphically representing data structures (e.g., linked lists, trees, graphs, etc.) and algorithms (e.g., sorting, searching, etc.) in programs[12]. Data structures can be considered logical structures with storage functionality, typically represented as physical structures in computer programs, such as constructing a binary tree using an array[13]. Thus, Data Structures and Algorithms Visualization (AV) tools can be viewed as PV tools specifically targeting variable storage structures. The functional definition of data structure and algorithm visualization tools in this report is derived from Nathasya[5]: Algorithms and Data Structures Visualization (AV) tools are a series of software that illustrates how algorithms operate on data structures through animations and intuitive graphical views, often employed in teaching.

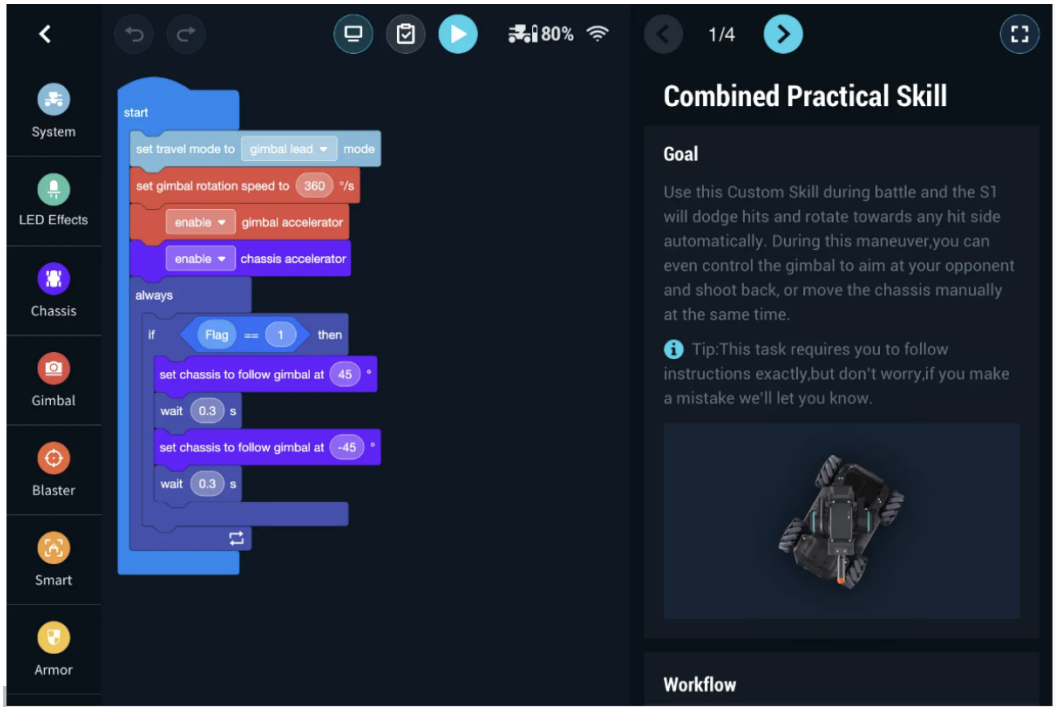


Figure 2.1: The Program Visualization of DJI Robo Master education robot, The implementation details of code blocks are hidden inside a "black box" and expressed in graphical form[2].

2.2 Existing Solutions Survey

Common AV systems typically share certain features: they often exist as websites offering encapsulated graphic libraries and corresponding animation libraries, such as graph structures formed by shapes and connecting lines. Parts of each shape can be manipulated to create a full animation. System developers usually handle all visualization work, for example, selecting multiple shapes from the library and defining their movement behaviour, writing relevant algorithm source code, and referencing animations by using program annotations on the statements, thereby completing a comprehensive data structure and algorithm visualization scheme. Users access the website and choose to watch pre-configured schemes. See Figure2.2. Some websites also integrate graphics and algorithm source code. See Figure2.3. While these systems can promote learning for beginners, they also have some drawbacks. Firstly, developers often take on the system's animation and programming roles, and users merely access the website as watchers, resulting in a lack of interaction with the system and limited hands-on practice. Additionally, the programming languages used for writing algorithm source code on websites may not be familiar to beginners, implying an increased learning cost. In this case, using pseudo-code as an auxiliary tool may reduce the learning difficulty.

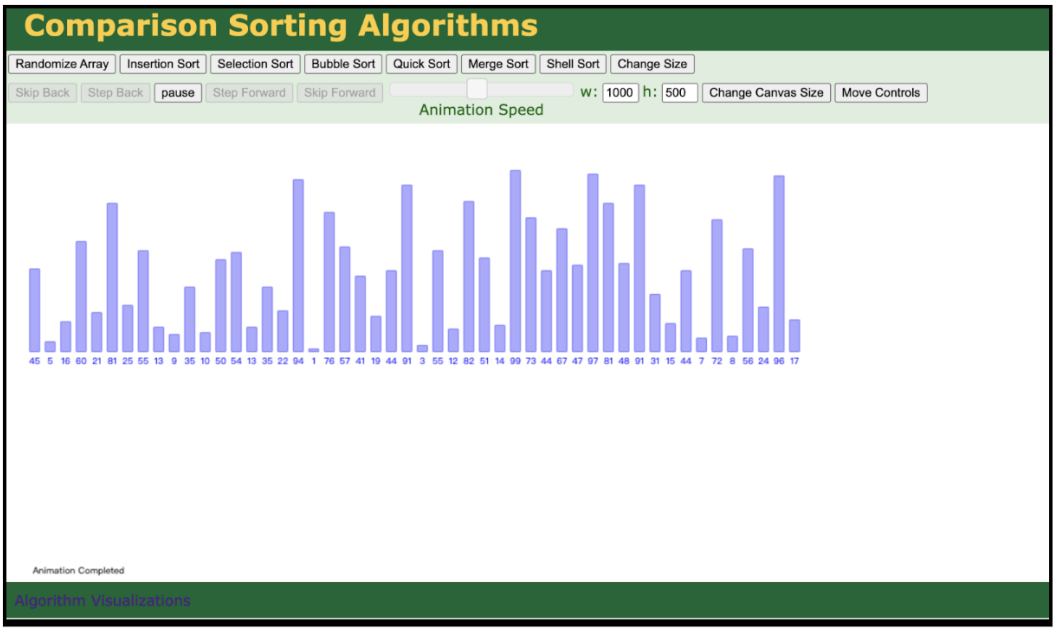


Figure 2.2: The figure shows the University of San Francisco's Data Structure Visualization[3].

Compared to publicly available AV systems, the four systems used on university campuses discussed in this section are more noteworthy. They share the common characteristic of being useful in supporting beginners in unsupervised learning or as supplementary tools for traditional teaching. Notably, two of the discussed AV systems have well-developed basic features and can also derive visualizations of the inner workings of complex data structures.

2.2.1 DS-PITON Algorithm Visualisation Tool

The content of subsection 2.2.1 is based on the work of Nathasya et al. [5].

The DS-PITON algorithm visualization tool is based on the Python program visualization (PV) tool PITON. It was developed to bridge the gap between theory and practice that beginners encounter when facing data structures and algorithms. It had a visualization algorithm based on PITON PV but optimized for the algorithm; Users can directly use the seven common data structures embedded in DS-PTION by creating objects and observing their visualization process on the UI. Figure2.4 and Figure2.5 shows how users use it through a simple UML diagram and a standard UI of the system.

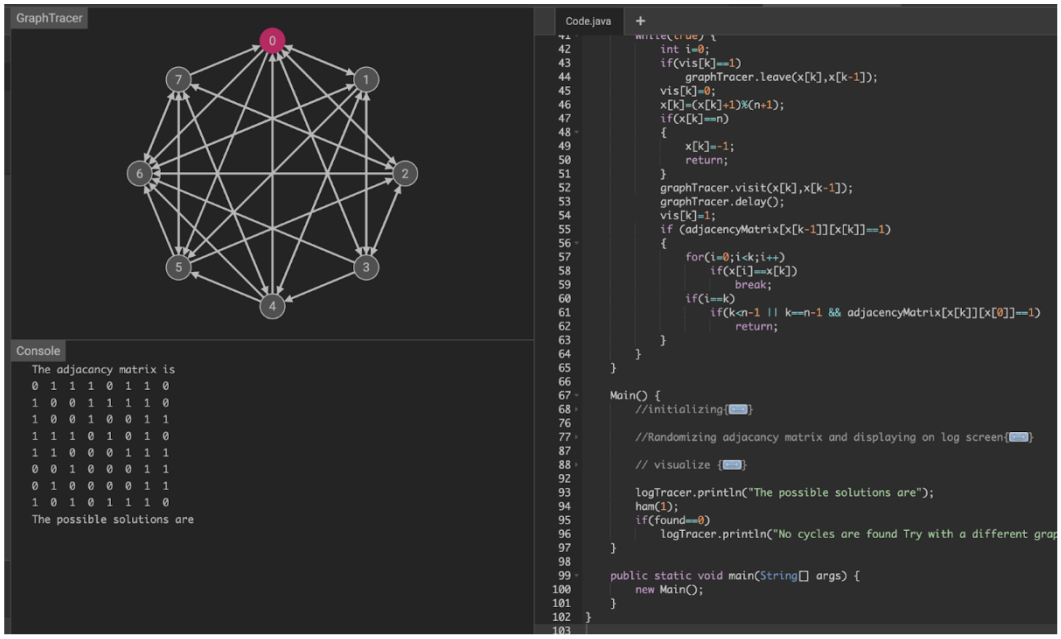


Figure 2.3: The figure shows one of the most popular Data Structure Visualization systems: Algorithm-visualizer[4].

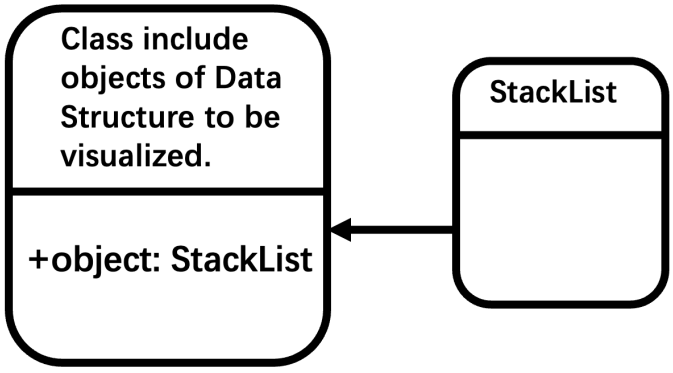


Figure 2.4: A Figure shows a UML diagram of how users use the DS-PITON developed data structure to the program based on the work of Nathasya et al. [5].

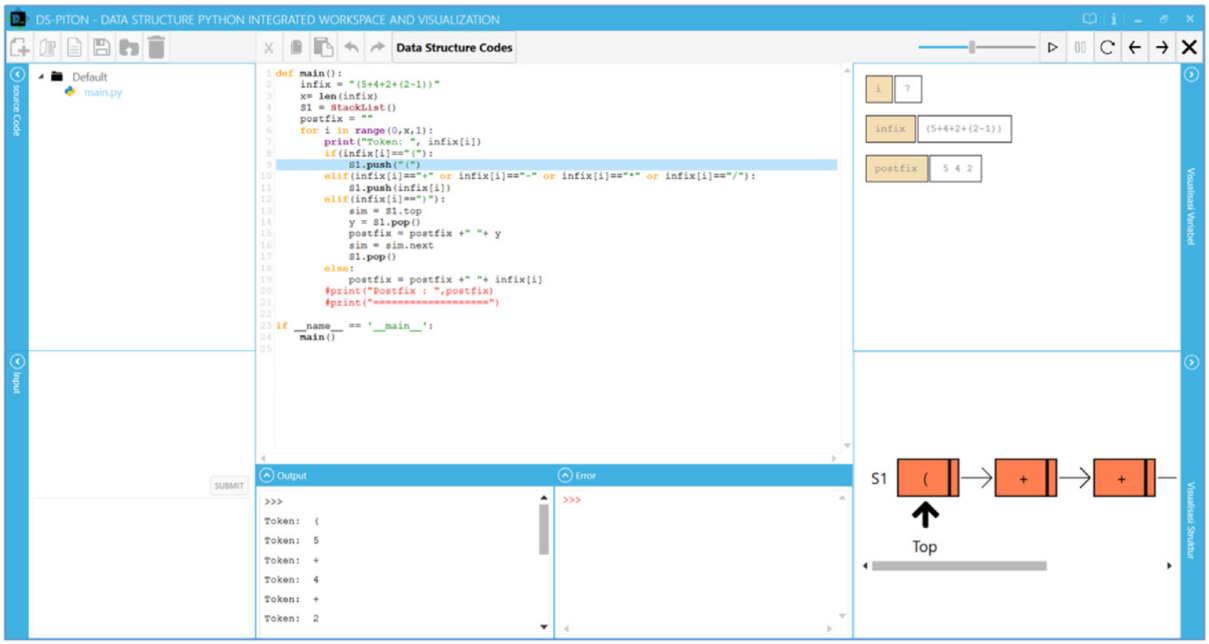


Figure 2.5: A Figure shows a standard GUI of DS-PITON[5].

Like common AV systems, this system can display data structures and algorithm operations using the beginner-friendly Python language as the system's interactive language. One of the advantages of this system is its easy-to-use interface. The user interface integrates the output console, the file path display area, and an area for prompting compilation errors. See Figure2.5. As a result, the interface is a simple Integrated Development Environment (IDE) with integrated visualization, which most people who have learned to program are familiar with, such as Eclipse[14]. What sets this visualization system apart from the others discussed later is its noteworthy feature of displaying data structures in static graphics and thoughtfully presenting the algorithms' source code. When the play button is clicked on the control panel in the top left corner, an animated sequence is played, highlighting the source code as the graphics change. Users can also use the fast-forward and rewind buttons to view different algorithm steps. See Figure2.5. Furthermore, this system can visualize data structures and components of complex structures. See Figure2.6

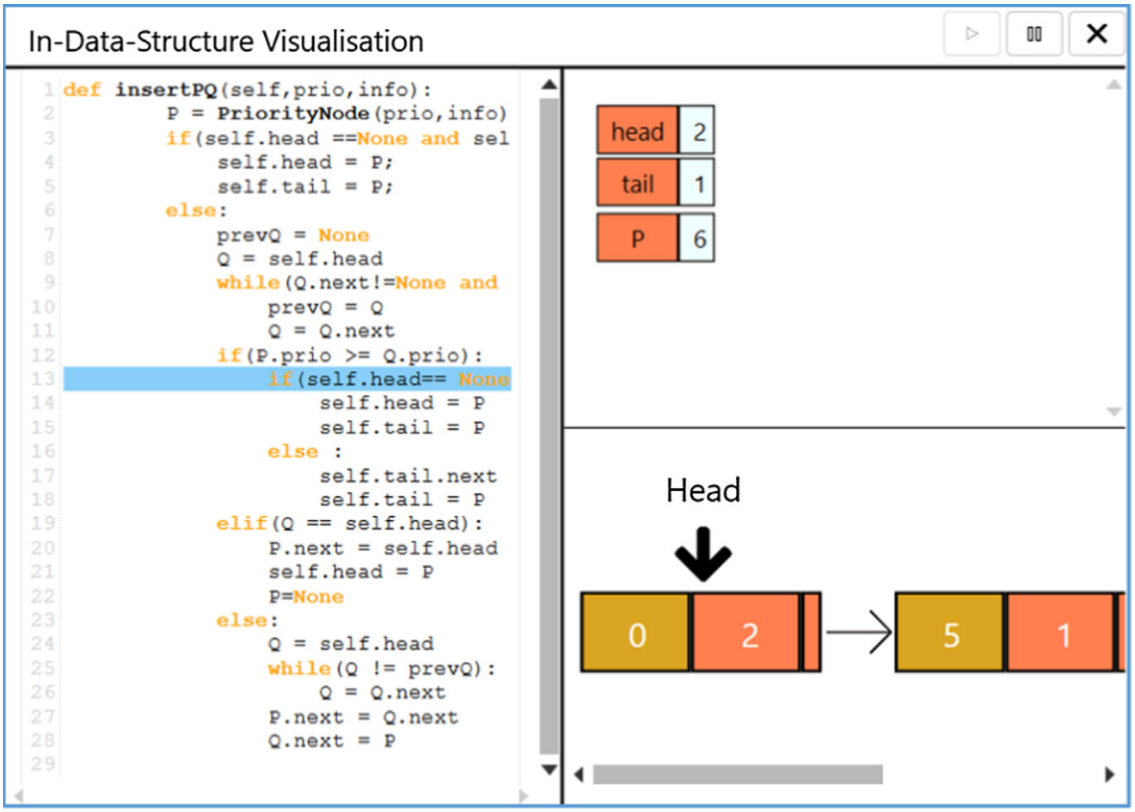


Figure 2.6: In-Data-Structure Visualization[5].

However, the shortcoming of this DS-PTION is also obvious. Firstly, it is known that only seven data structures are currently implemented by developers in this visualization system. Although it is based on a reliable PV system and shows the potential to allow secondary development, each visualized data structure has to be developed well, and the final visualization effect entirely depends on the developers' code quality. In addition, students are allowed to invoke the developed data structures and watch the implementing processes inside as an audience but are not allowed to participate in the development process. Therefore, the performance of the visualization system in teaching scenarios is excellent. It bridges the gap between theory and practice to a certain extent, but it could have stronger programming interactivity to support unsupervised learning scenarios better.

2.2.2 Swan Data Structure Visualization System

The Swan system can visualize the data structures and execution processes from C++ or C programs, and it has been used as a teaching and learning tool at Virginia Tech. The Swan system's working principle for visualisation is compiling the annotations on the statements in program files to generate views and executable processes. Hence, developers or views designers need to invoke functions from the Swan Annotation Interface Library(SAIL) to their program files, the functions in SAIL can provide functions including controlling the process of generating graphics and responding to the operations of Viewers on the UI[6].

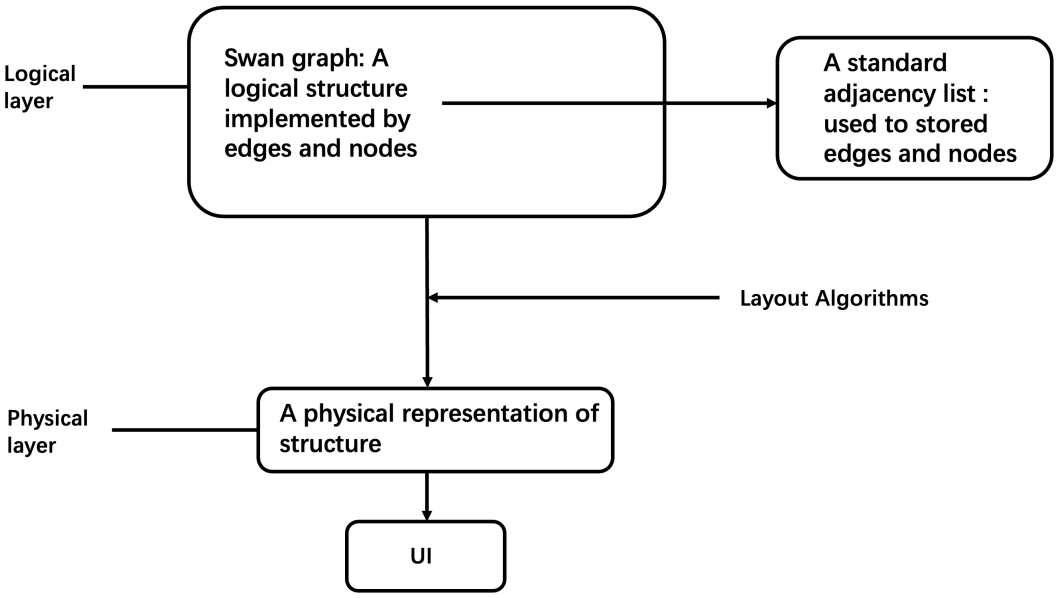


Figure 2.7: A Figure shows how to swan system visualizes data structures based on Shaffer(1996)'s work[6], the logical layer used to store logical structures created by annotation program and the layout algorithms transfer them to physical representation

It is worth mentioning that the Swan system can display the construction process of the data structure step by step on the user interface instead of directly mapping the entire structure like most systems, and it also allows users to directly make changes to the graphical view on the interface such as deleting or inserting nodes. The principle is that the developers of the Swan system did not choose only to visualize common data structures but decomposed them into an edge and a pair of nodes. Users can use annotation programs to build a logical structure such as graph structure and tree structure(Swan graph); a standard adjacency list is then used to store the edges and nodes in the logical structure. The layout algorithm will obtain the physical representation of the data structure through the standard adjacency list and store it in the physical layer. See Figure2.7. The physical layer can be used as a buffer to gradually map different parts of the data structure onto the UI.

This visualization system had three advantages: firstly, the algorithm realised the automatic layout. Second, it could visualize the logical and physical structures from programs such as an adjacency list used to build a graph. See Figure2.8. Additionally, it allowed users to interact in two ways: code programming and editing the graphical view of data structures after visualising through the user interface.

The application of automatic layout enabled code developers or designers to focus on codes and reduce their workload. Visualizing physical structures can intuitively let beginners understand how those complex structures are implemented. For example, the adjacency list in Figure2.8 can be implemented using LinkedList in Java [15]. This feature differs from the DS-PITON system's in-data-structure visualization (See Figure2.6), which is more biased towards methodology visualization. This makes the Swan system more professional because it reflects its optimization for computer science learners. Therefore, Swan System is not only easy to use but also more suitable

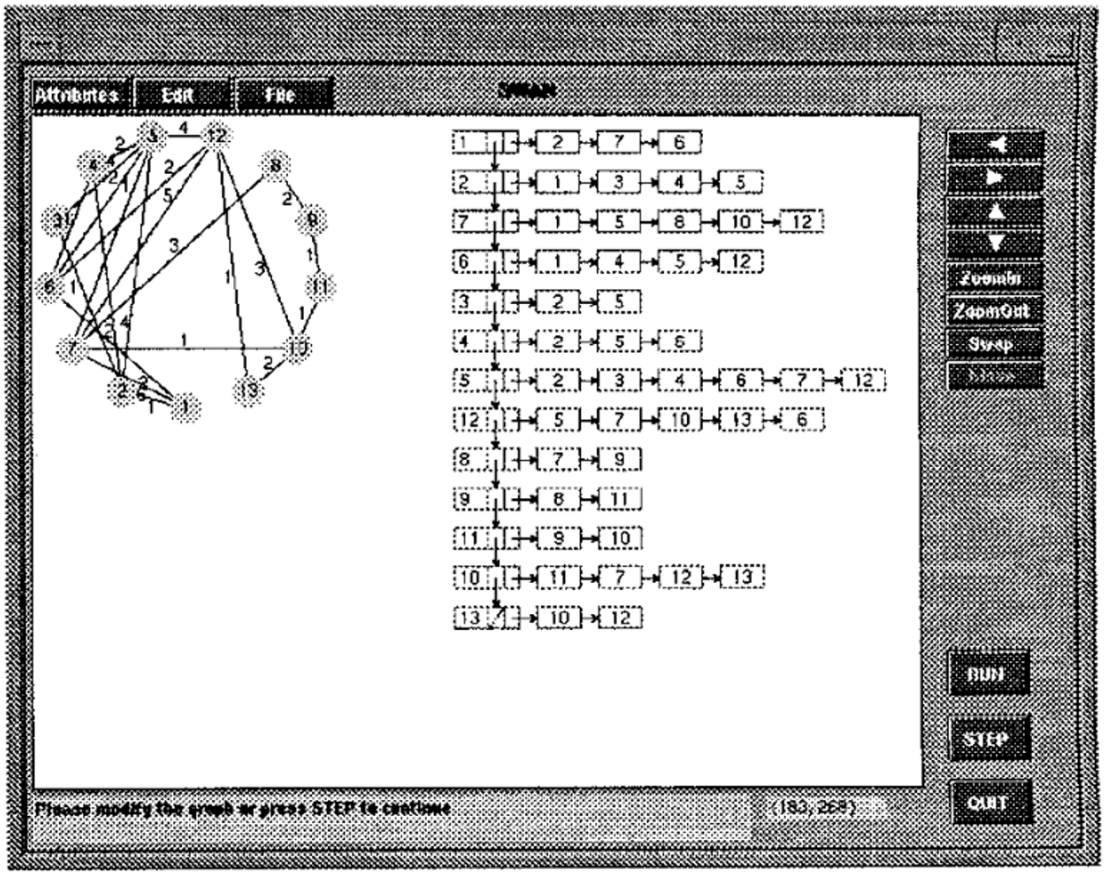


Figure 2.8: A Figure shows two views of a graph, the one on the right is how the data structure is implemented: the standard adjacency list. Users can interact with the view through the control panel on the right-hand side[6].

for beginners who are studying computer science.

2.2.3 JavaMy Software

The JavaMy system is similar to DS-TION in many ways but is developed in Java. Students who majored in computer science at TUOS have studied Java programming for a year, making Java the most familiar programming language for most students. This means that discussing the JavaMy system may inspire the implementation of this project. JavaMy achieves visualization by secondary development of common data structure classes in JDK, allowing users to choose the data structure to be automatically projected onto the user interface by creating objects. The process of realizing visualization in the JavaMy system is natural, as the secondary development of data structures inherits the native syntax of Java programs instead of relying on an annotation interface library like the Swan system discussed before.

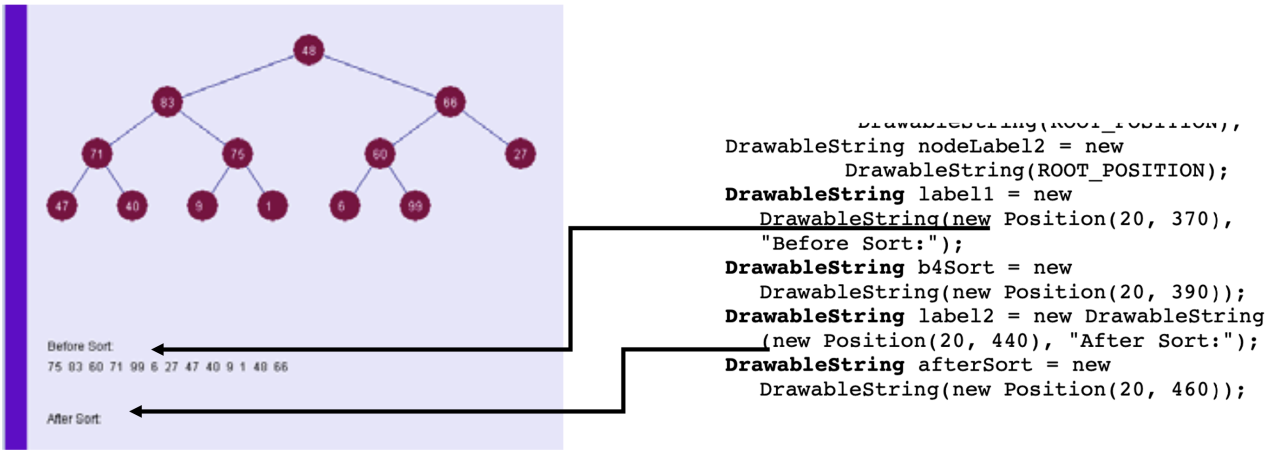


Figure 2.9: A figure shows to create the object of DrawableString need to provide the position information for the description of animation[7].

The JavaMy system offers a feature that provides users with various animation implementation algorithms. Animators can insert frames with animation descriptions by instantiating DrawableString objects, and adding more frames leads to richer animations. However, JavaMy doesn't implement an automatic layout algorithm for animation description strings; animators must manually provide position coordinates as parameters when creating a DrawableString object as a frame, as shown in Figure2.9. This means that creating a more detailed animation process requires extra teacher effort when used for teaching. Moreover, when using the system for learning, students cannot fully concentrate on implementing the algorithm, as they need to figure out the layout.

Similar to DS-TION, JavaMy can also visualize common data structure operations such as insertion and deletion. Still, it cannot visualize building data structures from scratch like the Swan system. This is because these two visualization approaches develop specific classes for visualising data structures. These classes are first compiled and then directly mapped to the UI after instantiation, without a physical layer as a buffer like in the Swan system. See Figure2.7.

2.2.4 Latex-based Visualization

```
private String toString( BinNode bn ) {
  if ( bn == null )
    return "";
  if ( bn.left == null &&
       bn.right == null )
    return
      "{ \\fbox{" +
      bn.element.toString() +
      "} }";
  return
    "[ ." +
    "{ \\fbox{" +
    bn.element.toString() +
    "} }\n" +
    ( bn.left != null ? " " +
      toString(bn.left) : "" ) +
    ( bn.right != null ? " " +
      toString(bn.right) : "" ) +
    " ]";
}
```

Figure 2.10: A figure shows codes to produce the latex source code by overwriting the toString() function in Java.[8].

The three AV systems discussed so far have implemented user interfaces, as students may want to learn as spectators. However, they have overlooked that traditional teaching lectures are the most important setting for students as spectators. Latex-based Visualization focuses on lecture scenarios and targets lecturers as its primary users. It automatically generates Latex source codes by overwriting Java's toString() function, thus creating multiple consecutive pdfs for the lecturer. See Figure2.10. Each pdf page contains a running state of the data structure and algorithm[8]. Lectures on data structures and algorithms usually have a lot of content to present in a limited time. Therefore, teachers try to avoid explaining too much about a certain problem, as it would reduce the efficiency of the class[16]. Additionally, preparing a large amount of lecture material is inherently cumbersome. As a result, Latex-based Visualization makes it possible to efficiently demonstrate the algorithm process in the classroom while reducing the burden on lecturers.

This approach to generating complex algorithm execution processes might not be ideal. Creating many PDF pages for each step in an algorithm can result in a large number of pages. Presenting numerous PDFs in a content-heavy lecture could be overwhelming and counterproductive, making it difficult for students to follow and understand the material.

2.3 Summary

This chapter briefly examines the advantages and disadvantages of common web-based visualization systems. It provides a detailed analysis of some notable examples: DS-PITON Algorithm Visualization Tool, Swan Data Structure Visualization System, JavaMy Software, and Latex-based Visualization.

Chapter 3

Requirements and Analysis

Chapter 3 discusses the questionnaire survey's purpose, methodology, findings, and requirements based on the survey and literature review. The chapter also highlights that the survey received ethical approval from the Department of Computer Science at the University of Sheffield. See Appendix A and B for the ethics review, and Appendix C shows the questionnaire survey copy.

3.1 Questionnaire Survey Finding

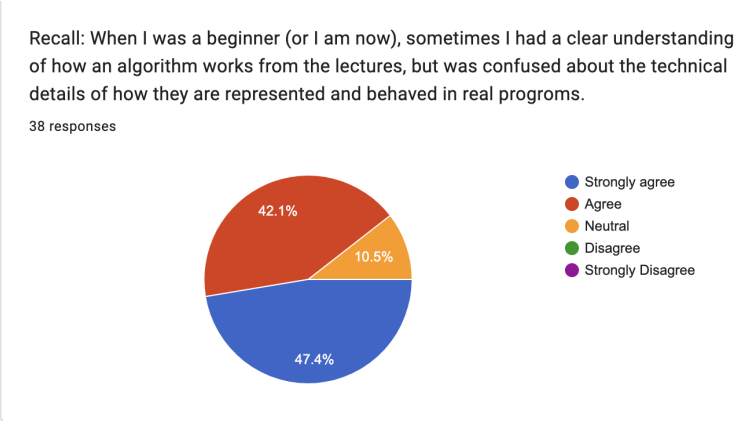


Figure 3.1: Q8: A pie chart demonstrated 89.5% strongly agree or agree the first recall scenario was true.

The questionnaire aimed to investigate beginners' difficulties in learning data structures and algorithms, focusing on the potential gap between theory and practice. Questions explored the prevalence of this gap and whether it could be bridged through continuous learning. Most participants were computer science and STEM undergraduates from TUOS with programming experience, considered beginners. Survey results were analyzed to identify possible causes and inform project requirements. Scene recall questions were included since some participants were postgraduates and might not be beginners at the time of the survey.

The questionnaire featured dynamic questions, requiring participants to answer each one before

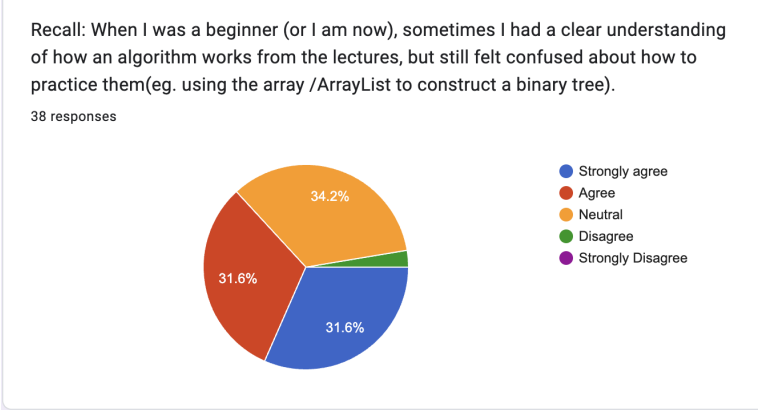


Figure 3.2: Q9: A pie chart demonstrated 63% strongly agree or agree the second recall scenario was true.

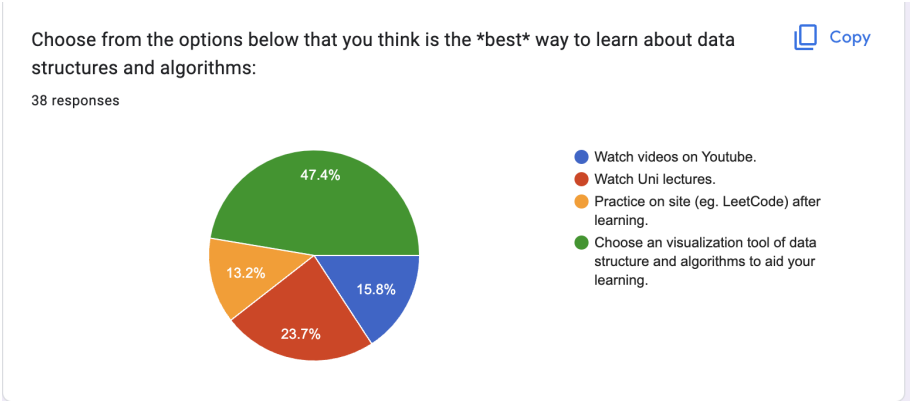


Figure 3.3: Q14: A pie chart indicated 47.4% of participants are willing to choose a visualization tool, and 23.7% of participants believe watching lectures is the best way to learn.

proceeding to the next. This approach guided them towards questions based on their prior responses. Participants answered questions about their experience in the relevant field and their specific difficulties. To avoid leading questions in scene recall situations, five levels of answer options were provided: “Strongly Agree”, “Agree”, “Neutral”, “Disagree”, and “Strongly Disagree”. They were also asked about learning difficulties and selected the most suitable solution from the provided options when facing challenges.

The survey results showed that 51 students participated, with 68.1% from the University of Sheffield and 74.1% having programming experience. Most participants identified as beginners with basic programming skills. In Question 8, 89.5% of participants had experienced confusion in implementing an algorithm after understanding its principle in a lecture. See Figure3.1. Question 9 revealed a 26.5% drop compared to Question 8, indicating that some students can bridge the gap through continuous learning. However, 63% still couldn’t practice the algorithms with deepened learning. The data suggests that beginners may understand data structures and algorithms but

struggle to apply abstract principles to programming, highlighting a "gap" between theory and programming.

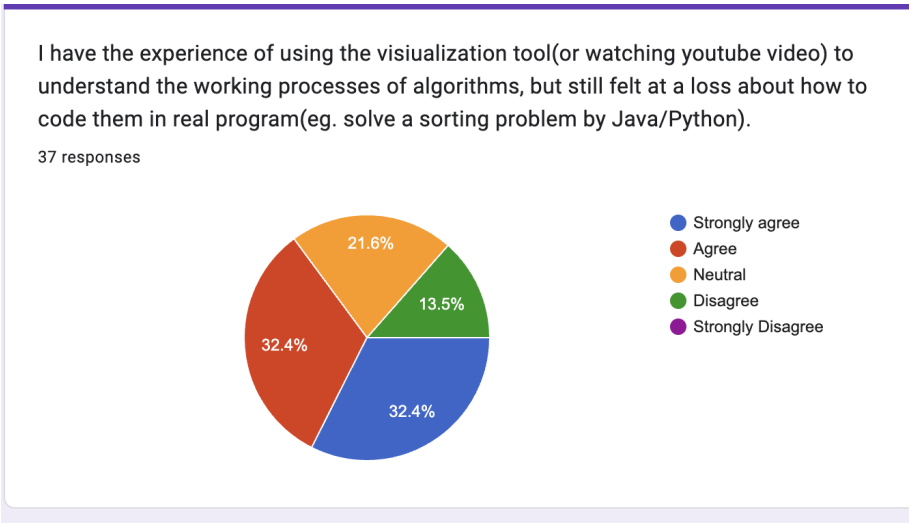


Figure 3.4: Q11: A pie chart indicated 64.8% participants had the experience that it still felt hard to implement a sorting algorithm after using the AV system or watching a related video.

The survey data also revealed participants' preferred methods for learning data structures and algorithms. Among the participants, 47.4% were willing to choose a visualization tool to assist their learning, while 23.7% believed watching school lectures was the best way to learn (Figure3.3). However, although nearly half of the participants appreciated the AV system, 64.8% still found it difficult to implement a sorting algorithm after watching related video teachings or using the AV system (Figure3.4).

3.2 Survey Result Analysis

The survey results show that beginners struggle to bridge the gap between theory and practice in learning data structures and algorithms. Participants also believe that visualization software and university lectures are effective learning methods. This section will discuss potential reasons for these findings.

University Modules Setup

Survey results suggest that university courses on data structures and algorithms often don't emphasize programming skills, requiring students to put extra effort into implementing algorithms. Universities offer programming and intermediate algorithm courses to introduce basic list-based data structures and complex concepts. However, some data structures, like trees and graphs, are more abstract and not directly supported by popular languages (ADS) [17]. This disconnect can confuse beginners and create a significant gap between theory and practice.

Effect of Covid-19

The questionnaire survey investigated students' learning experiences over the past three years, during which the COVID-19 pandemic occurred. The pandemic disrupted offline practice modules, possibly affecting learning. Universities acknowledged the importance of combining theory and practice, designing programming-based practical courses alongside lectures. For example, in TUOS's COM1005 module, freshmen attended face-to-face lab courses to apply Java and algorithms to build rule-based systems[9]. Figure3.5 shows a simple UML diagram of solving the water jugs problem as a practice case in COM1005, helping students bridge the gap between theory and programming. However, the pandemic shifted these practical courses online, potentially reducing interaction and limiting students' practice opportunities as they participated more passively.

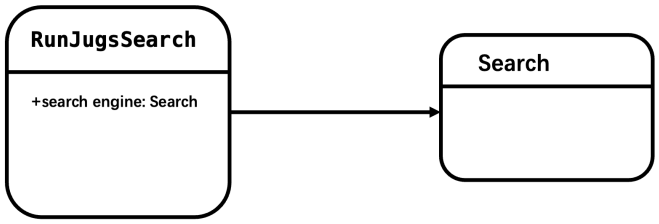


Figure 3.5: A UML diagram shows the association between Search and RunJugsSearch classes, Search class included functions of processing a tree structure[9]

Teaching Method

Teaching teams often use graphical views and pseudo-codes to convey data structures and algorithms processes, but this method has drawbacks. Graphical representations and pseudo-codes improve teaching efficiency by helping students understand complex concepts more easily [16]. However, these static representations can be incoherent, making it difficult for students to understand code examples or pseudo-codes in lectures [6]. As a result, they may struggle to convert static descriptions into dynamic processes and would need to review materials frequently. This issue mainly affects programming implementation, so students' imagination is crucial to bridge the gap between theory and practice.

3.3 Project Requirements

Scenario

The main usage scenario for this software system is within university campuses. Although 47% of survey respondents believe using visualization tools is ideal for learning, a significant portion still prefer watching university lectures. Visualization tools can't replace lectures since they can't cover

all topics like mathematical descriptions and derivations. However, they can address the issue of coherence in static lecture content. Thus, the software system can be specifically used in university lectures and for students' self-learning scenarios.

Features

Features can be considered keywords associated with the software at the beginning of its design. Table 3.1 presents seven software system features, a description of each feature, and the reasoning behind including each feature.

Table 3.1: Features

No.	Feature Names	Descriptions	Reasons
1	**Visualization:** Logical Structure of Data	The logical relationships of data structures, such as trees and graphs, must be visualized in the interface.	Graphical representations of data structures' logical relationships provide a more intuitive display of these relationships and algorithm execution processes. This approach makes it easier for students to understand abstract concepts, which is why it is widely used in teaching[18]. Consequently, graphical representations have become a fundamental feature of data structure visualization systems, as demonstrated by various systems discussed in Chapter 2.
2	**Visualization:** Physical Data Structures	The visualization should not only visualize abstract logical relationships but also attempt to visualize their actual composition in common programming languages, such as the List or Map in Java.	This feature is seen as a solution to address the issues exposed by the survey results, namely, an attempt to help beginners bridge the gap between practice and theory.

Continued on next page

Table 3.1 – *Continued from previous page*

No.	Feature Names	Descriptions	Reasons
3	Animation	An animation that demonstrates the algorithm execution process.	Teachers can use this feature to display animations during lectures, improving lecture material efficiency. The DS-PITON Algorithm Visualization Tool has a similar feature (See Chapter 2, 2.2.1), which has been proven to promote students' unsupervised learning processes[5] effectively.
4	Algorithm Source Code	The source code or pseudo-code related to the algorithm execution process will be displayed on the interface.	The Algorithm-visualizer mentioned in Chapter 2, with 42.7k stars on GitHub, is a popular visualization system. It displays source code on the left side of its interface, with dynamic changes marked as comments. The DS-PITON Algorithm Visualization Tool shares similar characteristics. Survey results indicate that students struggle to bridge the gap between practice and theory. Thus, this thesis project should include a feature to display code or pseudo-code on the interface.
5	Various data structure expressions are displayed on the interface and relevance.	The source code and graphical representation of data structures should be interconnected on the interface. Users should be able to freely define these relationships, such as highlighting the source code that triggers graphical changes or emphasizing its importance with different colours during animation steps.	The first three features describe the content that should be present on the system's interface, while this feature defines the specific behaviour of these contents. It can be considered a concrete solution addressing the usage scenarios and problems faced by the system.

Continued on next page

Table 3.1 – *Continued from previous page*

No.	Feature Names	Descriptions	Reasons
6	Interactivity	Allow editing of visualized data structures and source code. Control the playback of the animation on the interface.	A single shape cannot represent various data structures, such as binary trees and directed or undirected graphs, but can be represented by shapes composed of circles and lines, while sorting algorithms may use rectangles with varying heights for better visualization (Figure2.2). Users should be able to edit these composite shapes. Additionally, given the questionnaire results showing that learning data structures and algorithms is challenging, the visualization system should allow users to control animation playback for repeated learning or reviewing specific algorithm processes.
7	Sharing function	The edited graphics and animations can be shared.	Communication helps beginners absorb unfamiliar knowledge. Therefore, users should be able to share their work. Instructors can share algorithm animations as teaching materials, while students can exchange ideas by sharing their animations. Additionally, students can submit self-created animations through platforms like Blackboard for assignments or coursework.
8	Storage	All edited content should be stored anytime for easy access and sharing.	N/A

Stage Description: What Should be Developed?

The visualization system to be developed should target students and teachers, featuring two main interaction interfaces. One interface is for creating works by teachers, such as designing graphics and editing algorithm source code. The other is for viewing the generated content that faces students, including playing animations with changing graphics and highlighted code. Additionally, the system

should provide teacher-type users with an appropriate method for creating their animations. The backend component controls interface content and responds to interactions, while a data source stores user-generated works.

User Types and Platform

The main users of this system are university teachers and students who can assume roles as a designer, animator, programmer, or viewer. The system must run on devices TUOS teachers, students, and public lab computers use. While mobile devices like iOS and Android are common, cross-platform frameworks such as React Native and Flutter[19] aren't suitable due to limitations such as projection capabilities and some users' device preferences.

The primary devices for teachers and students are laptops and desktop computers. Two options were considered: building a Web application or developing a desktop application.

The decision between these two options depends on whether remote network access is necessary. Firstly, TUOS's IT services department manages lab computers, handling software updates and making the latest software version available. Users can also download the latest version for their devices from BlackBoard so that software doesn't need to be updated by the internet. Second, the desktop application allows for local file storage without requiring network access or additional server-side security measures.

Additionally, the system's sharing feature can be easily implemented with local file storage, while a web application would require more complex database and authentication setups. The system's interactivity aims to bridge the gap between theory and programming practice, and using a desktop application avoids the need for server-side compiler frameworks.

Given these considerations, developing a desktop application for this project is more suitable, as remote network access is not essential.

Developing Language

Chapter 2 highlights the importance of using beginner-friendly languages for visualization systems. As first-year computer science students at TUOS primarily study Java for programming, Java is the most familiar programming language for most students. Moreover, Java provides mature APIs such as Swing for interactive interfaces and rich window component support[20], allowing convenient and lightweight user interfaces without complex web development frameworks or servers. Thus, Java is an appropriate choice for developing this project.

Stories

Based on the identified system features and platform, this subsection will establish more specific functional requirements from the user's perspective. User stories can be employed to specify the necessary functional requirements and help developers manage and concentrate on each requirement[21]. The MoSCoW method will be used to rank each user story, assisting developers and project managers in prioritising requirements and urgency during project development[22]. MoSCoW method categorizes requirements into four categories:

- **Must have** - necessary for project completion.

- **Should have** - priority after Must have requirements.
- **Could have** - lower priority features that add value.
- **Won't have** - not needed in the current version but may be useful in the future.

Each story in the table will support the subsequent software evaluation and testing chapters.

Table 3.2: User Stories

No.	Story	Notes	MoSCoW
1	As a user, I should have a canvas for drawing data structures.	The interface is designed for teachers in the designer role, who will create and save the design for later editing by the animator to create animations, so this requirement is critical.	Must have
2	As a user, I should be able to use at least one shape to draw "nodes" for data structures.	N/A	Must have
3	As a user, I should be able to choose different types of connecting lines for drawing "edges" for data structures.	N/A	Must have
4	As a user, I should be able to define the locations of these shapes.	N/A	Must have
5	As a user, I should be able to adjust the size of these shapes freely.	If time constraints do not allow, appropriate graphics sizes for designing data structures will be provided for users in the system; the function to freely adjust the size will be implemented in the future.	Could have
6	As a user, I should be able to add and delete drawn graphics on the canvas.	N/A	Must have

Continued on next page

Table 3.2 – *Continued from previous page*

No.	Story	Notes	MoSCoW
7	As a user, I should be able to label the components of the data structure graphics with numbers on the canvas.	N/A	Must have
8	As a user, I should be able to add comments to the data structures I have drawn on the canvas.	N/A	Should have
9	As a user, I should be able to move the graphics on the canvas freely.	If time constraints do not allow for it, this requirement will not be implemented, and the user needs to remove existing graphics that are in the wrong position using the delete function and redraw new graphics.	Could have
10	As a user, I should have a player interface for watching animations or previewing the effects of my work.	N/A	Must have
11	As a user, I should be able to see the data structure graphics I have drawn on the canvas with its annotations and added code from the canvas in the player interface.	N/A	Must have
12	As a user, I should be able to play and pause the animation in the player interface.	N/A	Should have

Continued on next page

Table 3.2 – *Continued from previous page*

No.	Story	Notes	MoSCoW
13	I should be able to view the annotations of graphics drawn from the canvas in real-time during the animation playback to help me understand the content.	N/A	Could have
14	As a user, I should be able to forward and backward the animation in the player interface.	Forward and backward functions conflict with real-time user interaction during animation playback because caching all playback content in advance is necessary for these functions, while user interaction is unpredictable and cannot be cached in real-time. These conflicting requirements will be implemented based on time constraints.	Could have
15	As a user, I should be able to watch any step of the animation process by providing the step number.	Suppose time constraints do not allow for it. In that case, this requirement may not be implemented. The user will have to watch specific animation steps manually, forwarding or rewarding, which may not be as convenient.	Could have
16	As a user, I should be able to see the corresponding graphics changes during the animation process.	N/A	Must have

Continued on next page

Table 3.2 – *Continued from previous page*

No.	Story	Notes	MoSCoW
17	As a user, I should be able to see the corresponding code highlighting during the animation process.	N/A	Must have
18	As a user, I should be able to use one graphic at least to present the physical data structures in the programming language to define the steps of the animation.	N/A	Must have
19	As a user, I should be able to display my operations on these physical data structures, such as adding, deleting, modifying, and searching, as well as the dynamic processes of these operations, such as swapping the positions of two adjacent arrays elements.	N/A	Must have
20	As a user, I should also be able to control the corresponding code highlighting, such as highlighting a statement for a while to define the steps of the animation.	N/A	Must have
21	As a user, I should be able to control the changes made to the graphics drawn on the Canvas user interface, such as deletion, insertion, or highlighting, to define the animation steps.	N/A	Must have

Continued on next page

Table 3.2 – *Continued from previous page*

No.	Story	Notes	MoSCoW
22	As a user, I should be able to use this system for learning anytime without being restricted by the network.	N/A	Should have
23	As a user, I should be able to have a large enough player panel to watch more dynamic processes simultaneously.	N/A	Should have
24	As a user, I should be able to save my work locally.	N/A	Should have
25	As a user, I can update my saved work anytime.	If time constraint does not allow for it, this requirement will not be implemented, and the user will need to create a new project to make updates.	Could have
26	As a user, I should be able to share my work.	If time does not allow for it, this requirement will not be implemented, and the user will need to use an external screen recording app to record and share with others.	Could have
27	As a user, I should be able to customize new graphics and add them to the system's graphics library.	This dissertation project will implement a system with basic visualization and animation playback capabilities. Requirements regarding the system's graphics library and custom graphics will be implemented in future versions.	Won't have

3.4 Risks, Security and Project Evaluation

Security

This project is a Java desktop application that relies on local computing power rather than network resources. The research process of this dissertation will involve collecting user information through a questionnaire survey. However, the application does not require users to register or log in, so it will not collect sensitive personal information such as email addresses and names. All information in this project, including graphics designed with the drawing board interface, animations, and inserted code, will not be uploaded to remote servers but stored as local data sources in files and folders on the user's device. Users may need to use this application on public university computers, and their identity will be verified before accessing these computer systems. The security of this part is the responsibility of the TUOS IT Service department.

Risks

The risk associated with this application is that the extensive front-end interaction features on a single interface may lead to a large backend controller responsible for coordinating the implementation of each interface function. Decoupling operations and performing unit and manual testing are essential to mitigate this risk. The project has numerous user stories that must be implemented quickly. Using the MoSCoW method to prioritize these requirements can achieve a minimally viable application, thus reducing the risk.

Project Evaluation

Table 3.1 of User Stories will evaluate how well specific functional requirements have been met after completing the application.

The app aims to help teachers deliver efficient lectures and support beginners in learning data structures and algorithms. After completion, TUOS students and teachers will test the application by trying all features and assuming various system interaction roles, such as animator, designer, programmer, and viewer. They will then complete a questionnaire. The preferred evaluation method involves the Department of Computer Science lecturers testing the app and their students acting as viewers. Alternatively, the app can be provided only to students. The first student creates an algorithm animation and watches demos, while subsequent students watch the previous participant's animation and create new ones. This simulates a "teacher" role during testing. Students will complete a questionnaire and evaluate based on specific criteria:

- **UI design**
- **Functionality practicality**
- **Ease of use**
- **Application features**

Users who use this application, whether teachers or students, need to interact with the interface and system of the application, so UI design should make users feel enjoyable.

Regarding the practicality of the application, this will directly affect the project's purpose. Firstly, students will answer questions about their viewing experience or whether they would be willing to use the system player when learning related knowledge after watching pre-made animations. Secondly, teachers or participants acting as "teachers" will try to generate custom works after watching pre-made animations and answer questions about their experience using the system.

The survey will also include questions to assess the application's usability. The application must enable students to quickly understand the content displayed in each interface area and enable teachers to create animations or build graphics easily.

This chapter presents a feature table for the project, and the success of feature selection will be evaluated by collecting user preferences on features through survey questions.

Chapter 4

Design

This chapter will discuss this project's frontend interface design and backend architecture and how they work together to achieve functionalities.

4.1 Overview

This system is a desktop application implemented using Java, and this overview will briefly introduce its components. The application comprises two main user interfaces, several backend controllers, a model and a data source.

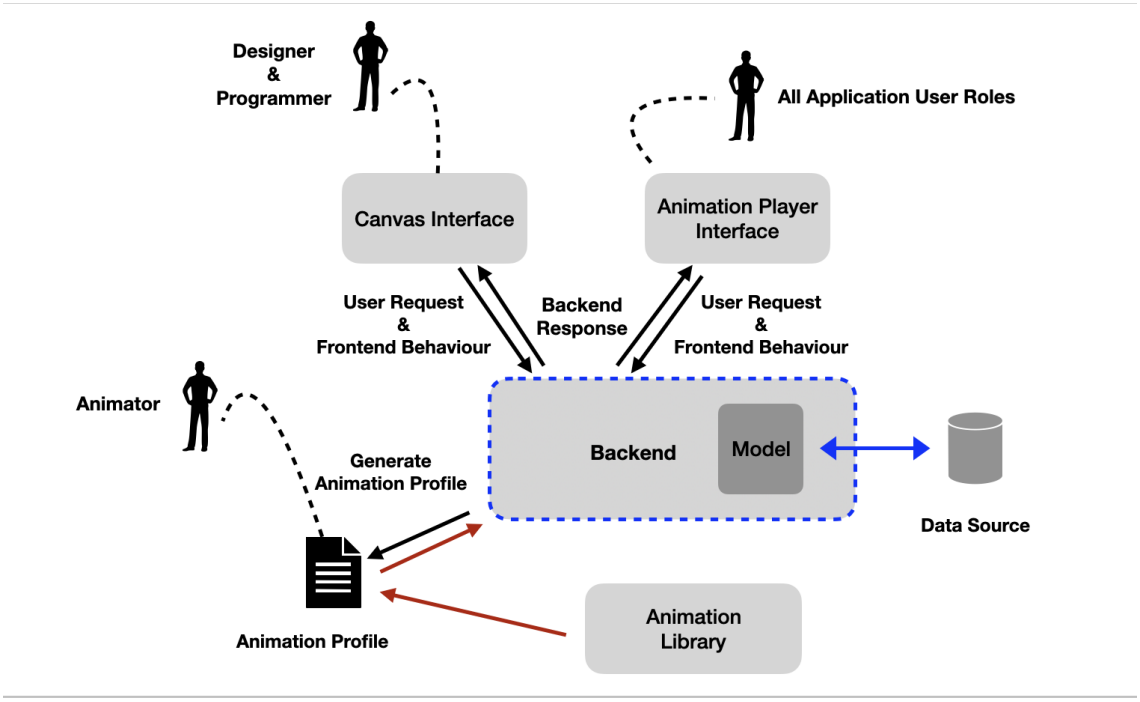


Figure 4.1: The System Overview

Canvas

Canvas is an interface designed for this application's designer role, enabling users to draw data structure relationships using built-in graphics and input devices. Users can add annotations and insert relevant source code or pseudo-code for highlighting per the programmer's role. All Canvas data will be saved to the data source for generating algorithm execution animations.

Animation Player

The Animation Player interface animates data structures and algorithm execution processes using information from the Canvas interface and data source. Viewer role users can watch the demo for learning. Programmer, animator, and designer roles can preview the generated graphic data structures, inserted code, and create animations using the Animation Profile.

Backend

The backend in this application mainly handles user actions, processes data communication between Canvas and Animation Player interfaces, collaborates with the Model to manage data, and generates an Animation Profile for the animator role to create animations.

Animation Profile and Library

The animator role defines the animation for each step displayed in the Animation Player interface by calling the commands from Animation Library or inserting them into the program statements related to the algorithm execution process in the Animation Profile. Using the Animation Profile instead of allowing users to create animations directly on the interface is a balanced choice that will be analyzed later.

Data Source and Model

The project does not require handling massive amounts of data and complex data structures. It only needs to handle simple data structures generated by users locally for visualization purposes. Therefore, the data source will not be a database but a locally generated text file that saves Java instance objects compiled by the Model. To achieve this, a Java class will be designed that uses arrays or other data structures as attributes to store data in the text file.

The system model will then instantiate this class's objects and compile them into the file to implement the writing function. The model should also be able to read and decode the text file as objects to manage the data stored in the file. See Figure4.2.

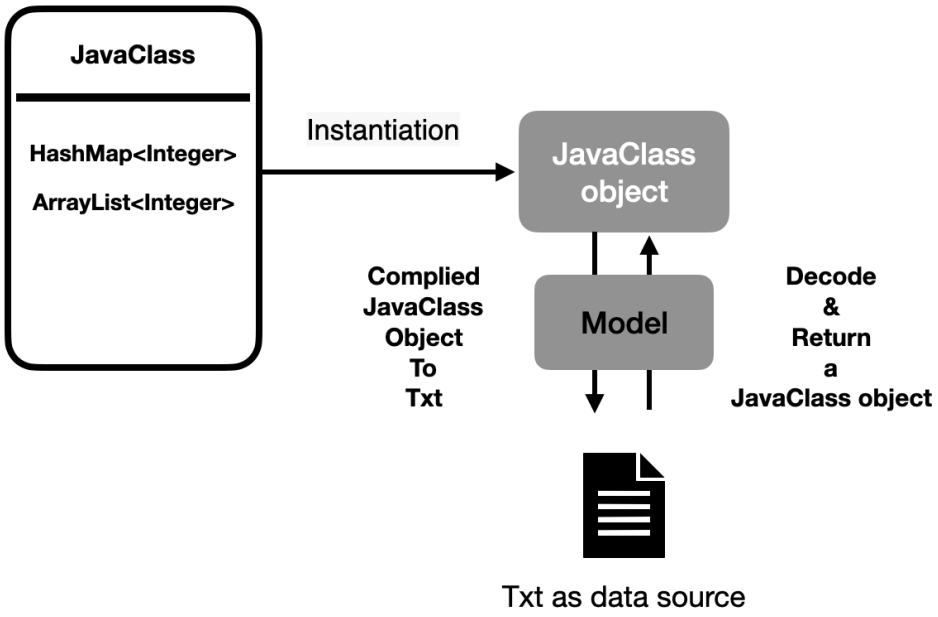


Figure 4.2: The text file as data source work flow diagram

4.2 User Interface

Moqups is a UI design tool that allows users to create visual prototypes or mock-ups of user interfaces before moving on to the functional implementation phase[23]. The Canvas and Animation Player interfaces in this project will use Moqups for creating mock-ups. Mock-ups are essential for better understanding project requirements and serve as a foundation for further iterations and improvements in software and other design fields[24].

Canvas Appearance and Activity

When using the Canvas user interface, there should be an area providing various drawing functions and a sufficiently large area for users to create graphics on the interface using mouse clicks, dragging, or pressing keyboard buttons. As shown in Figure4.3, the Canvas user interface has a menu bar or functional area (Area B) on the right side. It functions to switch or provide different drawing functions by responding to user clicks on the interface buttons. If users want to upload code related to the algorithm execution process, they can paste it into a text input box in Area B. Area A is the space for users to draw graphics using a mouse or keyboard.

Users mainly draw the logical relationships of data structures on the Canvas interface. TUOS beginners in the Com1009 and Com1005 modules are mainly exposed to logical data structures, including stacks, queues, directed graphs, un-directed graphs, and tree structures[10]. See Figure4.4.

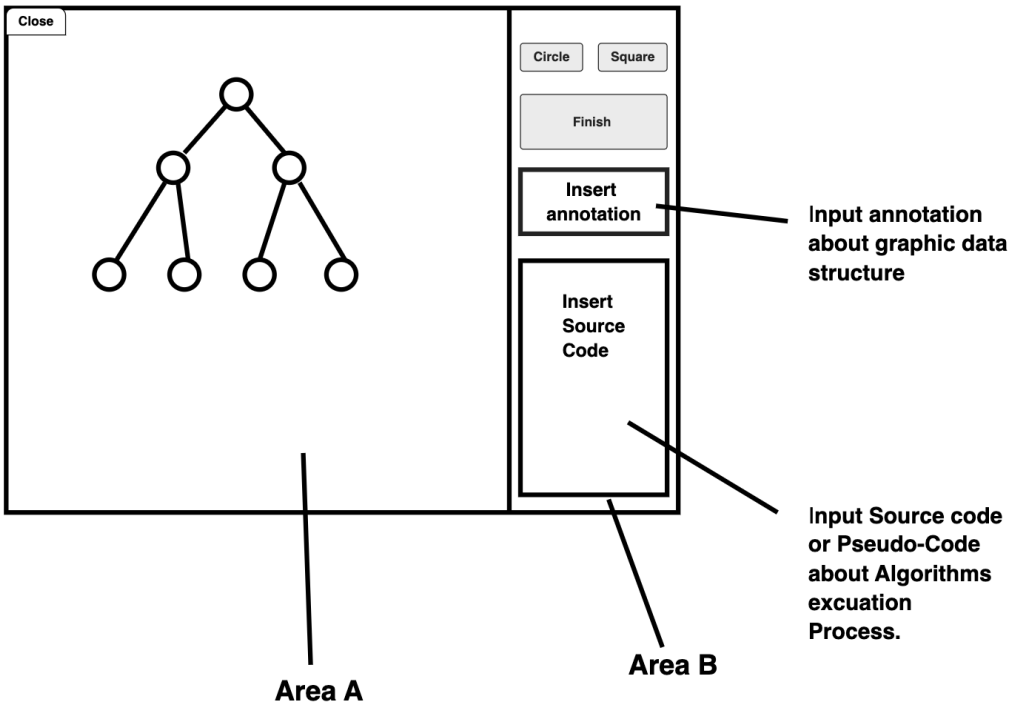


Figure 4.3: The Canvas user interface mocks-up, designed by Moqups.

Stacks and queues can be expressed using arrays commonly seen in programming languages, which will be discussed later.

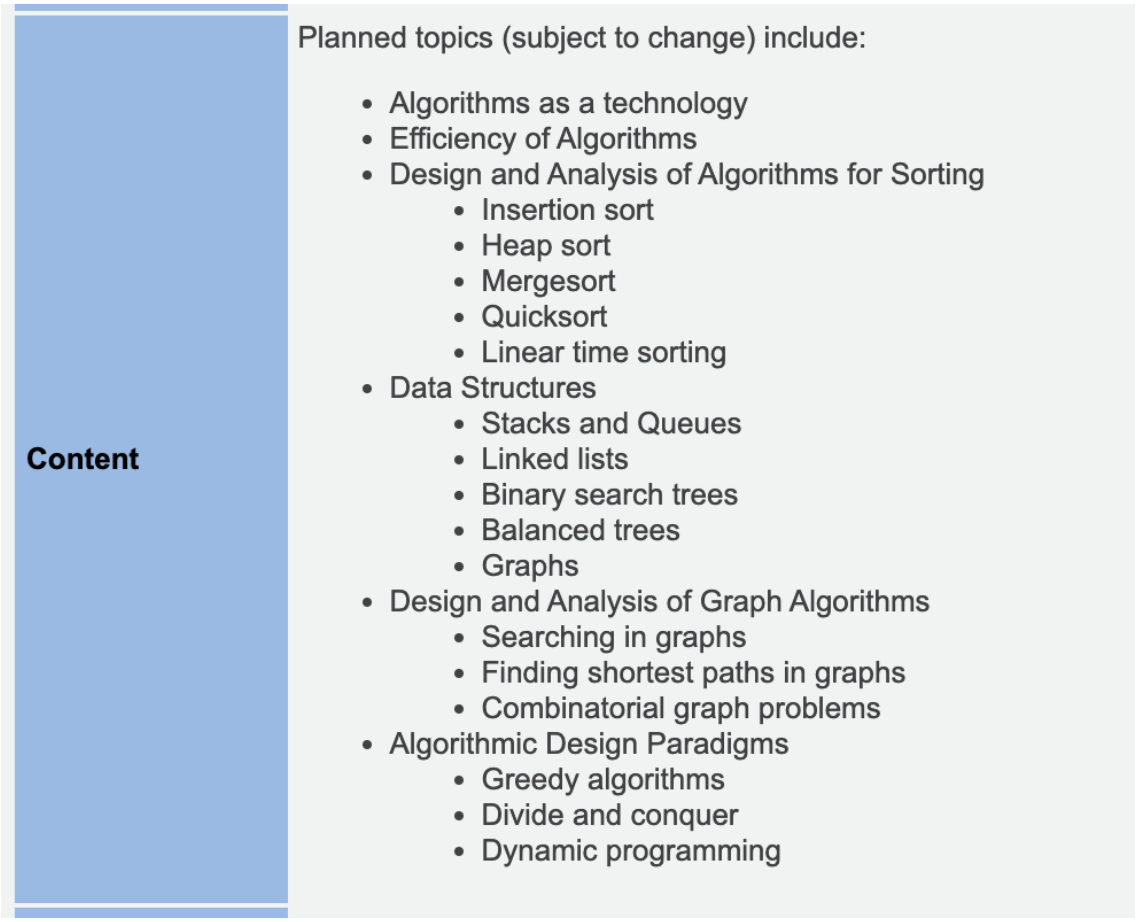


Figure 4.4: The detailed page screenshot of TUOS Com1009 module[10].

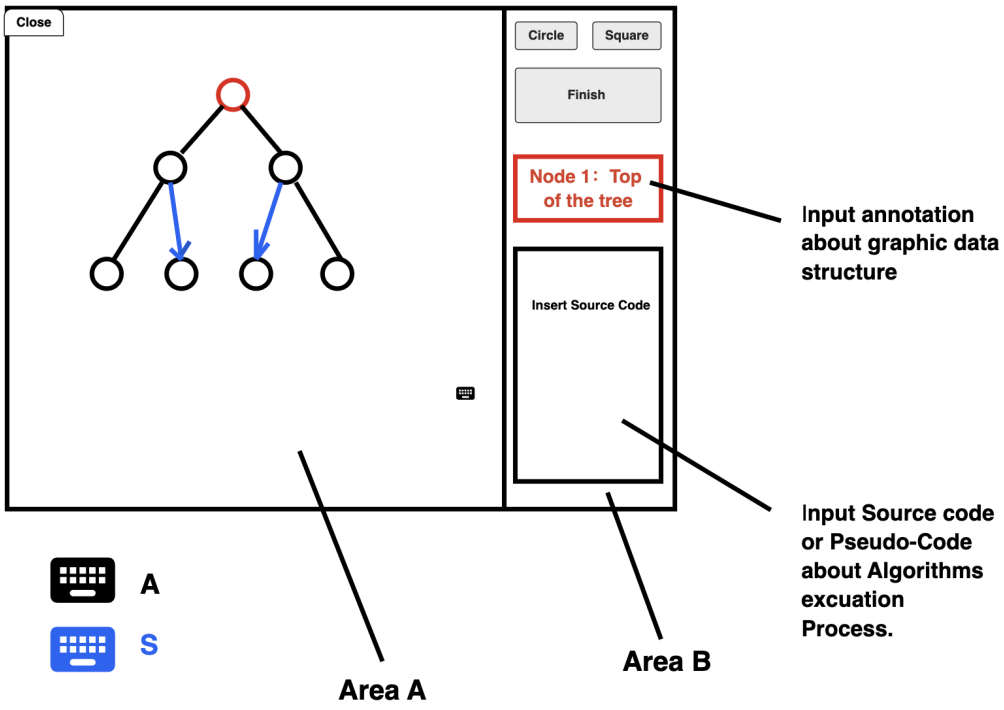


Figure 4.5: The red preview effect occurs when users select a node or plan to add annotations. The 'A' and 'S' keys define blue directed and red undirected edges for data structures. This design is created by Moqups.

The canvas interface enables users to draw three types of logical relationships: directed graphs, undirected graphs, and undirected or directed tree structures. Initially, simple mouse clicks were used, but interaction behaviours were modified to allow annotations. Users now double-click to create nodes and click again for a preview. Annotations can be added in the text area on the right. Some interactions use keyboard inputs to simplify the control panel. Holding 'A' and clicking on two nodes draws an undirected edge, while holding 'S' creates a directed edge. See Figure4.5.

According to the requirements, users can save their work locally by clicking the "Finish" button or closing the window after completing all tasks. The system prompts users to save by creating and naming a new project or allows them to choose not to save their work. See Figure4.6.

Animation Player Appearance and Activity

The animation player of the system is used to display the animations defined by users in the animation profile and is oriented towards multiple system roles. One of the goals of this project is to bridge the gap between theory and practice for beginners. Therefore, this interface will integrate four types of information and display them simultaneously, consisting of five areas: A, B, C, D,

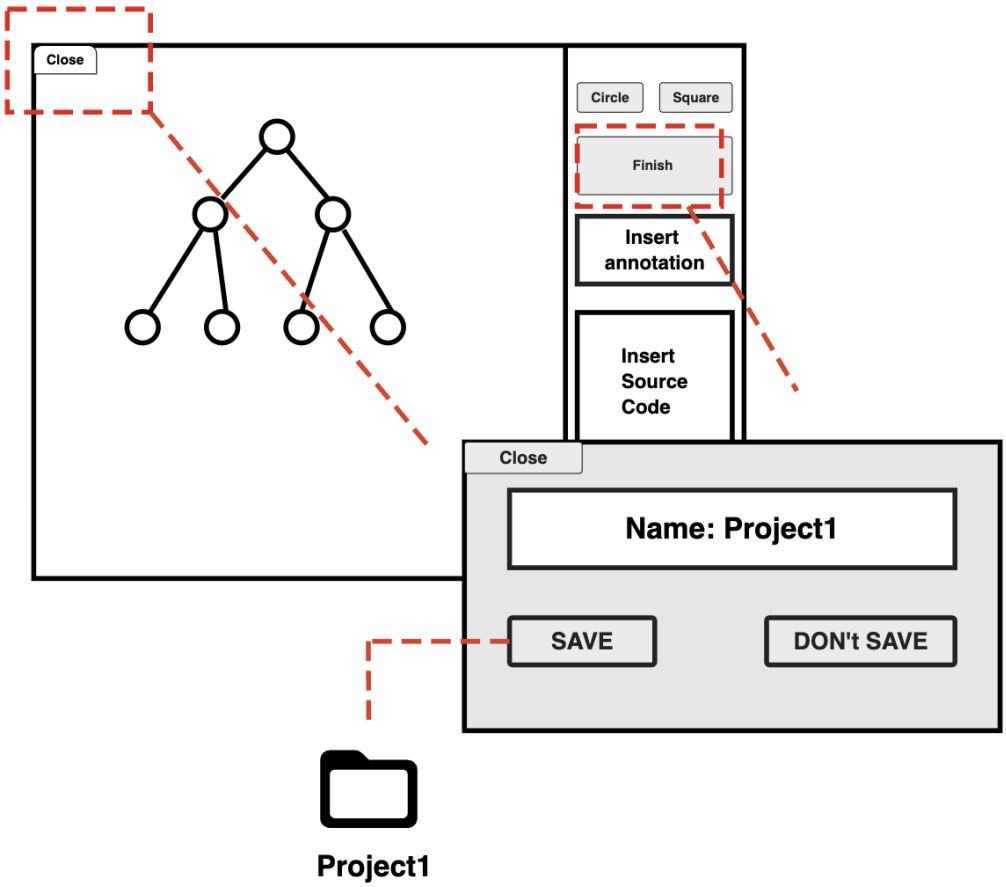


Figure 4.6: The prompt window to create and name a new project after users complete all interface operations.

and F: (See Figure4.7).

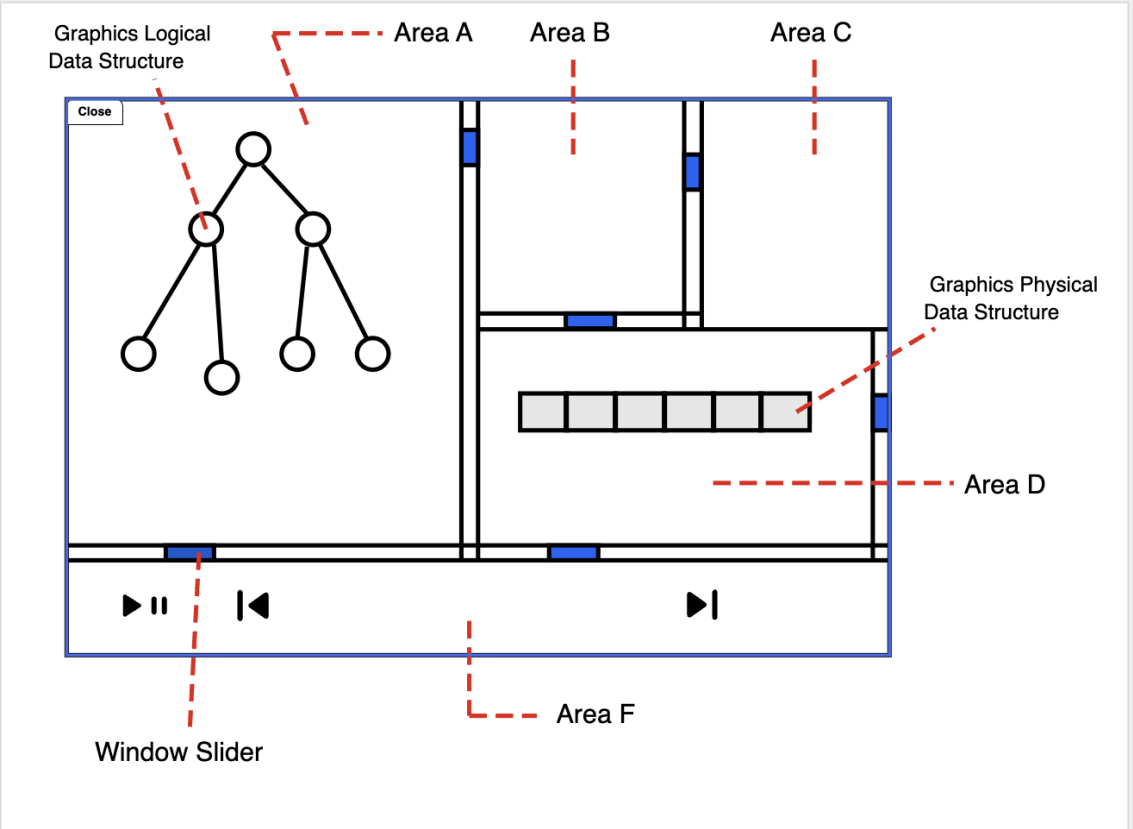


Figure 4.7: The Animation Player user interface mocks-up, designed by Moqups.

- **Area A** - Display dynamic visualization of graphic logical data structures.
- **Area B** - Display dynamic highlight of algorithm source code or pseudo-code.
- **Area C** - Display the annotations of the logical data structure created on canvas UI.
- **Area D** - Display dynamic changes and animated process of physical data structure.
- **Area E** - This is the control panel area of the interface, responsible for implementing functions such as pause or play for the animation.

The player interface is used to preview the generated animation, and users control the playback and pause of the animation through the control panel area E. A complete animation consists of multiple animation steps, each corresponding to the dynamic changes of the graphics in areas D, B, and A. See Figure4.8.

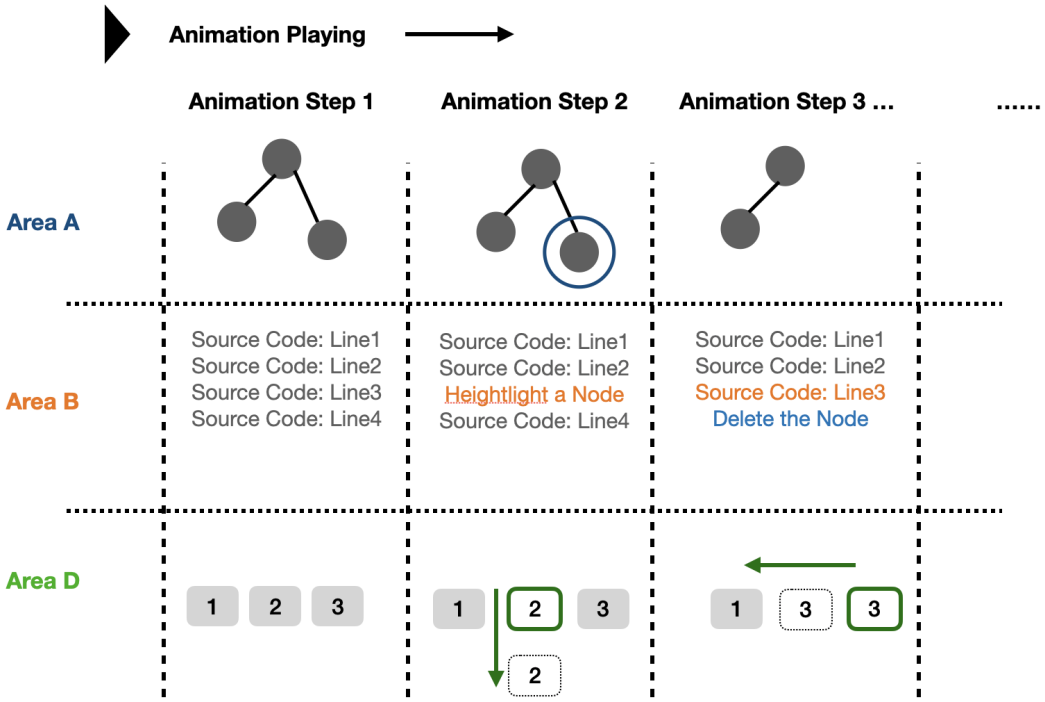


Figure 4.8: Diagram: A example shows dynamic changes of graphic content in each animation step displayed on areas A, B, and D.

Moreover, to avoid confusion among beginner-type users about the logical relationship between the algorithm execution process and data structures when watching animations in the three areas, the interface will allow users to preview the designer's annotations left on the canvas for a specific graphic in area A by moving the mouse inside the graphic during animation playback. The displayed annotations will appear in area C until the user moves the mouse out of the graphic. See Figure4.9.

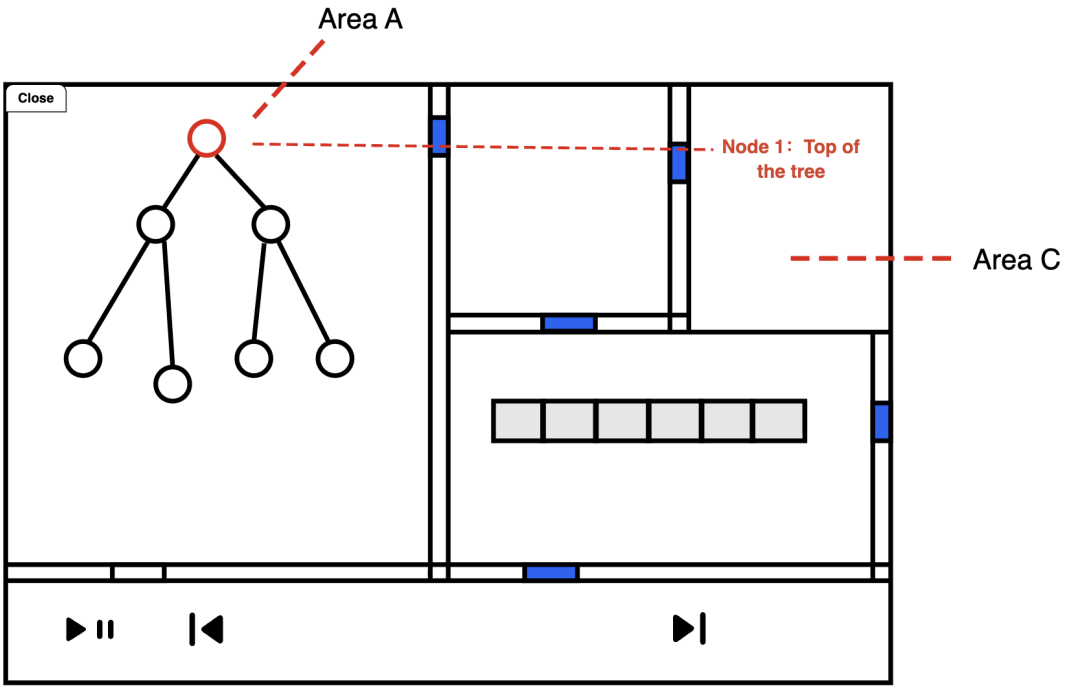


Figure 4.9: Figure shows the preview effect example when users try to figure out the annotation of a node by moving the mouse.

4.3 Backend Architecture

This section will discuss two common backend architectures and conclude with the most suitable architecture for implementation in this project.

Model-View-Controller

MVC architecture is a common backend architecture that separates views, controllers, and models to facilitate software development management[25]. In MVC architecture, the View is responsible for composing the page using various functions, while the Controller responds to changes in view elements through interface listening functions; for example, a button click event from the user will first be passed to the button listener function in the Controller, which then calls functions in the View to change the interface content in response. To enable interaction between the View and Controller, the Controller must know the View's implementation logic; Data is stored in the Model, which retrieves the necessary content data for the new page from the database and notifies the View to update when the user performs a page-turn operation in an application's view[26]. See Figure4.10. In a web application, this notification mechanism between model and view is typically accomplished by the Model generating HTML content and sending it to the View[27]. The observer pattern is used in Java desktop applications to implement the notification mechanism. In the observer pattern, the view class acts as the observer registered with the Model, while the Model is

the observable subject. When data in the Model changes, for example, due to updates brought by read and write operations in the database, the Msodel notifies all registered observers to update[28]. See Figure4.11.

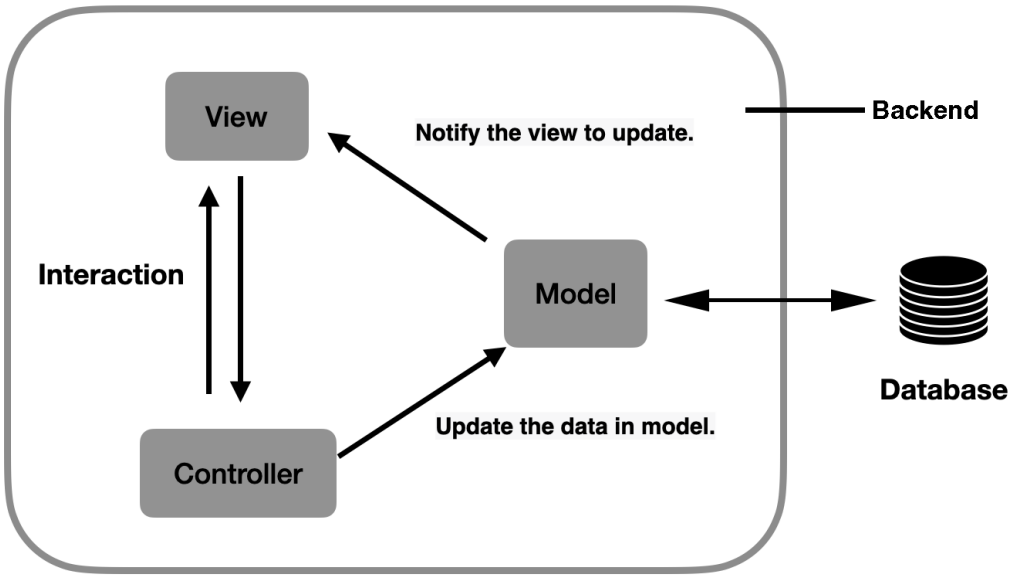


Figure 4.10: The figure illustrates the communication and working methods between the components in the MVC architecture.

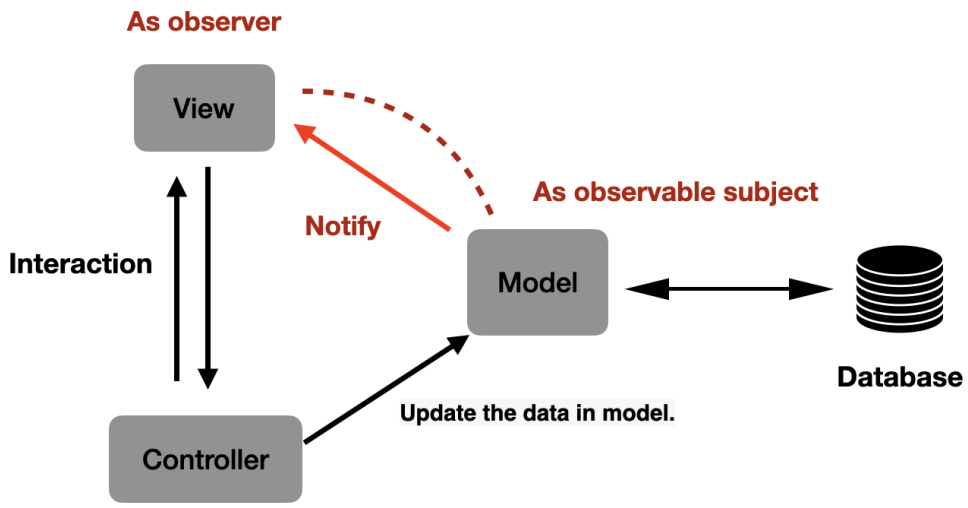


Figure 4.11: The figure shows MVC architecture and observer pattern in java.

Model-View-ViewModel

The MVVM architecture resembles the MVC approach, with the primary distinction being substituting the controller in MVC with the ViewModel in MVVM. Instead of interacting directly with the View, the ViewModel communicates via data binding and event notification[29]. In object-oriented programming languages, this can be explained as the View class creating a ViewModel instance and registering event listeners for view components, such as a button, through this instance. During this process, a one-way dependency relationship exists from the View class to the ViewModel, ensuring that potential software upgrades, even those requiring the entire View class to be removed from the backend, will not cause problems since the ViewModel is inherently versatile.

Data binding allows the View to update without depending on the ViewModel automatically. The underlying principle involves the ViewModel and View sharing a data storage structure, so when the ViewModel updates the data, the content on the View also updates. Conversely, when user actions modify the view data, the ViewModel will also update, establishing bidirectional communication between these two components. See Figure4.12.

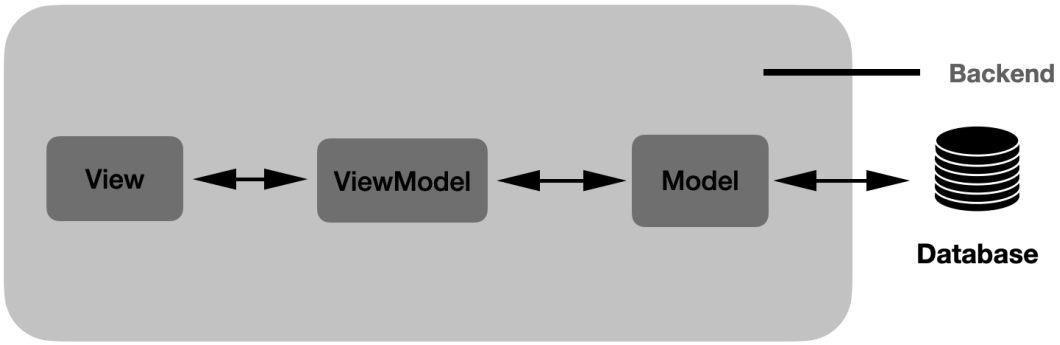


Figure 4.12: The figure illustrates the communication and working methods between the components in the MVVM architecture.

The ViewModel can also act as a "relay station" between the Model and the View. It can convert updated data from the Model into a format displayed on the View and pass information from the View to the Model[29]. See Figure4.12.

Comparison and Summary

As shown in Figures4.10 and 4.12, the MVC architecture is clear, straightforward, and easy to understand, making it suitable for rapid development work. Compared to MVC, the MVVM architecture is more decoupled. Most notably, this architecture does not require direct interaction between the Model and the View. Additionally, due to data binding and event communication, there is less coupling between the View and ViewModel in MVVM. A significant portion of the logic implemented in the controller can be transferred to the ViewModel.

This project is a desktop application involving many graphical user interface interactions. Due to its extensive interaction logic with interface components, it can be expected to encounter difficulties in conducting reliable unit tests because of tight coupling. Moreover, the application includes an Animation Profile independent of the backend for controlling animations. This will involve user editing of interface animations, which may lead to coupling. Therefore, the MVVM architecture, known for its excellent maintainability, will be used to implement this project.

4.4 Animation Profile and Library

The Animation Profile and Library are designed to be independent of the backend architecture. See Figure4.1. The reasons for their existence will be discussed.

What are Animation Profile and Animation Library?

In this project, the Animation Profile and Library are independent of the backend. The Animation Library consists of multiple classes, including graphics for users to edit animations and functions to change the graphics on the interface, making it convenient for users. Java IO is an API in the Java programming language that helps developers handle data input and output while processing the data. Java IO mainly exists in two forms: byte stream and character stream[30]. The Animation

Profile will be an automatically generated .java file using Java IO. Once the user completes the drawing of the logical data structure on the canvas panel and creates a project, it will be generated and stored together with the text file used to save canvas data in the project folder created by the user. The animation profile has pre-written Java source code for launching the application. Users can call functions from the Animation library within the animation profile to create an animation and then start the entire application to preview the animation playback.

Why need the Animation Profile?

There are three reasons why a Java file is needed to serve as an animation profile.

First, using an interface for creating algorithm animations is intuitive but inconvenient. Algorithms provide automation and basic intelligence for programs[18]. And the execution process should not be expressed by mechanically moving shapes step-by-step on the interface. Using the interface interaction method to describe an algorithm's operation process is challenging, even for simple recursions.

Second, using an Animation Profile to create animations is based on survey results. Users, especially teachers, can write actual, executable algorithm source code and insert methods from the animation library within the statements by providing a Java source file. As a result, when the algorithm is executed, the animation generation commands above the statements will also be called to create corresponding animations. This approach benefits beginners, as they can genuinely understand the relationship between algorithms and animations.

Lastly, the application enables users to share without relying on the internet. The Animation Profile contains the code for user-created animations; therefore, by sharing this file, any user who has installed the application can view other users' animated works.

4.5 Summary

Chapter 4 describes the front-end interface design, interface activities, back-end architecture design, and the Animation Profile and Library used for creating animations.

Chapter 5

Implementation and testing

This chapter will describe the specific implementation of the project design through code snippets, pseudo-code, and UI screenshots.

5.1 Implementation

5.1.1 User Interface

The content of subsection 5.1.1 is based on the Oracle JavaSE 7 and JDK 7 API documentation[31].

Canvas

This application's canvas, animation player interface, and pop-up windows will be built using window containers in Java Swing. Swing is a lightweight toolkit for building Java graphical user interfaces, which is itself written in Java. The final canvas interface effects are shown in Figure5.1, Figure5.2, Figure 5.3, Figure5.4 and Figure5.5.

The Canvas user interface's outermost layer is created using JFrame in Java Swing, a common window container for Java GUI applications. Inside the Canvas user interface, two JPanel instances are added to the JFrame container, creating a control area with buttons, a text input area, and a drawing area. See Figure5.6. Pop-up windows in the application are created similarly.

Animation Player

The Animation Player user interface uses the same JFrame outer container as the Canvas UI and incorporates smaller interface containers and components. The final actual effect is shown in Figure5.7.

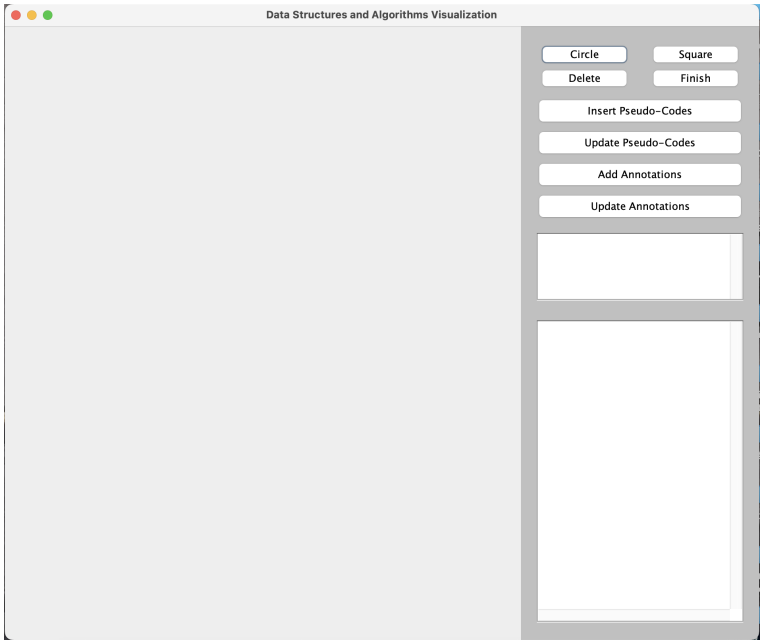


Figure 5.1: The Canvas user interface.

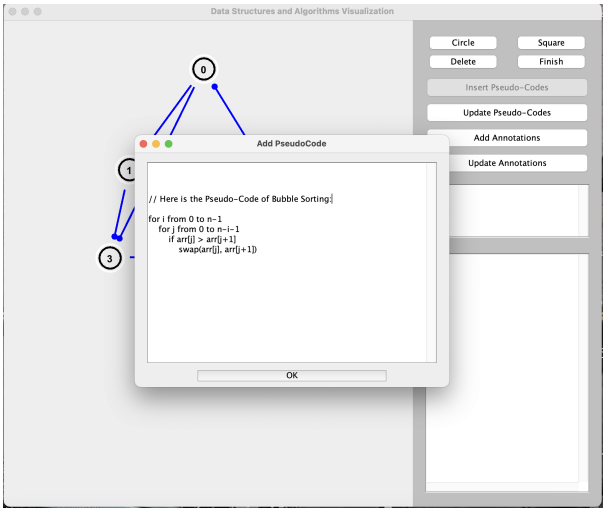


Figure 5.2: The Canvas user interface effect is when the user inputs the source code or pseudo-code to the system by a pop-up window on the interface.

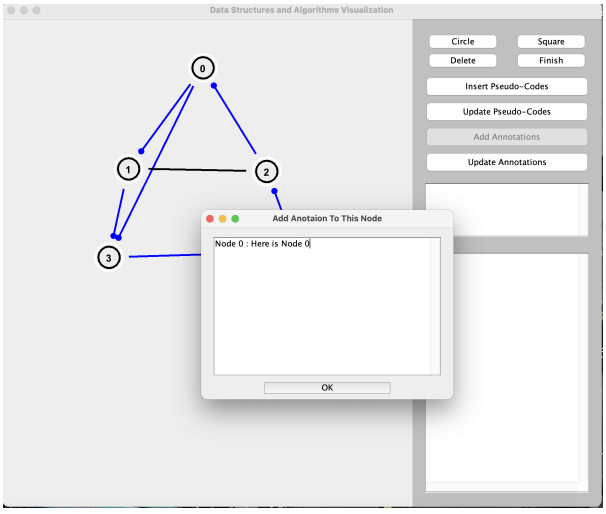


Figure 5.3: The Canvas user interface effect when the user inputs the annotation to logical data structure designed by a pop-up window on the interface.

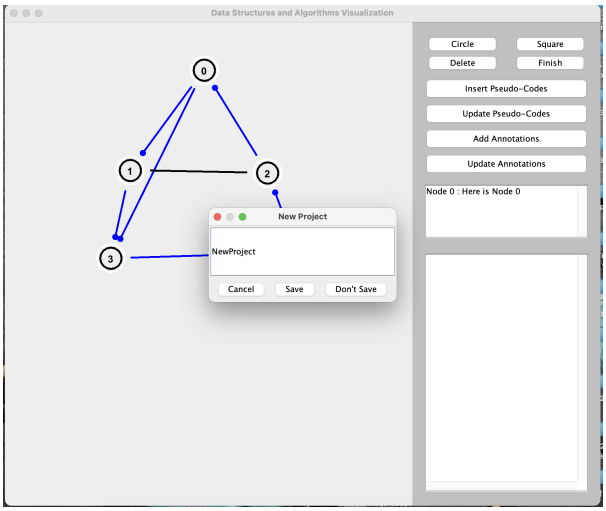


Figure 5.4: The Canvas user interface effect is when the user finishes the design and tries to save or name a new project by a pop-up window.

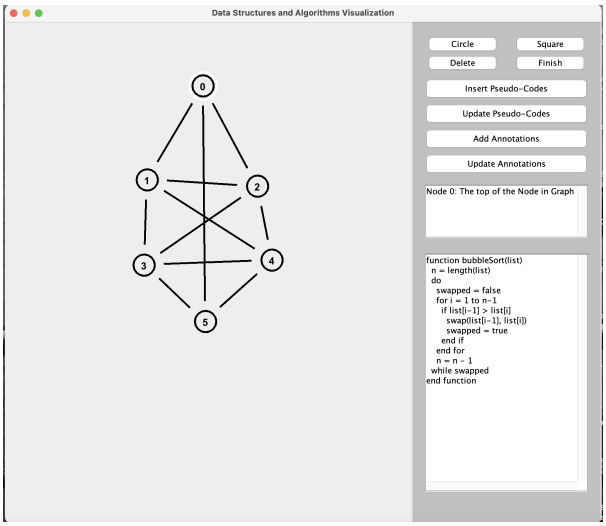


Figure 5.5: The Canvas user interface effect when the user had inserted the pseudo-code or source code about the algorithm execution process, the bigger text input area on the bottom right shows the pseudo-code or source code, the smaller text input area shows the annotations of each node.

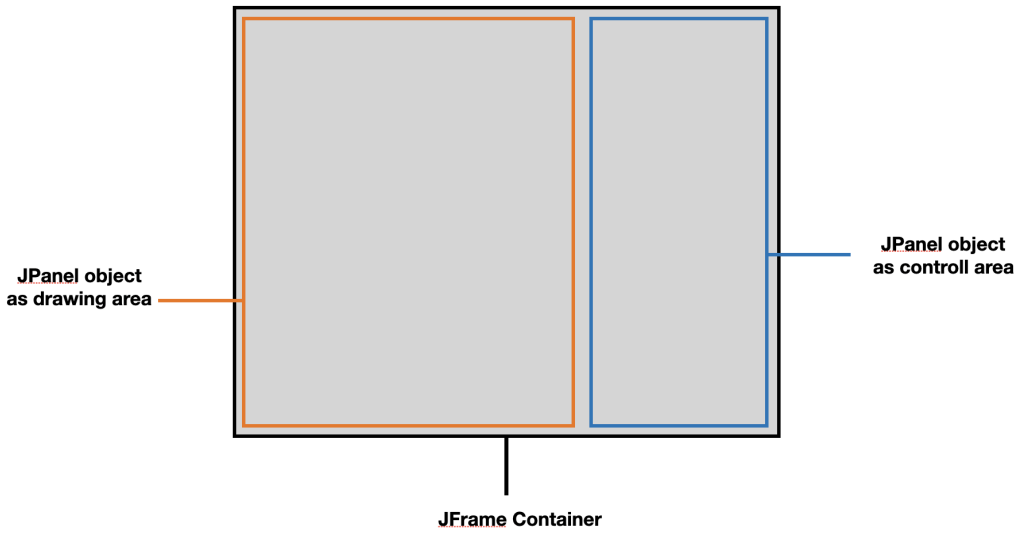


Figure 5.6: Figure shows the Canvas user interface's Layout by Java Swing components: JFrame and JPanel.

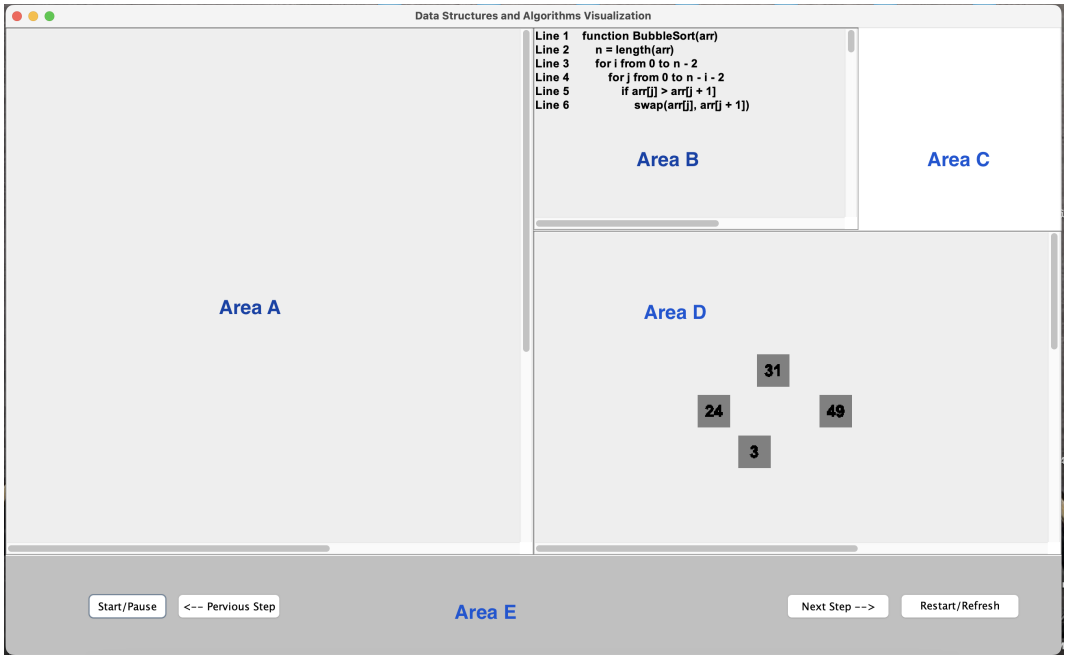


Figure 5.7: The Animation Player user interface effect.

The Animation Player user interface has multiple areas; the specific implementation of each area will be introduced separately:

- **Area A, B and D** - Area A is the largest area on the UI, located on the left side of the interface. It provides ample space for users to view content. A JScrollPane instance creates a scrollable area container with two sliders, allowing users to scroll in all directions. JScrollPane, a Java Swing framework component, can contain a JPanel or any component inheriting from JComponent. When the component size exceeds the visible area of the JScrollPane, scrollbars are automatically added for scrolling and viewing hidden content. A display panel, inherited from the JPanel class, with a size of 1000 x 1000 pixels, is placed inside as a display panel. Areas B and D's layout and implementation are similar to area A, enabling users to scroll panels using sliders or the mouse. Figure5.8 explains the code details for area A, with areas B and D implemented similarly.

```
236
237     /**
238      *
239      * @param coordX
240      * @param coordY
241      * @param panelWidth
242      * @param panelHeight
243      * @returnA ScrollPane object as Area A to show
244      * the logic data structure created in Canvas.
245      */
246     private JScrollPane initLogicDSPanel(int coordX, int coordY,
247             int panelWidth, int panelHeight) {
248
249         LogicDSPanel dpp = new LogicDSPanel();
250         // To Static variable in helper:
251         PlayerUIHelper.setDesignPersentPanel(dpp);
252         JScrollPane scrollPane = new JScrollPane(dpp);
253         scrollPane.setBounds(coordX, coordY, panelWidth, panelHeight);
254         return scrollPane;
255
256     }
```

Figure 5.8: The figure demonstrates a function for implementing area A. The function takes position and size information as parameters, creates an information display panel at line 249, instantiates a JScrollPane to make a scrollable panel at line 252, loads the information panel into it, and returns a complete sub-interface as area A. Areas B and D are implemented similarly.

- **Area C** - Area C, situated in the top right corner of the interface, is used for displaying annotations of graphical information related to logical data structures on the Canvas interface during animation playback. See the actual effect in Figure5.9. Figure5.10 provides the code details for area C.

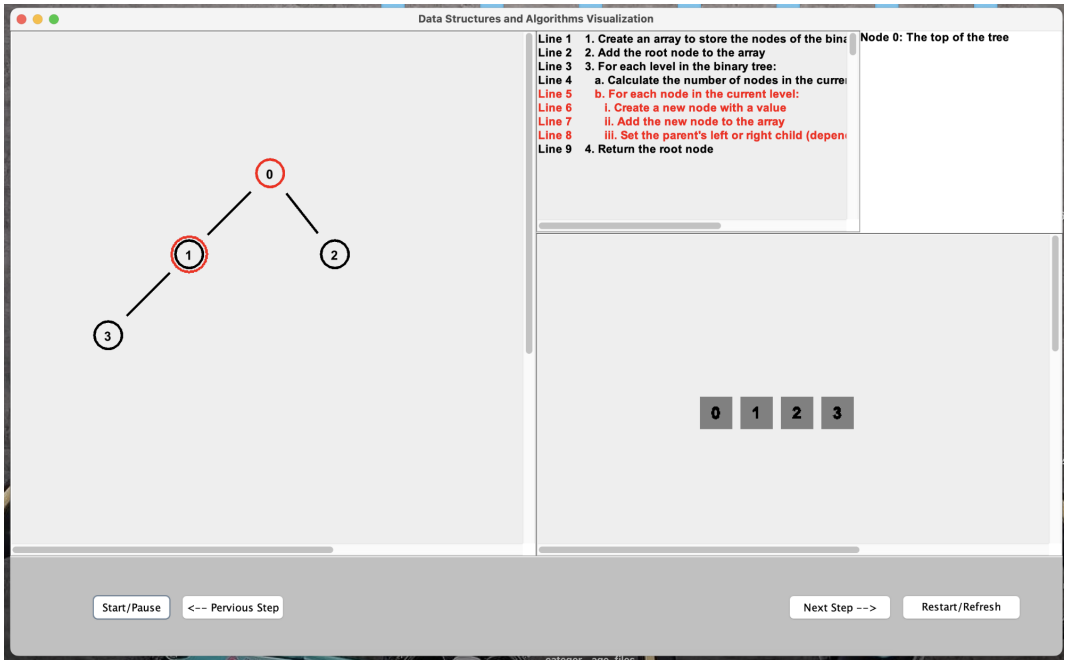


Figure 5.9: The figure shows the actual effect when the user is previewing the annotation about the node in a binary tree created from Canvas, the red circle with 0 on it is the previewed node, and area C shows its annotation: "Node 0: the top of the tree".

```
122
123     /**
124      *
125      * @param coordX
126      * @param coordY
127      * @param panelWidth
128      * @param panelHeight
129      * @return
130      */
131     private JTextArea initAnnotationArea(int coordX, int coordY,
132             int panelWidth, int panelHeight) {
133
134         JTextArea jtextArea = new JTextArea();
135
136         jtextArea.setBounds(coordX, coordY, panelWidth, panelHeight);
137
138         jtextArea.setEditable(false);
139
140         PlayerUIHelper.setAnnotationArea(jtextArea);
141
142         return jtextArea;
143     }
144
```

Figure 5.10: The figure shows the function used to implement the area C. This function takes area C's position and size information, creates a JTextArea object for displaying source code at line 134, and finally returns the object as a complete sub-interface.

- **Area E** - Area E is the control panel at the bottom of the Animation Player user interface, which controls the animation's pause and plays.

5.1.2 Graphics Drawing

The content of subsection 5.1.2 is based on the Oracle JavaSE 7 and JDK 7 API documentation about Java Swing[31].

The application's drawing functionality is mainly implemented on the Canvas user interface, where users can draw basic data structures from the Com1009 and Com1005 modules using shapes and lines, as shown in Figure4.4. These basic data structures, such as queues, are list-based structures often represented by arrays in programming languages. Users will customize these list-based data structures to create animations with the support of the library. Since arrays are common physical data structures, this process can visualize basic physical data structures. Furthermore, list-based structures are frequently used to construct abstract logical structures in programming languages. By placing them in the animation player user interface for user customization, their relationships can be dynamically expressed, showing changes in logical and physical data structures in actual programming languages. This subsection will discuss the implementation of tree and graph structure drawing functionality on the Canvas user interface.

Choose the shape

Users can choose between square and circle shapes for drawing nodes by clicking buttons on the Canvas UI control panel. See Figure5.11. When a button is clicked, an ActionListener in Java responds. ActionListener is an interface in Java GUI programming that listens for user actions like button clicks or menu selections. It is registered on a component and requires implementing the actionPerformed method. This method is automatically called when an event is triggered, allowing for proper processing. Figure5.12 shows the code snippet for implementing the actionPerformed method for shape selection.

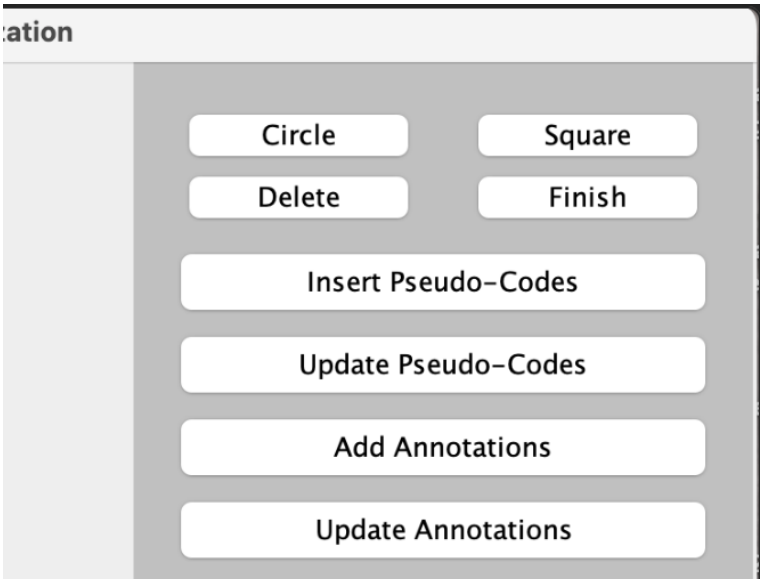


Figure 5.11: The figure shows buttons in Canvas UI.

```
1049    @Override
1050    public void actionPerformed(ActionEvent e) {
1051
1052        if (e.getActionCommand().equals("Circle")) {
1053
1054            this.app_UI.requestFocusInWindow();
1055            this.isCircleBtn = true;
1056
1057        }
1058
1059        else if (e.getActionCommand().equals("Square")) {
1060
1061
1062        }
```

Figure 5.12: The code snippet shows the actionPerformed method to respond to the users' button clicks of "Circle" or "Square".

Drawing A Node

After the user selects the desired shape, they can double-click on the Canvas to place it. This can be accomplished using an interface called MouseListener. The class implementing this interface must override the mouseClicked method to respond to user mouse click actions. A numbered shape can be placed on the canvas by calling a drawing function within this method. See Figure5.13

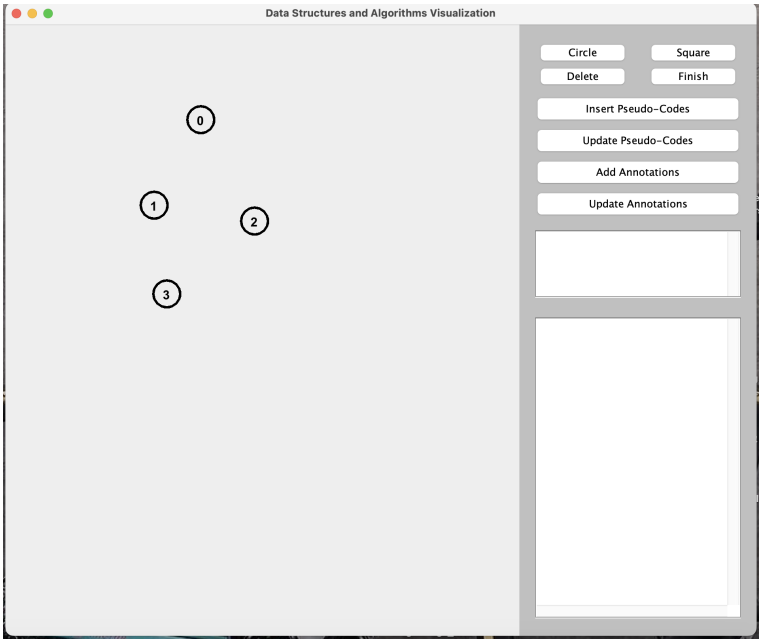


Figure 5.13: The Figure shows four numbered circles as nodes have been left on the Canvas.

Link Nodes

Users need to hold down the keyboard's 'A' and 'S' keys while clicking on two pre-drawn shapes to connect them with directed and undirected connections. Specifically, holding down the 'A' key

will draw a black line between the two shapes, while holding down the 'S' key will draw a blue line between them, with the endpoint of the blue line segment indicating its direction. See Figure5.14. Both keyboard and mouse listener interfaces are required to implement this functionality, and their respective methods must be overridden to make these functions work together.

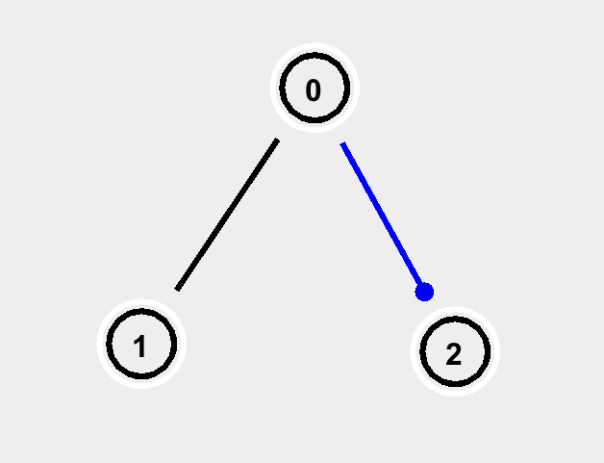


Figure 5.14: The figure shows a structure on Canvas with black un-directed and blue directed connections, the blue connection line in the figure indicates that Node 0 points to Node 2.

The application uses HashMap to store and manage the graphical information drawn on Canvas. HashMap is a data structure in Java that allows storing key-value pairs using hash tables, enabling fast data search and access[32]. Features such as deleting or drawing new shapes require iterating through the data structure, performing the necessary operations, and refreshing the interface. The program's efficiency depends on the time complexity of traversing the data structure. HashMap has unordered elements and finds values by converting keys using a hash algorithm[33]. This eliminates the need to traverse each element individually, reducing time complexity. Therefore, HashMap is a suitable choice for managing interface graphical information.

Algorithm 1 Pseudocode for Node Drawing and Connection

```
1: function drawNode(x, y):
2:     key = [x, y]
3:     hashMap[key] = []
4: function connectNodes(node1_key, node2_key, undirected):
5:     hashMap[node1_key].append(node2_key)
6: if undirected then
7:         hashMap[node2_key].append(node1_key)
8: end if
9: function main():
10:    // When the user draws a new node on the interface
11:    drawNode(x, y)
12:    // When the user establishes a connection between two nodes
13:    connectNodes(node1_key, node2_key, undirected)
```

When a user draws a new node on the interface, a new key must be created in the HashMap.

The key is represented by an array that stores the Node coordinates. Once the user establishes a connection between two nodes, an array will be created as an adjacency list to store all the successors of a node and form a complete key-value pair structure with the corresponding key. If the user establishes an undirected connection, the two connected nodes are each other's successors, meaning two key-value pair structures will be created in the HashMap. Insertion operations will be performed as the number of successors of the nodes increases. The above process will be repeated as the user draws more complex shapes or connections between nodes on the interface. See Algorithm 1 Pseudo-code for Node Drawing and Connection for the algorithm.

Delete Nodes

Users can delete drawn nodes by clicking on the node and then clicking the "Delete" button. The program removes the node and its connections with child nodes from the HashMap, stores the interface graphics, and then refreshes the interface to reflect the updated HashMap.

Algorithm 2 Pseudocode for Removing a Node and Its Connections from a Mixed Graph or Tree

```
1: function removeNodeConnections(node_key):
2:     // Iterate through the neighbors of the node to be removed
3: for neighbor_key in hashMap[node_key] do
4:         // Remove the connection to the node from the neighbor's adjacency list
5:         hashMap[neighbor_key].remove(node_key)
6: end for
7:     // Remove the node from the hashMap
8:     del hashMap[node_key]
9: function main():
10:    // When the user requests to remove a node and its connections
11:    removeNodeConnections(node_key)
```

Algorithm 2 outlines the process of updating a data structure. Its core function, "removeNodeConnections", removes a node and its direct connections from a mixed graph. It iterates over the node's neighbours, deletes the connection from each neighbour's adjacency list, and removes the node from the HashMap. The node_key represents the key of the node to be removed. The main function simulates user interaction with the canvas user interface, and when the user requests to remove a node and its connections, the "removeNodeConnections" function is called.

Refreshing the interface means the program must redraw the updated HashMap onto the interface. Therefore, a corresponding redraw function must be written to map the HashMap. Algorithm 3 presents the pseudo-code for the redraw algorithm. The core function, "drawMixedGraphOrTree", iterates through each node key in the HashMap and uses the 'drawNode' function to draw nodes on the interface. It then checks each connection's direction, using "drawDirectedEdge" for directed connections (blue lines) and "drawUndirectedEdge" for undirected connections (black lines). The drawing effect is shown in Figure5.14.

Deleting a node involves three algorithms: creating and inserting elements into the HashMap, deleting elements from the HashMap, and mapping the HashMap. As these algorithms almost

Algorithm 3 Pseudocode for Drawing a Mixed Graph or Tree on Interface

Input: hashMap
Output: The mixed graph is drawn on the interface

```
1: function drawNode(node_key):
2:     Draw a circle at the coordinates given by node_key
3: function drawDirectedEdge(node1_key, node2_key):
4:     Draw a blue line from node1_key to node2_key with an arrowhead pointing to node2_key
5: function drawUndirectedEdge(node1_key, node2_key):
6:     Draw a black line between node1_key and node2_key
7: function drawMixedGraphOrTree(hashMap):
8:     FOR each node_key in hashMap:
9:         drawNode(node_key)
10:        FOR each connection in hashMap[node_key]:
11:            IF connection is directed:
12:                drawDirectedEdge(node_key, connection)
13:            ELSE:
14:                drawUndirectedEdge(node_key, connection)
15: function main():
16:     drawMixedGraphOrTree(hashMap)
```

cover handling all interface behaviours, they will also be used in the Animation Player user interface. To adhere to the DRY (Don't Repeat Yourself) principle in the programming process and benefit the implementation of other functionalities, these three algorithms will be placed in GraphicAlgorithms.java and exist as static methods. Static variables and static methods in a Java class belong to the class and cannot be called by the Java class instances[32]. Therefore, other classes can directly call these functions using the class name followed by the method name, which is convenient.

Add Annotation to Node

Users can add annotations to each drawn node. They can paste the annotations in the pop-up window by clicking on a node to select it and then clicking the "Add Annotations" button on the interface. Subsequently, the program will store a key-value pair in a HashMap with the node as the key and the annotation for that node as the value.

Insert Pseudo-code or Source Code

Adding source code or pseudo-code related to the drawn data structure to the system is similar to adding annotations. However, the annotations are stored in an array, with each line of the annotation serving as a separate array element. This approach enables the implementation of line-by-line code highlighting functionality in the subsequent Animation Player user interface.

5.1.3 Data Saving and Loading

When users finish drawing the data structure on the canvas interface, they can click the "Finish" button to create and name a new project folder. This folder stores their drawn graphics, graphic

annotations, and inserted system source code. Upon creation, an Animation Profile is automatically generated for customizing an animation related to the drawn data structure. Figure5.15 illustrates the folder contents. The text file saves the graphic data, including drawn graphics, graphic annotations, and inserted system source code.

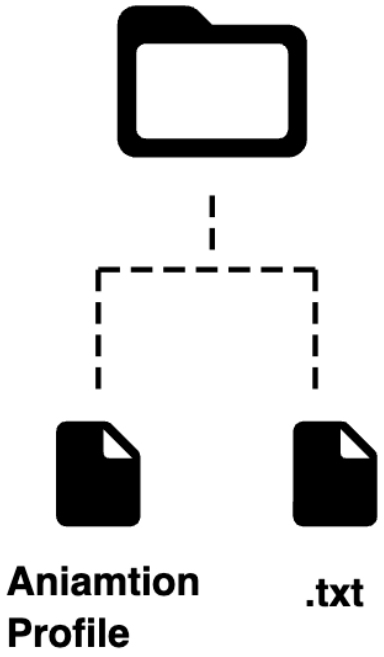


Figure 5.15: The figure shows a newly created folder after the user has finished the data structure design on canvas UI.

Generate Animation Profile and Data Management

In this application, the model compiles a Java object into a data source and reads or writes data to the target data source according to the data source path. Java IO is needed to implement this process, a core library in the Java programming language for handling input and output operations, such as file reading, file writing, and data stream processing[30]. By using the Java IO library, developers can easily handle files and data streams to achieve data storage and transmission.

Figure5.16 shows the writeObject function in the Model. Line 59 checks if the target text file exists and, if not, creates a new file. At line 65, An ObjectOutputStream instance writes Java objects containing canvas graphic information, such as HashMaps for graphics and annotations, and an array for source code, into the text file. Figure5.17 presents the readObject function, which reads data from the text file and returns the object.

After the user clicks the "Finish" button, the program will display a pop-up window, as shown in Figure5.14. Users can name the new project folder here. Subsequently, the program will create a folder with the user's specified name in the application's project folder under the src/user/ directory

```
49
50    public static <T> void writeObject(T t,String path,String... strs){
51
52        String str = "";
53
54        for (String s:strs) {
55
56
57            str = str + "/" + s;
58            path = path+"/"+s;
59            DAO.isHaveFile(path);
60        }
61
62        File file = new File(path+"/"+t.getClass().getSimpleName()+".txt");
63
64        try {
65            ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(file));
66            oos.writeObject(t);
67            oos.close();
68            System.out.println("===== test: UtilObjectIO: WRITE" + str +"/"+ t.getClass().getSimpleName() + "Success! =====");
69        } catch (IOException e) {
70            System.out.println("===== test: UtilObjectIO error: Write exception ..."+e.getMessage() + "=====");
71            e.printStackTrace();
72        }
73    }
```

Figure 5.16: The figure shows the writeObject function in the model.

```
20    public static <T> T readObject(Class<T> clss,String path ,String... strs){
21        String str = "";
22        for (String s:strs) {
23            str = str+"/"+s;
24            path = path+"/"+s;
25            DAO.isHaveFile(path);
26        }
27        File file = new File(path+"/"+clss.getSimpleName()+".txt");
28
29        T t = null;
30
31        try {
32            ObjectInputStream ois = new ObjectInputStream(new FileInputStream(file));
33            try {
34                t = (T) ois.readObject();
35                System.out.println("test: UtilObjectIO: READ"+str+"/"+clss.getSimpleName()+"Success! ");
36                ois.close();
37            } catch (ClassNotFoundException e) {
38                System.out.println("test: UtilObjectIO error:"+str+"No File"+e.getMessage());
39                e.printStackTrace();
40            }
41        } catch (IOException e) {
42            System.out.println("test: UtilObjectIO error: Read exception ..."+e.getMessage());
43            e.printStackTrace();
44        }
45        return  t;
46    }
```

Figure 5.17: The figure shows the readObject function in the model.

and finally call the writeObject function in the model with the folder's directory as a parameter to complete the creation of the text file.

Figure 5.18: The figure shows the code snippet and partial explanation of the Java File Generator.

This application's JavaFileGenerator class creates Java files as Animation Profiles in the project folder. It stores the content to be written into a Java file in a String variable called "fileContent." A File class instance from the Java IO library generates a Java file, and a FileWriter instance writes the content into it. This process uses the path of the user-created project folder as a parameter. See Figure5.18.

Load Graphic Data and Launch Animation Player

After naming and creating the project folder, the program should automatically display the Animation Player user interface and map the graphic data from the text file to Area A on the interface for users to preview the effects. As discussed earlier, to facilitate sharing animation functionality, the Java source file serving as an Animation Profile has the Animation Player user interface startup function written in by the file generator. This requires the program to compile it into a .class file and then run it. The JavaCompiler, an interface in the Java programming language for dynamically compiling Java source code, can compile the code in the Java file; this interface is usually found in the javax.tools package and provides a programmatic way to compile source code[34]. Then, use functions from the Java IO library to locate the path of the .class file. Finally, use Java reflection with the class file path as a parameter to run the compiled file and launch the Animation Player. Reflection is a powerful feature that allows programs to inspect and modify object behaviour at runtime, often used in reverse engineering to explore a legacy system's "black box"[35].

5.1.4 Play Animation

The content of subsection 5.1.4 is based on the Oracle JavaSE 7 and JDK 7 API documentation about Java Swing[31]; the Java Runnable Interface[36].

The Animation Player UI displays different dynamic change information in various areas. Figures 4.8 and 5.19 demonstrate this process, with all areas forming a complete animation of the algorithm's execution.

On the Animation Player user interface, the three main areas, A, B, and D, are implemented using outer JScrollPane instances as containers, each containing a larger display panel instance of a class inherited from JPanel. These display panel instances will be used to carry the dynamic visualization content defined by the user.

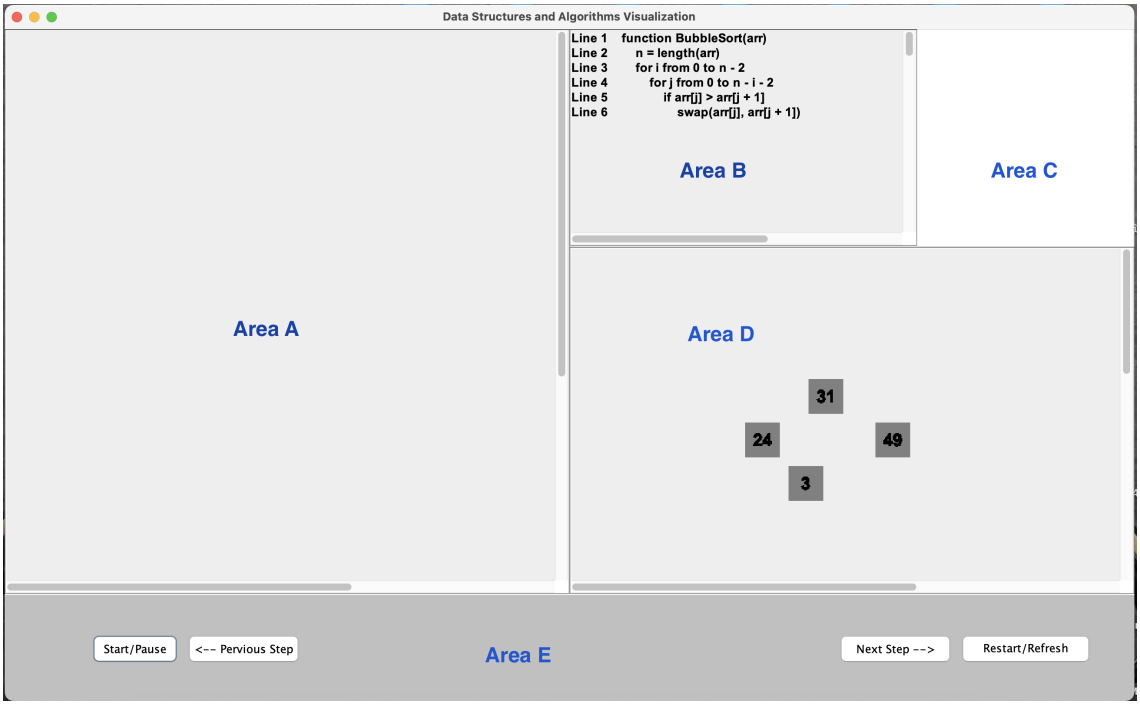


Figure 5.19: The figures shows the different areas on the Animation Player user interface.

The key to implementing the animation playback functionality lies in overriding the paintComponent method of the custom JPanel component. This method is automatically called when the JPanel component and its subclasses are executed, i.e., when the interface is launched. When the repaint method is called on instances of the JPanel component and its subclasses, the paintComponent method is also invoked. The graphic drawing logic on the interface can be implemented using the Graphics g parameter as a brush within this method. After changing the position information of the interface graphics and calling the repaint method, the movement of the interface graphics can be realized, which is a frame animation. Repeating the above process and atomizing the changes in position information will draw a complete and smooth dynamic changing process on the interface and can also implement the deletion and insertion of graphic nodes.

However, this application places large panels displaying dynamic changes in ScrollPane containers.

In Java GUI programming, scrolling these components causes the paintComponent method to redraw graphics, interrupting the animation. Java Swing's single-threaded model may lead to thread blocking and stuttering animations due to user scrolling and animation playback. To address these issues, the application uses Java multithreading and double-buffering techniques.

Java Multithreading

The application will have two threads: the main and the animation threads. The animation thread handles drawing and dynamic graphics changes in the Animation Player user interface's three areas, while the main thread redraws the region panels when users scroll. Both threads must call the repaint method on the same three display panel instances, sharing the same three display panel objects and their Graphics g brushes. To enable resource sharing between multiple threads, a class called PlayerUIHelper will be created, defining several sets of static object variables and encapsulating them as a shared repository between the two threads. This programming pattern is known as the Singleton pattern. The Singleton Pattern ensures a class has only one instance in an application; it provides a global access point for obtaining this unique instance and is used for sharing resources or maintaining a global state throughout the application[37]. Figure5.20 shows the flowchart for implementing the Singleton pattern using PlayerUIHelper.

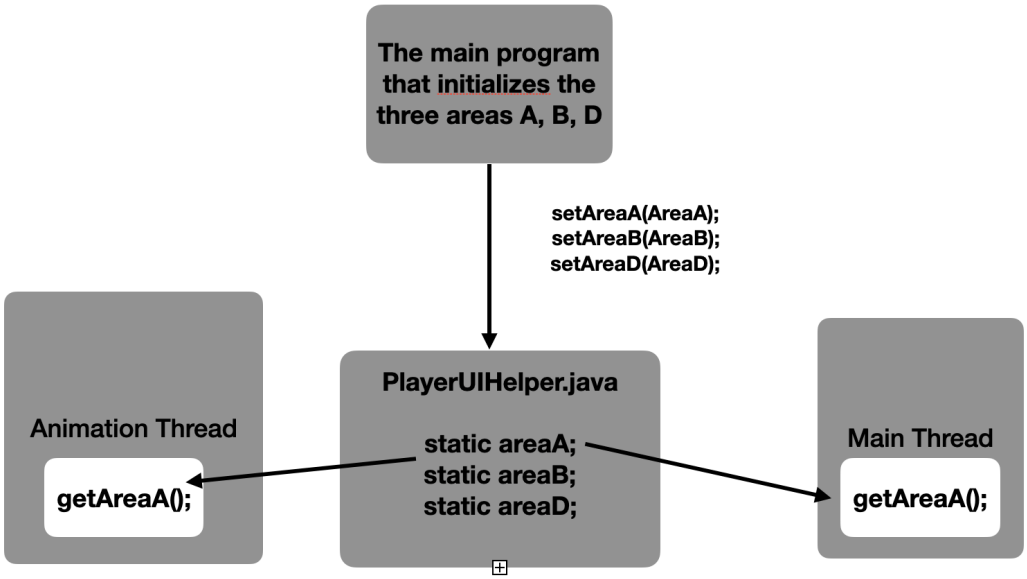


Figure 5.20: The figure shows the singleton pattern for sharing resources between two threads. Using the setAreaA method to set up the static variable in PlayerUIHelper and the getAreaA method to get the variable.

By employing the Singleton pattern and multithreading, the problems of animation stuttering caused by thread blocking and interface scrolling are effectively addressed. Consequently, Area D can smoothly display the animation process of the physical data structure changes. Additionally,

since a separate thread controls the animation drawing logic, pausing all animations on the interface can be achieved by making the thread enter a suspended state using a thread lock. Waking the thread up allows it to continue executing the drawing logic, which resumes the animation playback. Combined with interface button listeners, this enables the implementation of playing and pausing animations on the interface.

Double-buffering

The double-buffering technique is used further to enhance the smoothness of the animations in area D. It achieves this by drawing the upcoming frame content to an additional buffer layer and then using the buffer layer to overlay the entire panel instead directly drawing the content onto the target panel. Figure5.21 illustrates the difference between this technique and direct drawing.

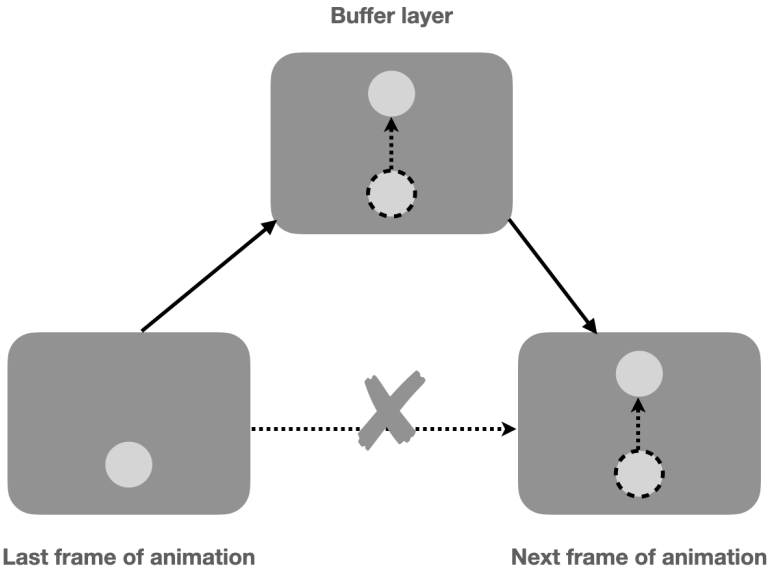


Figure 5.21: The figure shows the double-buffering technique that enhances the smoothness of the animations in the application.

5.1.5 Create Animation

Users can express physical data structures and their changes using graphic resources from the library, highlight related source code, and edit designed logical data structure graphics.

Implementation of Animation Library

The Animation Library contains three packages, each corresponding to the content in areas A, B, and D of the Animation Player. See Figure5.22. The functions in these packages can be used to draw graphics using the Singleton pattern. See Figure5.20. The utility classes in these packages

can obtain the panel objects, the Graphic variables serving as brushes, and hashmaps and arrays used to store the graphic data on the Canvas. Using the three algorithms discussed in subsection 5.1.2 to update the contents of these data structures, the corresponding graphics can be drawn on the panels using the obtained panel objects and pens.

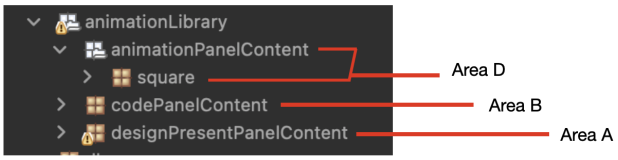


Figure 5.22: The figure shows the package structure from the animation library.

Here is an example of four functions for creating animations using squares in area D in the animation library and highlighting the code in area B. Users can instantiate an object to access the library and then call these functions:

- **drawASquare** This function's parameters will draw a numbered square on Area D.
- **panTO_LEFT** This function will pan the view to the left by a certain distance.
- **panTO_RIGHT** This function will pan the view to the right by a certain distance.
- **heightLightCodeLine** This function will highlight a specific line of code in area B.

  ....

Examples will be provided later to discuss how these functions can be used in an animation profile.

Implementation of Animation Profile

After the user creates a project folder, a file named AnimationController.java will be generated as an Animation Profile. Its contents will be automatically filled in to facilitate the user in making animations.

Firstly, the user must create functions in this file and call functions in the library to create animations. For example, when a teacher wishes to display the source code corresponding to a change, he can call the highlight method from the library to highlight the statement and then use the panTo_Left method to move a square representing an array element. It is worth noting that users can insert animation functions from the library into their actual algorithm code, providing more accurate reference and precise simulation for beginners to help bridge the gap between theory and practice. See Figure5.23 and Figure5.24. Then, the user puts these functions into an array and runs the file. See Figure5.25. The system will load an animation player and traverse and run each function stored in the array in the Animation thread to implement the animation playback. Therefore, the animation playback process runs the user-defined animation functions. Java is an object-oriented language that does not allow functions to be stored in a data structure. However,

functional programming features were added in Java 8, making this approach feasible[38]. See Figure5.25.

Figure 5.23: The figure shows the mixture of visualization methods from the library and source code statements of the bubble sort algorithm in the Animation Profile.

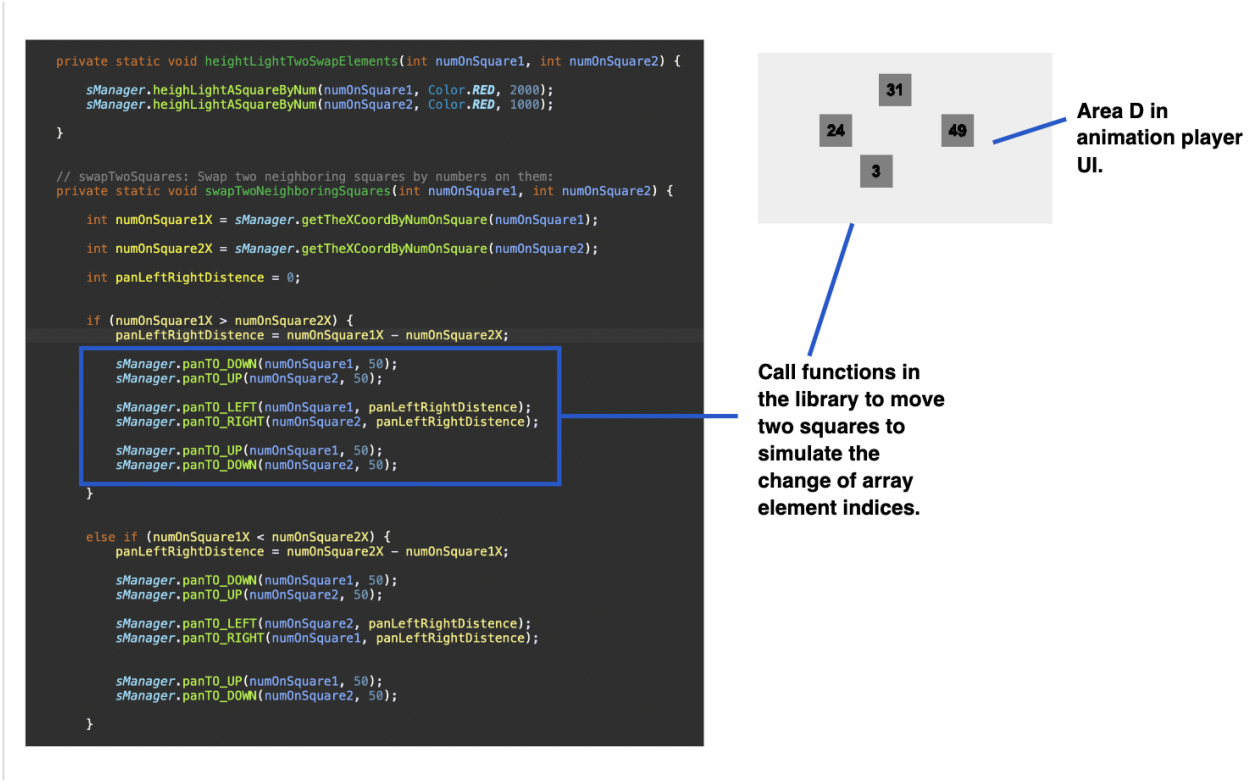
5.1.6 Sharing Function

Users can share their project folders to allow others to view their animations. The graphic data and Animation Profile are stored in the project folder as text and Java files. Java IO writes the code for launching the application in advance when the Animation Profile is generated. Other users who have installed the application can view the animations by running the Java file in the shared project folder.

5.2 Testing

This desktop application will be distributed to users for testing, and users will be asked to complete a questionnaire, the results of which will be discussed in Chapter 6. Chapter 5 will complete the functional testing by comparing it with the user stories table and then demonstrate the automated test results.

5.2.1 Functional Testing




```
private static void heightLightTwoSwapElements(int numOnSquare1, int numOnSquare2) {

    sManager.heighLightASquareByNum(numOnSquare1, Color.RED, 2000);
    sManager.heighLightASquareByNum(numOnSquare2, Color.RED, 1000);

}


// swapTwoSquares: Swap two neighboring squares by numbers on them:
private static void swapTwoNeighboringSquares(int numOnSquare1, int numOnSquare2) {

    int numOnSquare1X = sManager.getTheXCoordByNumOnSquare(numOnSquare1);

    int numOnSquare2X = sManager.getTheXCoordByNumOnSquare(numOnSquare2);

    int panLeftRightDistence = 0;


    if (numOnSquare1X > numOnSquare2X) {
        panLeftRightDistence = numOnSquare1X - numOnSquare2X;

        sManager.panTO_DOWN(numOnSquare1, 50);
        sManager.panTO_UP(numOnSquare2, 50);

        sManager.panTO_LEFT(numOnSquare1, panLeftRightDistence);
        sManager.panTO_RIGHT(numOnSquare2, panLeftRightDistence);

        sManager.panTO_UP(numOnSquare1, 50);
        sManager.panTO_DOWN(numOnSquare2, 50);
    }


    else if (numOnSquare1X < numOnSquare2X) {
        panLeftRightDistence = numOnSquare2X - numOnSquare1X;

        sManager.panTO_DOWN(numOnSquare1, 50);
        sManager.panTO_UP(numOnSquare2, 50);

        sManager.panTO_LEFT(numOnSquare2, panLeftRightDistence);
        sManager.panTO_RIGHT(numOnSquare1, panLeftRightDistence);


        sManager.panTO_UP(numOnSquare1, 50);
        sManager.panTO_DOWN(numOnSquare2, 50);

    }
```

Figure 5.24: The figure shows a user-defined function in the Animation Profile by invoking the library moving squares function.

```
public static ArrayList<Consumer<String>> getAnimationPanel_Animation_STEPs() {
    ArrayList<Consumer<String>> animationSteps = new ArrayList<>();

    animationSteps.add(AnimationController::animation_STEP1);
    animationSteps.add(AnimationController::bubbleSorting);

    return animationSteps;
}
```

Figure 5.25: The figure shows how the user can visualize the animation functions they want to run by putting them in an array. The system automatically generates the getAnimationPanel_Animation_STEPs() function and the array, which does not require any user-defined code.

Table 5.1: Functional Testing

No.	Story	Actual Outcome	MoCoW	Result
1	As a user, I should have a canvas for drawing data structures.	Compiling and running Run.java will open the Canvas.	Must have	Pass
2	As a user, I should be able to use at least one shape to draw "nodes" for data structures.	Clicking the "Circle" button on Canvas will allow users to use circular shapes to draw nodes in the data structure.	Must have	Pass
3	As a user, I should be able to choose different types of connecting lines for drawing "edges" for data structures.	Pressing the "A" or "S" key on the keyboard and then clicking on the target node with the mouse will allow the user to construct edges for the data structure using black lines or blue directional lines.	Must have	Pass
4	As a user, I should be able to define the locations of these shapes.	Double-clicking on Canvas allows users to place graphics anywhere, and position layouts can be achieved by calling functions from the Animation Library in the Animation Profile.	Must have	Pass
5	As a user, I should be able to adjust the size of these shapes freely.	N/A	Could have	Not Implemented

Continued on next page

Table 5.1 – *Continued from previous page*

No.	Story	Actual Outcome	MoSCoW	Result
6	As a user, I should be able to add and delete drawn graphics on the Canvas.	Double-click with the mouse to draw shapes on the Canvas interface. Select a shape and click the "Delete" button to remove the shape and its connections.	Must have	Pass
7	As a user, I should be able to label the components of the data structure graphics with numbers on the canvas.	The shapes drawn on the Canvas interface will be automatically labelled, and the Animation Library provides functions to customize the numerical labels on the shapes.	Must have	Pass
8	As a user, I should be able to add comments to the data structures I have drawn on the Canvas.	Clicking on the "Add Annotations" button on the Canvas interface after selecting a shape will allow users to add annotations.	Should have	Pass
9	As a user, I should be able to move the graphics on the Canvas freely.	N/A	Could have	Not Implemented

Continued on next page

Table 5.1 – *Continued from previous page*

No.	Story	Actual Outcome	MoSCoW	Result
10	As a user, I should have a player interface for watching animations or previewing the effects of my work.	After completing the drawing and saving work, the Animation Player will automatically pop up. The Animation Player can also be opened by allowing the Animation Profile.	Must have	Pass
11	As a user, I should be able to see the data structure graphics I have drawn on the Canvas with its annotations and added code from the Canvas in the Player interface.	After launching the Animation Player, the user-designed graphical data structure will be displayed in the largest area on the left-hand side.	Must have	Pass
12	As a user, I should be able to play and pause the animation in the player interface.	Clicking the Start/Pause button on the Animation Player can pause the animation playback.	Should have	Pass
13	I should be able to view the annotations of graphics drawn from the canvas in real-time during the animation playback to help me understand the content.	On the Animation Player user interface, users can move the mouse to preview the annotations of the graph in real time. See Figure5.9.	Could have	Pass
14	As a user, I should be able to forward and backward the animation in the player interface.	N/A	Could have	Not Implemented

Continued on next page

Table 5.1 – *Continued from previous page*

No.	Story	Actual Outcome	MoSCoW	Result
15	As a user, I should be able to watch any step of the animation process by providing the step number.	N/A	Could have	Not Implemented
16	As a user, I should be able to see the corresponding graphics changes during the animation process.	Users can freely define animations on the Animation Player, and viewers can see the corresponding changes.	Must have	Pass
17	As a user, I should be able to see the corresponding code highlighting during the animation process.	Users can freely define animations on the Animation Player, and viewers can see the corresponding changes.	Must have	Pass
18	As a user, I should be able to use one graphic at least to present the physical data structures in the programming language to define the steps of the animation.	The Animation Library defines a Square class that provides motion and drawing functions related to squares.	Must have	Pass
19	As a user, I should be able to display my operations on these physical data structures, such as adding, deleting, modifying, and searching, as well as the dynamic processes of these operations, such as swapping the positions of two adjacent arrays elements.	The Animation Library defines functions for controlling the single-directional motion of shapes and adding and deleting shapes. Users can combine these functions to create their automation methods.	Must have	Pass

Continued on next page

Table 5.1 – *Continued from previous page*

No.	Story	Actual Outcome	MoSCoW	Result
20	As a user, I should also be able to control the corresponding code highlighting, such as highlighting a statement for a while to define the steps of the animation.	The Animation Library provides a function to highlight the insertion system's source code, which the user can call in the Animation Profile. See Figure5.9	Must have	Pass
21	As a user, I should be able to control the changes made to the graphics drawn on the Canvas user interface, such as deletion, insertion, or highlighting, to define the animation steps.	Calling the Animation Library's delete, insert and highlight functions can achieve this.	Must have	Pass
22	As a user, I should be able to use this system for learning anytime without being restricted by the network.	The application is a Java desktop application, and all data is saved locally, so it does not require network support.	Should have	Pass
23	As a user, I should be able to have a large enough player panel to watch more dynamic processes simultaneously.	The area on the Animation Player used to display graphic information is scrollable, allowing users to see a larger area.	Should have	Pass
24	As a user, I should be able to save my work locally.	The system will automatically generate a project folder for saving in the "src/user/" directory.	Should have	Pass

Continued on next page

Table 5.1 – *Continued from previous page*

No.	Story	Actual Outcome	MoSCoW	Result
25	As a user, I can update my saved work anytime.	Users can edit the Animation Profile file at any time to control the dynamic changes of the graphics on the Animation Player. Still, they cannot modify it again through Canvas.	Could have	Partial
26	As a user, I should be able to share my work.	The user's works are saved in a folder named by the user, and sharing can be achieved by sharing this folder.	Could have	Pass
27	As a user, I should be able to customize new graphics and add them to the system's graphics library.	N/A	Won't have	Not Implemented

Overall: 21/27

- **"Must Have" requirements completed:** 14/14
- **"Should Have" requirements completed:** 5/5
- **"Could Have" requirements completed:** 2/7
- **"Won't Have" requirements completed:** 0/1

5.2.2 User Testing

In addition to functional testing, user testing was also conducted, where users were allowed to use the application and provide feedback through a questionnaire(See Appendix E) on important criteria. The results are analyzed and presented in detail in Chapter 6.

Chapter 6

Results and discussion

The chapter presents the findings and results of user testing and critically discusses the goals achieved and possible further application implementations.

6.1 Findings

Participants willing to test the application will receive a user installation guide (See Appendix D) and join online training led by the project developer. The testing process involves watching a previous participant's algorithm animation, creating their own, and passing it to the next participant. Finally, they complete a questionnaire and submit their animation code. Four TUOS computer science undergraduates participated, with two creating animations.

UI design

The survey results regarding the UI's appearance were mixed. Half of the participants were satisfied, while the other half felt neutral or dissatisfied. See Figure6.1. Those who were unhappy thought the application should have more style or colour. See Figure6.2. This outcome may be due to using Java Swing instead of a more modern framework such as JavaFX for front-end development.

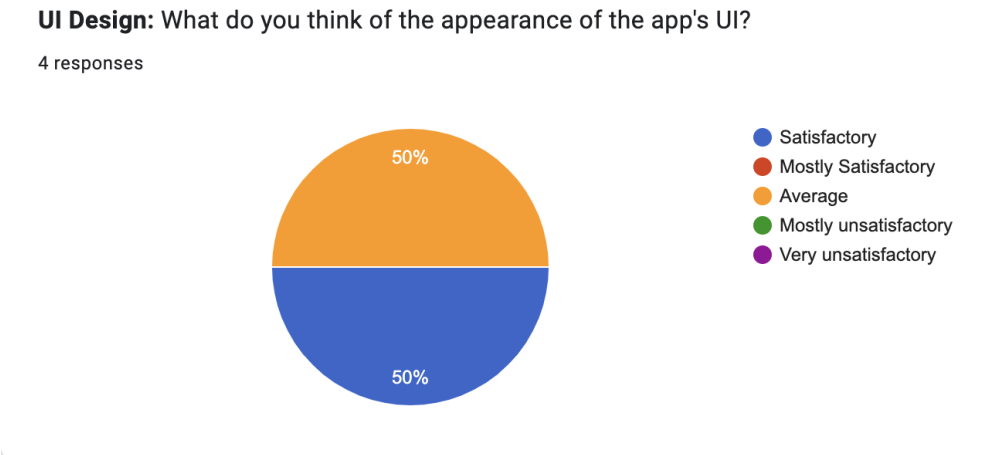


Figure 6.1: The figure shows the survey results regarding the UI appearance.

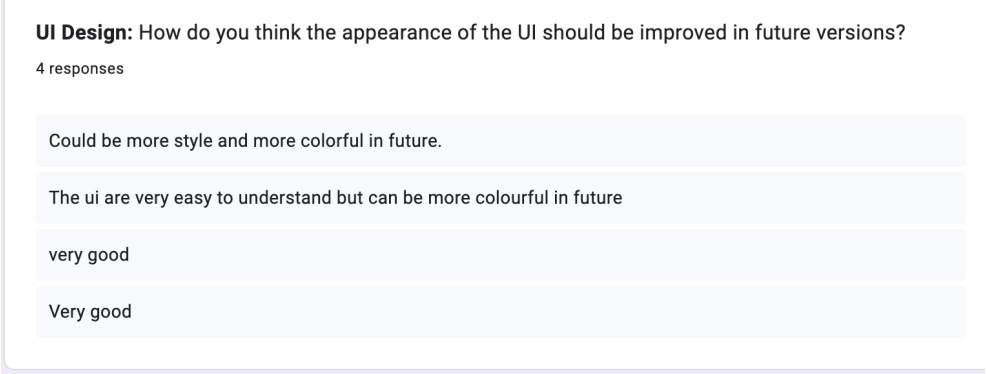


Figure 6.2: The figure shows the participants' comments on the UI appearance.

Functionality practicality

The application's practicality was assessed through participants' satisfaction with its performance and installation process. As a Java-based desktop application, it was expected to be fast, and all participants were satisfied with its speed. Moreover, since most TUOS computer science undergraduates have studied Java programming, all participants found the installation process straightforward. See Figure6.3.

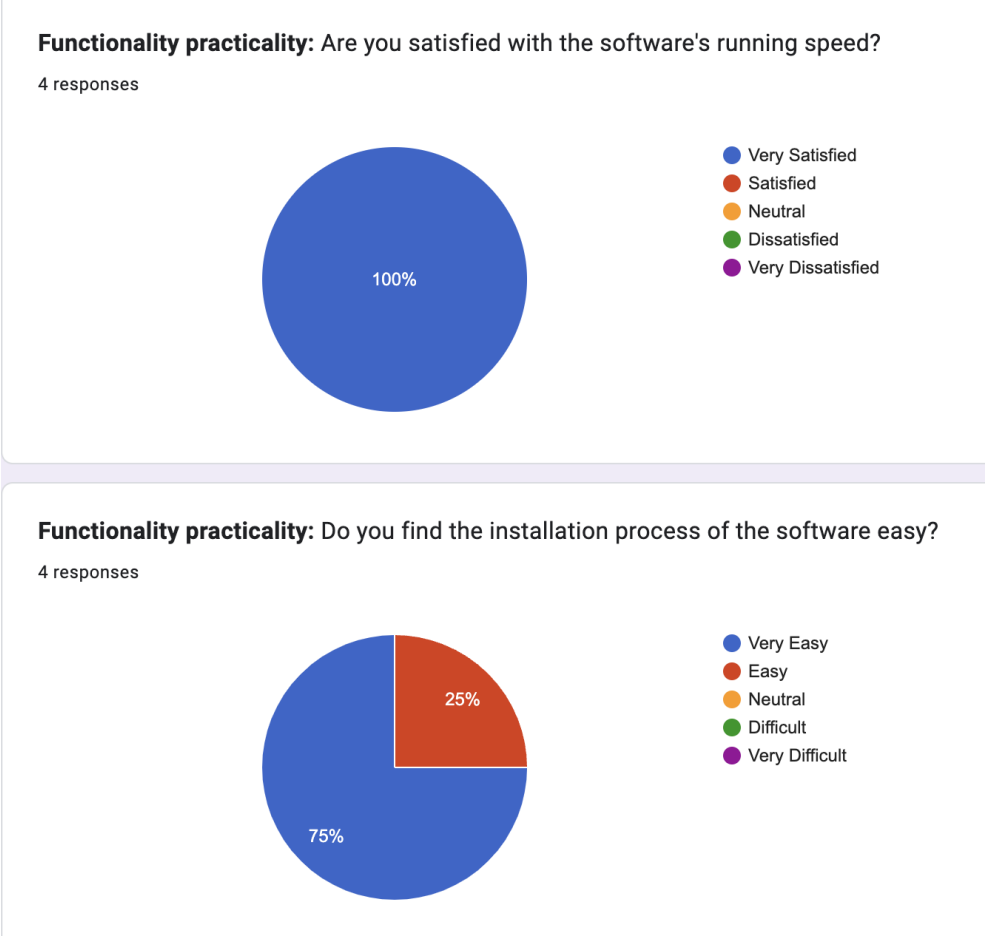


Figure 6.3: The figure shows the survey results regarding the UI application practicality.

Ease of use

The application's usability will be evaluated by the ease of creating animations with the animation profile and Canvas and the clarity of the Animation Player. If a participant fails to create an animation, the prior participant's animation will be shared with the current participant or adjusted testing order, ensuring questionnaire completion.

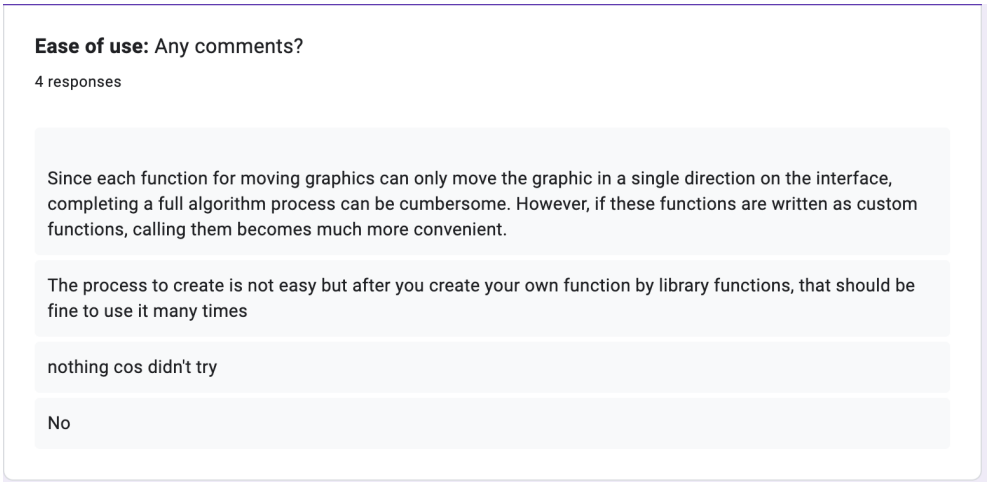


Figure 6.4: The figure shows the comments regarding the ease of creating animations with the support of the Animation Library in the animation profile.

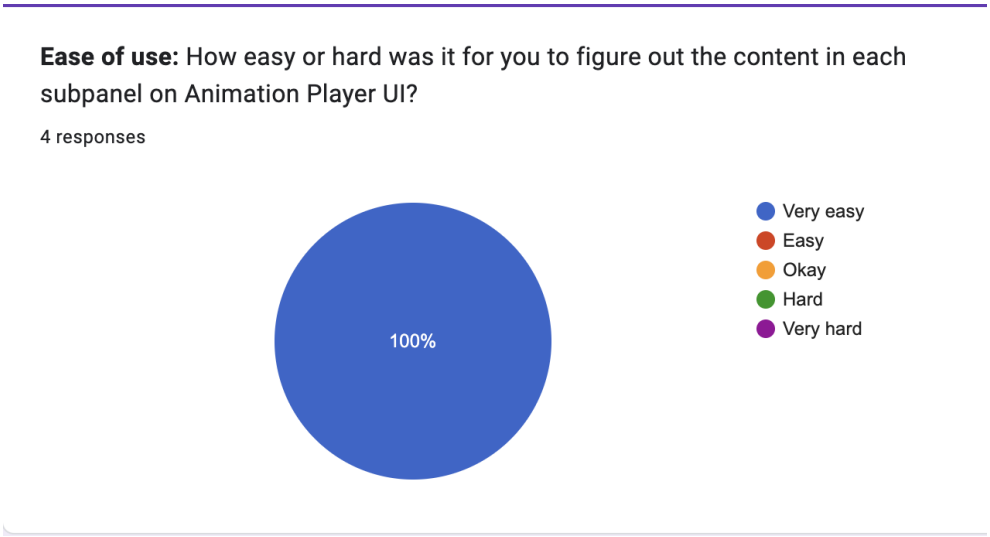


Figure 6.5: The figure shows the survey results regarding the clarity of the Animation Player.

Finally, all participants understood the Animation Player's content and found the Canvas user interface user-friendly. See Figure6.5 and Figure6.6. However, two participants who successfully created bubble sort and array-based binary tree animations initially considered the process tedious. See Figure6.4. However, given the reusability of functions, such as swapping element indices in most sorting algorithms, the process is a one-time effort. Figure6.7 and Figure6.8 show screenshots of animation created by participants.

Application feature

Half the participants considered the drawable graphic data structure their favourite application feature. In contrast, the other half found the multi-content display on the Animation Player to be their favourite application feature. See Figure6.9.

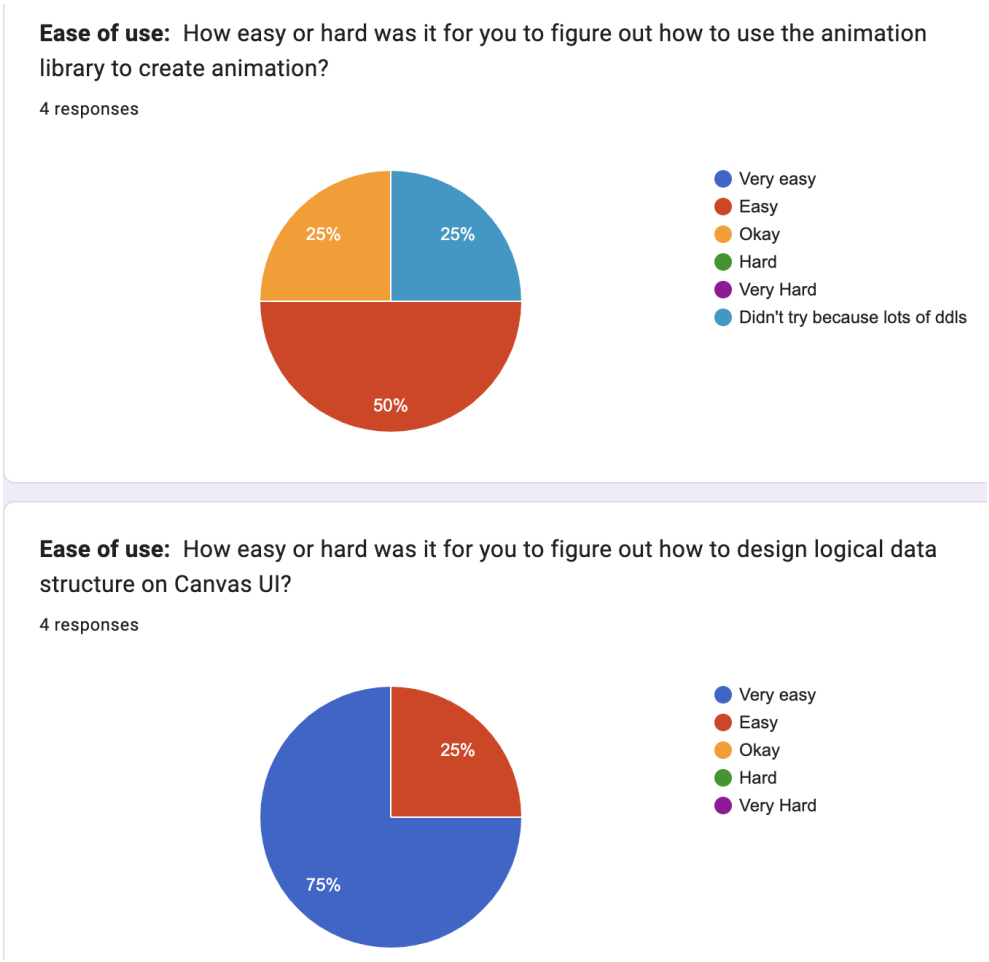
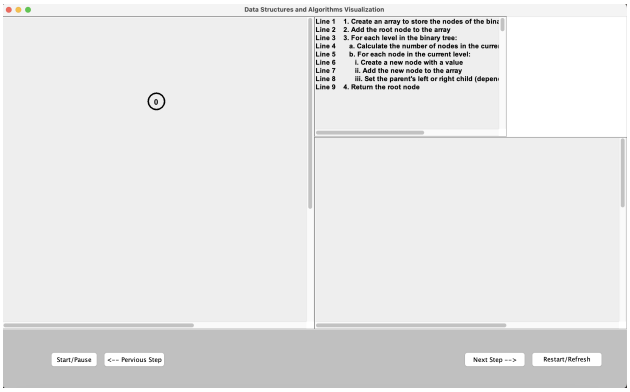
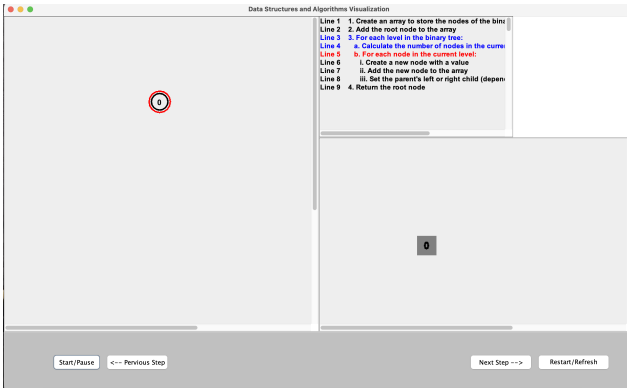


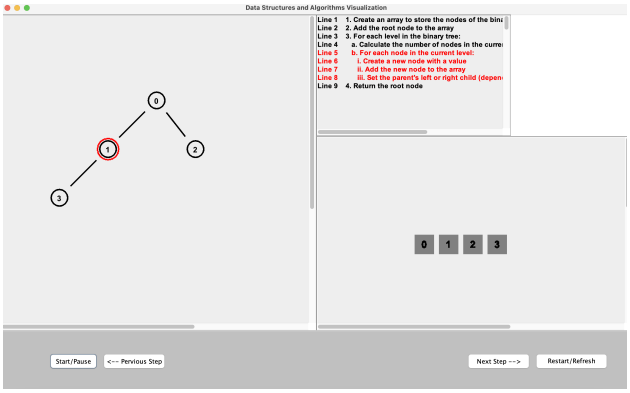
Figure 6.6: The figure shows the survey results regarding the ease of creating animations with the Animation Profile and Canvas.



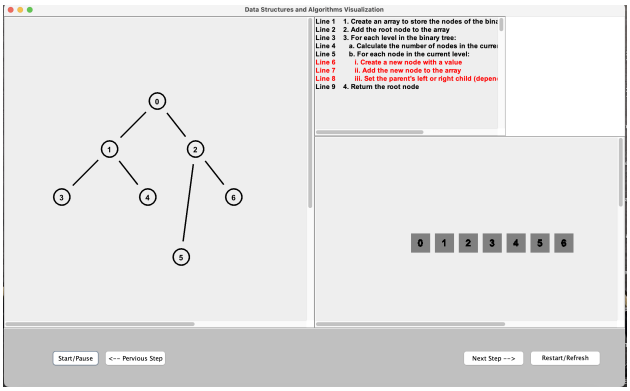
(a)



(b)



(c)

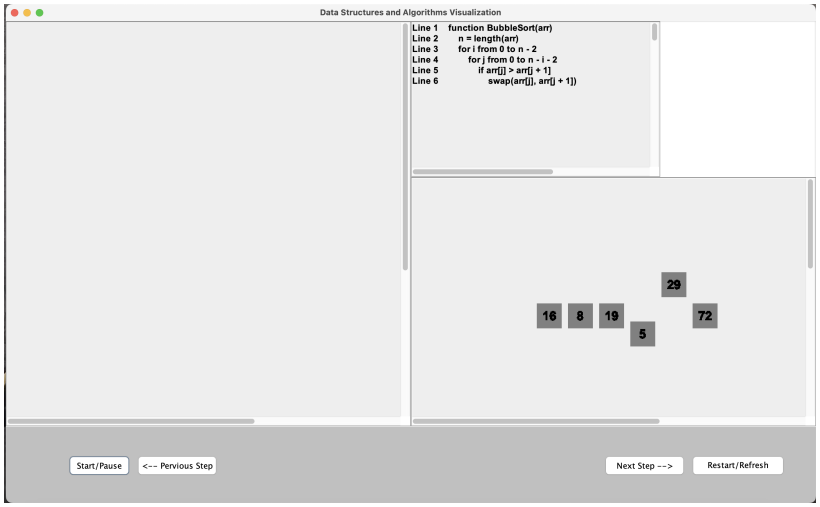


(d)

Figure 6.7: These are screenshots of the animation process of constructing a binary tree using an array on Animation Player, taken by one of the testing participants.

(a)

(b)



(c)

Figure 6.8: These are screenshots of the animation process of the bubble sorting algorithm on Animation Player, taken by one of the testing participants.

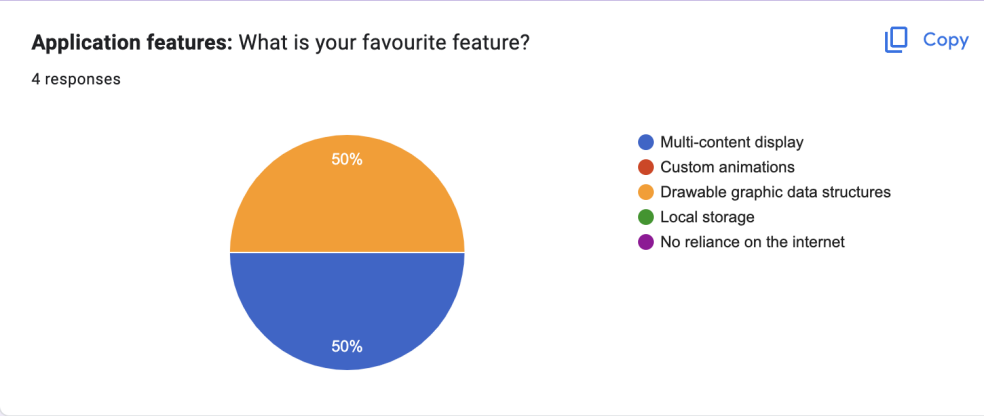


Figure 6.9: The figure shows survey results regarding the favourite application feature.

Other Findings

Some participants reported that they appreciated the ability to resize the canvas UI to fit their limited screen space. See Figure6.10 This was made possible by the efficiency of HashMap, which allows for fast traversal and redrawing of graphical elements on the interface. As a result, users can quickly resize the UI without experiencing significant lag or delay in redrawing the graphics.

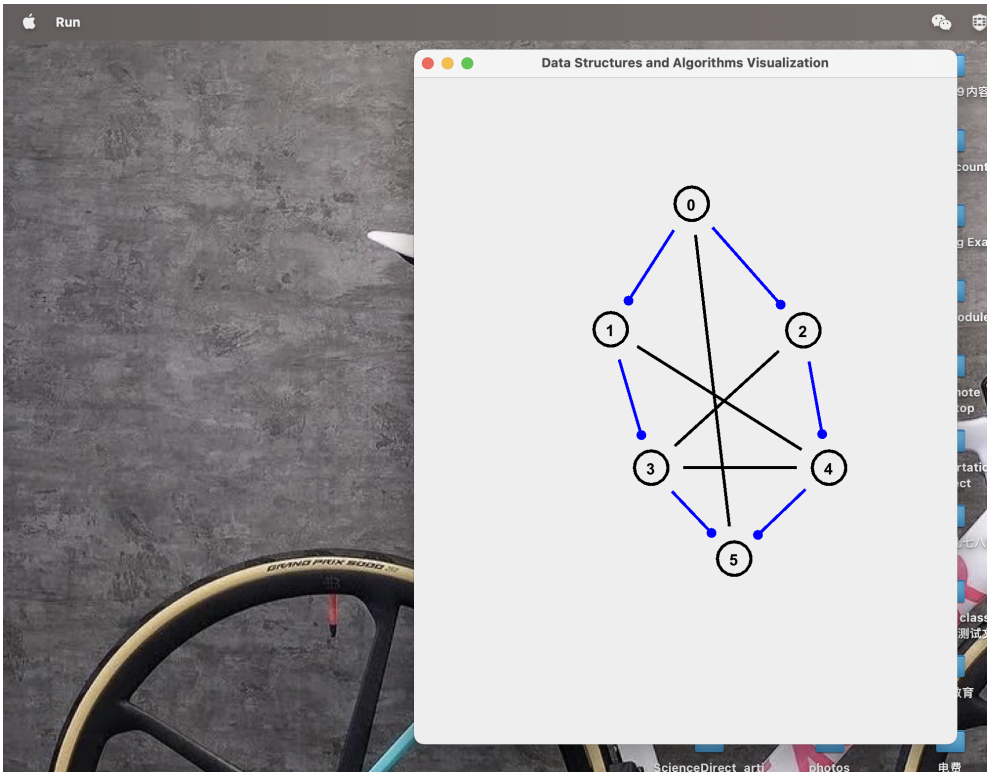


Figure 6.10: The figure shows a screenshot of a smaller-size canvas UI on the screen.

Findings Summary

User testing results revealed a challenge with the app: using programming to create animations can be cumbersome. This stems from a requirement paradox since the app targets teachers and students. Students prefer seeing algorithmic statements closely related to the animation, so the Animation Library methods move graphics in one direction at a time, allowing for accurate algorithm simulation. However, this complicates creating reusable animation functions, and users must express simple swapping processes using multiple methods.

6.2 Goals Achieved

- Reviewed literature on the common algorithm and data structure visualization systems and examined four innovative systems.
- Analyzed the difficulties encountered by 51 students in learning data structures and algorithms by questionnaire survey.
- Analyzed the reasons behind the questionnaire survey results.
- Identified and prioritized the user requirements and application features based on the analysis results.
- Designed the UI mockups and backend by Moqups.

- Developed the desktop application in Java.
- All "Must Have" and "Should Have" requirements have been implemented.
- Evaluated the project by four students' feedback and functional testing.

6.3 Further Work

This section will describe the functionalities that should be added to the app in the future, obtained from unmet requirements, user feedback, and difficulties encountered during testing.

- **Unit Testing** Unit testing is effective testing to reduce software risk. Although this application's backend uses a more decoupled MVVM architecture, the presence of components like the Animation Library and Animation Profile increases coupling, making effective unit testing difficult. The application requires a more decoupled backend controller to accommodate efficient unit testing.
- **Custom Graphics Library** The application should provide a graphics library and an interface for users to customize additional graphics to represent various data structures.
- **Community-oriented** The application should create a small community within the university campus based on its sharing functionality. Users can share their Animation Profiles and contribute new animation methods developed using the Animation Library. Other community members could call these methods directly, effectively reducing the tedious work of creating animations.
- **Buffering animation process** This feature is the foundation for implementing free playback controls, such as forward and backward playback.
- **Moving graphics freely through the interface** Users can move graphics forming data structures on the Canvas using their mouse. This will enhance the application's user engagement, as it can create more complex data structures.

Chapter 7

Conclusion

Knowledge of algorithms and data structures is crucial for computer science undergraduate students. University modules on algorithms and data structures often focus more on theoretical knowledge than programming implementation. Additionally, teachers need to create many lecture materials to demonstrate the algorithm's execution process composed of static graphic changes. As a result, students may struggle to apply the learned theoretical knowledge to programming practice, and teachers' lectures may not be as efficient. The Java desktop application in this project aims to provide teachers with a convenient tool by combining algorithm source code and animation generation instructions to create dynamic processes that simultaneously display various related information. Students can also bridge the gap between practice and theory by examining the code in the Animation Profile.

This report presents the project's research process: Firstly, various algorithm visualization schemes were evaluated, and a survey questionnaire was used to understand the difficulties faced by beginners. Then, based on the survey results and literature analysis, the application's use scenarios, development languages, and user requirements were determined, and the user interface appearance, backend architecture, and data storage methods were designed. Subsequently, the functionalities related to the requirements were implemented using Java.

The application was tested by users who provided feedback. Most participants found the information displayed on the interface simple and easy to understand and attempted to create animations. The application also employed functional testing to ensure the implementation of prioritized features and identify future work.

In conclusion, the project has successfully implemented most of its functionalities and achieved its goals. As anticipated, it can facilitate more efficient teaching for instructors and help beginners better learn data structures and algorithm knowledge.

Bibliography

[1] N. Wirth, *Algorithms & data structures*. Prentice-Hall, Inc., 1985.

[2] "Robomaster s1," 2023.

[3] D. Galles, "Visualization of algorithms and data structures," 2011.

[4] J. Park, "Algorithm visualizer." https://github.com/algorithm-visualizer/algorithm-visualizer, 2022.

[5] R. A. Nathasya, O. Karnalim, and M. Ayub, "Integrating program and algorithm visualisation for learning data structure implementation," *Egyptian Informatics Journal*, vol. 20, no. 3, pp. 193–204, 2019.

[6] C. A. Shaffer, L. S. Heath, and J. Yang, "Using the swan data structure visualization system for computer science education," *ACM SIGCSE Bulletin*, vol. 28, no. 1, pp. 140–144, 1996.

[7] T. Chen and T. Sobh, "A tool for data structure visualization and user-defined algorithm animation," in *31st Annual Frontiers in Education Conference. Impact on Engineering and Science Education. Conference Proceedings (Cat. No. 01CH37193)*, vol. 1, pp. TID–2, IEEE, 2001.

[8] A. S. Erkan, T. VanSlyke, and T. M. Scaffidi, "Data structure visualization with latex and prefuse," *ACM SIGCSE Bulletin*, vol. 39, no. 3, pp. 301–305, 2007.

[9] UniversityOfSheffield, "This is detailed information of com1005 @ONLINE," Sept. 2022.

[10] T. U. o. S. Department of Computer Science, "Com1009: Introduction to algorithms and data structures." https://www.dcs.shef.ac.uk/intranet/teaching/public/modules/level1/com1009.html, Accessed: April 30, 2023.

[11] G.-C. Roman, K. C. Cox, C. D. Wilcox, and J. Y. Plun, "Pavane: a system for declarative visualization of concurrent computations," *Journal of Visual Languages & Computing*, vol. 3, no. 2, pp. 161–193, 1992.

[12] C. D. Hundhausen, S. A. Douglas, and J. T. Stasko, "A meta-study of algorithm visualization effectiveness," *Journal of Visual Languages & Computing*, vol. 13, no. 3, pp. 259–290, 2002.

[13] D. E. Knuth, *Art of computer programming, volume 2: Seminumerical algorithms*. Addison-Wesley Professional, 2014.

[14] G. Murphy, M. Kersten, and L. Findlater, "How are java software developers using the eclipse ide?," *IEEE Software*, vol. 23, no. 4, pp. 76–83, 2006.

[15] M. A. Kolosovskiy, "Data structure for representing a graph: combination of linked list and hash table," *arXiv preprint arXiv:0908.3089*, 2009.

[16] M. Bikanga Ada, "Teaching algorithms and data structures: A tale of two approaches," 2020.

[17] D. Thalmann and N. Magnenat-Thalmann, "Design and implementation of abstract graphical data types," in *COMPSAC 79. Proceedings. Computer Software and The IEEE Computer Society's Third International Applications Conference, 1979.*, pp. 519–524, 1979.

[18] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press, 2009.

[19] A. Biessek, *Flutter for Beginners: An introductory guide to building cross-platform mobile applications with Flutter and Dart 2*. Packt Publishing Ltd, 2019.

[20] M. Loy, R. Eckstein, D. Wood, J. Elliott, and B. Cole, *Java swing*. " O'Reilly Media, Inc.", 2002.

[21] S. Dimitrijević, J. Jovanović, and V. Devedžić, "A comparative study of software tools for user story management," *Information and Software Technology*, vol. 57, pp. 352–368, 2015.

[22] R. Popli, N. Chauhan, and H. Sharma, "Prioritising user stories in agile environment," in *2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*, pp. 515–519, IEEE, 2014.

[23] V. Houghton, "Moqups: an easy way to create and share mockup designs online without having to know code," 2014.

[24] D. Murray, "Interaction design," *University of London International Programmes. United Kingdom*, 2010.

[25] D. M. Selfa, M. Carrillo, and M. D. R. Boone, "A database and web application based on mvc architecture," in *16th International Conference on Electronics, Communications and Computers (CONIELECOMP'06)*, pp. 48–48, IEEE, 2006.

[26] N. Ighodaro, "How laravel implements mvc and how to use it effectively," 2018.

[27] A. Leff and J. Rayfield, "Web-application development using the model/view/controller design pattern," in *Proceedings Fifth IEEE International Enterprise Distributed Object Computing Conference*, pp. 118–127, 2001.

[28] Y. H. Ding, C. H. Liu, and Y. X. Tang, "Mvc pattern based on java," in *Applied Mechanics and Materials*, vol. 198, pp. 537–541, Trans Tech Publ, 2012.

[29] C. Anderson, "The model-view-viewmodel (mvvm) design pattern," in *Pro Business Applications with Silverlight 5*, pp. 461–499, Springer, 2012.

[30] "java.io — android developers." https://developer.android.com/reference/java/io/package-summary.

[31] "Java se 7 and jdk 7 api documentation: javax.swing.lookandfeel." https://docs.oracle.com/javase/7/docs/api/javax/swing/LookAndFeel.html.

[32] K. Arnold, J. Gosling, and D. Holmes, *The Java programming language*. Addison Wesley Professional, 2005.

[33] J. T. Streib, T. Soma, J. T. Streib, and T. Soma, "Hashing," *Guide to Data Structures: A Concise Introduction Using Java*, pp. 339–362, 2017.

[34] Oracle, "Javacompiler (java platform se 8)." https://docs.oracle.com/javase/8/docs/api/javax/tools/JavaCompiler.html, 2014.

[35] L. Chen, J. Wang, M. Xu, and Z. Zeng, "Reengineering of java legacy system based on aspect-oriented programming," in *2010 Second International Workshop on Education Technology and Computer Science*, vol. 3, pp. 220–223, IEEE, 2010.

[36] Oracle, "Interface runnable." https://docs.oracle.com/javase/8/docs/api/java/lang/Runnable.html, 2023.

[37] J. W. Cooper, "Java design patterns: a tutorial," 2000.

[38] V. Subramaniam, "Functional programming in java: harnessing the power of java 8 lambda expressions," *Functional Programming in Java*, pp. 1–196, 2014.

Appendices

Appendix A

Ethic Application

Application 049844

Section A: Applicant details

Date application started:
Wed 28 September 2022 at 23:45

First name:
Hongyu

Last name:
Wang

Email:
hwang135@sheffield.ac.uk

Programme name:
Dissertation Project 2022-23

Module name:
COM3610 Dissertation Project

Last updated:
24/10/2022

Department:
Computer Science

Applying as:
Undergraduate / Postgraduate taught

Research project title:
Data Structure and Algorithm Visualization Tool for Beginners in Department of CS

Has your research project undergone academic review, in accordance with the appropriate process?
No

Similar applications:
- not entered -

Section B: Basic information

Supervisor

Name	Email
Mike Stannett	m.stannett@sheffield.ac.uk

Proposed project duration

Start date (of data collection):
Tue 11 October 2022

Anticipated end date (of project)
Wed 10 May 2023

3: Project code (where applicable)

Project externally funded?
No

Project code
- not entered -

Suitability

Takes place outside UK?
No

Involves NHS?
No

Health and/or social care human-interventional study?
No

ESRC funded?
No

Likely to lead to publication in a peer-reviewed journal?
No

Led by another UK institution?
No

Involves human tissue?
No

Clinical trial or a medical device study?
No

Involves social care services provided by a local authority?
No

Is social care research requiring review via the University Research Ethics Procedure
No

Involves adults who lack the capacity to consent?
No

Involves research on groups that are on the Home Office list of 'Proscribed terrorist groups or organisations?
No

Indicators of risk

Involves potentially vulnerable participants?
No
Involves potentially highly sensitive topics?
No

Section C: Summary of research

1. Aims & Objectives

Aims and Objectives:

It is well known that data structures plus algorithms equal programs, so they are crucial for computer science learners, however, learning algorithms and data structures can be challenging for beginners.

Develop a project using an object-oriented programming language designed to better assist first-year students in learning data structures and algorithms. The computer science department at the University of Sheffield is made up of international students, some of whom may not have a superior computer background in high school. This means that they may find it difficult when faced with the abstract algorithms and data structures in COM1005 and COM1009. Therefore, developing a software project that can visualize the working process of common algorithms and data structures will significantly reduce the cost of learning and also, users can set some assignments with the form of quiz to test their performance, so the software may allow various users to access it (eg. teacher and student). This project will firstly use a questionnaire to collect and research the difficulties that people face when learning algorithms and data structures, then based on the survey results, the form of the software(such as whether it should be an application or based on Web) will be considered.

2. Methodology

In this project, the morally sensitive part is the need for a questionnaire, which may be disseminated through social media or lectures in the form of a QR code or a website. It will contain 10 to 14 questions, which will be divided according to the needs of the project. The questionnaire is dynamic which means paticipants will answer questions depends on their subjects and there are three types of questions:

1. About the subjects and degree studied by the participant.
2. Difficulties experienced in learning the relevant knowledge.
3. The helpful learning tools tried during the learning process.

Among them, for the latter two types of questions, multiple choices will be provided. In addition, the survey will also describe three or four specific scenarios in the second type of question, providing single-choice options from "Strongly Agree" to "Strongly Disagree."

Specific examples of questionnaires will be provided in Section F with supporting documentation.

3. Personal Safety

Have you completed your departmental risk assessment procedures, if appropriate?

Yes

Raises personal safety issues?

No

No

Section D: About the participants

1. Potential Participants

Considering that the purpose of this project is to develop learning tools for beginners in the Department of Computer Science of the University of Sheffield, the students majoring in computer science at the University of Sheffield must be potential participants.

In addition, data structures and algorithms are one of the research directions of this project, relavent knowledge is also taught in many STEM subjects, so the opinions of mostly STEM students in UK universities are informative, so potential participants in this survey should be all students studying STEM in the UK (or students with relevant experience).

2. Recruiting Potential Participants

Questionnaires may be distributed to potential participants via email, social media, and QR codes in lectures. The information obtained by the participants in the questionnaire is divided into three types according to the project：
1. Levels and majors.
2. Difficulties encountered in learning algorithms and data structures.
3. Learning tools tried during the learning process in this area.

Participants can express their interest in this by clicking on the link in the email or by entering the questionnaire via a QR code and choosing to submit.

2.1. Advertising methods

Will the study be advertised using the volunteer lists for staff or students maintained by IT Services? No

- not entered -

3. Consent

Will informed consent be obtained from the participants? (i.e. the proposed process) Yes

Users can enter the questionnaire in different ways (eg, QR code or link), answering the questions and clicking the submit button below the questionnaire will default to consent.

4. Payment

Will financial/in kind payments be offered to participants? No

5. Potential Harm to Participants

What is the potential for physical and/or psychological harm/distress to the participants?

We do not envisage any circumstances under which particpants will be exposed to potental harm.

How will this be managed to ensure appropriate protection and well-being of the participants?

We do not envisage any circumstances under which particpants will be exposed to potental harm.

6. Potential harm to others who may be affected by the research activities

Which other people, if any, may be affected by the research activities, beyond the participants and the research team?

No other people beyond the participants and the research team will be affected by the research activities.

What is the potential for harm to these people?

No other people beyond the participants and the research team will be affected by the research activities.

How will this be managed to ensure appropriate safeguarding of these people?

No other people beyond the participants and the research team will be affected by the research activities.

7. Reporting of safeguarding concerns or incidents

What arrangements will be in place for participants, and any other people external to the University who are involved in, or affected by, the research, to enable reporting of incidents or concerns?

Each question in the questionnaire will be a separate section, therefore, at the top of each section there will be a link to an information sheet, and participants can use another link from the sheet to a google form in order to report.

Who will be the Designated Safeguarding Contact(s)?

The person in charge of the project who is Hongyu Wang will be the Designated Safeguarding Contact, his academic Email address (hwang135@sheffield.ac.uk) will be provided on the top of each question section and the information sheet.

How will reported incidents or concerns be handled and escalated?

Participants can directly sent email to hwang135@sheffield.ac.uk or use the google form to report their concern.

The concerns reported by the participants will be classified and dealt with directly by Hongyu Wang (The person in charge of the project.) , and the results will be notified through the participants' email addresses (by default, the participants' email addresses will be collected by the questionnaire).

Section E: About the data

1. Data Processing

Will you be processing (i.e. collecting, recording, storing, or otherwise using) personal data as part of this project? (Personal data is any information relating to an identified or identifiable living person).
Yes

Which organisation(s) will act as Data Controller?

University of Sheffield only

2. Legal basis for processing of personal data

The University considers that for the vast majority of research, 'a task in the public interest' (6(1)(e)) will be the most appropriate legal basis. If, following discussion with the UREC, you wish to use an alternative legal basis, please provide details of the legal basis, and the reasons for applying it, below:

- not entered -

Will you be processing (i.e. collecting, recording, storing, or otherwise using) 'Special Category' personal data?
No

3. Data Confidentiality

What measures will be put in place to ensure confidentiality of personal data, where appropriate?

All personal information collected by the questionnaire created by university google service (require two-factor authentication by the University to login) and will be stored or processed in a password/fingerprint-encrypted personal computer.

4. Data Storage and Security

In general terms, who will have access to the data generated at each stage of the research, and in what form

The project is led by a junior student, and a supervisor from the college will provide guidance and give a grade for the project. In addition, there will be a second marker to give another grade for this project in order to ensure the fairness of this project. Therefore, these three people need to refer to the data and have access to the data.

Some personal data will be generated in this project so those data need to be kept secret to people who are not allowed to access to protect participants' privacy. Therefore, any people who has the access to project data need to be identifiable.

What steps will be taken to ensure the security of data processed during the project, including any identifiable personal data, other than those already described earlier in this form?

According to the guidance of uk's ICO(Information Comments Office), people's views or opinions were classified as personal data. so the data collected in this project are belong to personal data.

The questionnaire will collect people's opinions and their academic institutions information, will not collect their email addesses and name or some much identifiable information, so the data in this project will be anonymous and not identifiable.

About steps will be taken to ensure the security of data processed during the project: The questionnaire was first generated using the University of Sheffield's Google service, so all editing, saving, publishing and viewing of responses to this form will require two-factor authentication by the University and login to university google account.

Subsequently, the collected information will be automatically organized by Google form, and more data processing and analysis processes will be performed in the Google account authenticated by the university or exported locally for processing. The local data will be stored in a password-passed and fingerprint encrypted Mac.

Will all identifiable personal data be destroyed once the project has ended?
Yes

Please outline when this will take place (this should take into account regulatory and funder requirements).

Considering the final score of the dessertation project may marked as "Fail" and the project may have to be restarted, so the data produced by the questionnaire will be destroyed 4 months after the first final score of the project released in order to support the resit project.

Section F: Supporting documentation

Information & Consent

Participant information sheets relevant to project?
Yes

Document 1112538 (Version 7) — All versions

Consent forms relevant to project?
Yes

Document 1112539 (Version 3) — All versions

Additional Documentation

Document 1112446 (Version 1) — All versions
This document provide the description and initial anlysis of this project

Document 1112540 (Version 3) — All versions
Example of google questionnaire

External Documentation

https://docs.google.com/forms/d/e/1FAIpQLSefX8OhBJaTVOTIYXoAMy4ZlW-XrBLZj_naZy0oWPEPfrIBJQ/viewform?usp=sf_link

This url will show how the real questionnaire works.

Section G: Declaration

Signed by:
Hongyu Wang
Date signed:
Thu 20 October 2022 at 20:40

Offical notes

- not entered -

Appendix B

Ethics Approval Letter

Hongyu Wang
Registration number: 200172350
Computer Science
Programme: Dissertation Project 2022-23

Dear Hongyu

PROJECT TITLE: Data Structure and Algorithm Visualization Tool for Beginners in Department of CS
APPLICATION: Reference Number 049844

On behalf of the University ethics reviewers who reviewed your project, I am pleased to inform you that on 24/10/2022 the above-named project was **approved** on ethics grounds, on the basis that you will adhere to the following documentation that you submitted for ethics review:

- University research ethics application form 049844 (form submission date: 20/10/2022); (expected project end date: 10/05/2023).
- Participant information sheet 1112538 version 7 (20/10/2022).
- Participant consent form 1112539 version 3 (19/10/2022).

If during the course of the project you need to deviate significantly from the above-approved documentation please inform me since written approval will be required.

Your responsibilities in delivering this research project are set out at the end of this letter.

Yours sincerely

Com Ethics
Ethics Administrator
Computer Science

Please note the following responsibilities of the researcher in delivering the research project:

- The project must abide by the University's Research Ethics Policy: https://www.sheffield.ac.uk/research-services/ethics-integrity/policy
- The project must abide by the University's Good Research & Innovation Practices Policy: https://www.sheffield.ac.uk/polopoly_fs/1.671066!/file/GRIPPolicy.pdf
- The researcher must inform their supervisor (in the case of a student) or Ethics Administrator (in the case of a member of staff) of any significant changes to the project or the approved documentation.
- The researcher must comply with the requirements of the law and relevant guidelines relating to security and confidentiality of personal data.
- The researcher is responsible for effectively managing the data collected both during and after the end of the project in line with best practice, and any relevant legislative, regulatory or contractual requirements.

Appendix C

Chapter3 Questionnaire Copy with Consent Form and Information Sheet

Survey with consent form

*Required

Information Sheet

Invitation:

You are being invited to take part in a research project. Before you decide whether or not to participate, it is important for you to understand why the research is being done and what it will involve. Please take time to read the following information carefully and discuss it with others if you wish. Ask us if there is anything that is not clear or if you would like more information. Take time to decide whether or not you wish to take part. Thank you for reading this.

What is the project's purpose?

It is well known that data structures plus algorithms equal programs, so they are crucial for computer science learners, however, learning algorithms and data structures can be challenging for beginners. We are conducting a project using an object-oriented programming language designed to better assist first-year students in learning data structures and algorithms. The University of Sheffield is made up of students with different backgrounds, some of whom may not have a superior computer background in high school. This means that they may find it difficult when faced with the abstract algorithms and data structures.

Therefore, this project is focused on designing and building a Data Structures and Algorithms Visualisation Tool for Beginners in the Department of Computer Science.

Why have I been chosen?

Although this project is focused on designing and building a Data Structures and Algorithms Visualisation Tool for Beginners in the Department of Computer Science at the University of Sheffield, All students in the UK who have the related experience will provide the positive effect to this project and make the final design better if they are happy to fill up the questionnaire.

Do I have to take part?

It is up to you to decide whether or not to take part. If you do decide to take part you will be given this information sheet to keep (and be asked to sign a consent form) and you can still withdraw at any time* without any negative consequences. You do not have to give a reason. If you wish to withdraw from the research, please contact the person in charge of the project: Hongyu Wang(hwang135@sheffield.ac.uk).

Please note that by choosing to participate in this research, this will not create a legally binding agreement, nor is it intended to create an employment relationship between you and the University of Sheffield.

What will happen to me if I take part? What do I have to do?

- You will participate in and complete the survey by filling out a Google questionnaire with around 10 to 14 multiple/single choice questions online.

- The types of questions you will answer broadly fall into two categories: 1. Your experience in learning data structures and algorithms. 2. Problem-solving methods you have tried during the learning process.

- You may take around 5 to 8 mins to finish all the questions from the questionnaire.

- You need to think and answer each question as truthfully as possible based on your own situation.

- According to the UK's **Information Commissioner's Office (ICO)**, your views on the questions in the questionnaire will be classified as **personal data** and they will be collected in this survey.

What are the possible disadvantages and risks of taking part?

We are not aware of any risks associated with taking part in the project.

Will my taking part in this project be kept confidential?

All the information that we collect about you during the course of the research will be kept strictly confidential and will only be accessible to members of the research team. You will not be able to be identified in any reports or publications unless you have given your explicit consent for this. If you agree to us sharing the information you provide with other researchers (e.g. by making it available in a data archive) then your personal details will not be included unless you explicitly request this.

What is the legal basis for processing my personal data?

According to data protection legislation, we are required to inform you that the legal basis we are applying in order to process your personal data is that 'processing is necessary for the performance of a task carried out in the public interest' (Article 6(1)(e)).

Further information can be found in the University's Privacy Notice: https://www.sheffield.ac.uk/govern/data-protection/privacy/general.

What will happen to the data collected, and the results of the research project?

Due to the nature of this research it is very likely that other researchers may find the data collected to be useful in answering future research questions. We will ask for your explicit consent for your data to be shared in this way.

Who is organising and funding the research?

The project was initiated and funded by the Department of Computing at the University of Sheffield as a dissertation project for third-year undergraduates.

The person in charge of the project: Hongyu Wang (hwang135@sheffield.ac.uk)

Supervisor: Dr Mike Stannett(m.stannett@sheffield.ac.uk)

No external organisation/third party funds the project, except the University of Sheffield.

Who is the Data Controller?

The University of Sheffield will act as the Data Controller for this study. This means that the University is responsible for looking after your information and using it properly.

Who has ethically reviewed the project?

This project has been ethically approved via the University of Sheffield's Ethics Review Procedure, as administered by the department of computer science.

What if something goes wrong and I wish to complain about the research or report a concern or incident?

If you are dissatisfied with any aspect of the research and wish to make a complaint, please contact Hongyu Wang (hwang135@sheffield.ac.uk) in the first instance.

If you feel your complaint has not been handled in a satisfactory way you can contact the Head of the Department of Computer Science Professor Guy Brown(g.j.brown@sheffield.ac.uk).

If the complaint relates to how your personal data has been handled, you can find information about how to raise a complaint in the University's Privacy Notice: https://www.sheffield.ac.uk/govern/data-protection/privacy/general.

Contact for further information

The person in charge of the project: Hongyu Wang (hwang135@sheffield.ac.uk).

Supervisor: Dr Mike Stannett(m.stannett@sheffield.ac.uk).

The Head of the Department of Computer Science: Professor Guy Brown(g.j.brown@sheffield.ac.uk).

*** Participants can withdraw from any on-going or future data collection, but the data collection stage of survey will end 14 days after the questionnaire released and all data will be collated and anonymized (not identifiable), so data cannot be removed from the study/project beyond this deadline.**

Consent Form

Taking part in the project:

- *I have read and understood the project information sheet dated 18/10/2022 or the project has been fully explained to me. (If you answer No to this question please do not proceed with this consent form until you are fully aware of what your participation in the project will mean)*
- *I have been given the opportunity to ask questions about the project.*
- *I agree to take part in the project. I understand that taking part in the project will include completing a questionnaire.*
- *I understand that by choosing to participate as a volunteer in this research, this does not create a legally binding agreement nor is it intended to create an employment relationship with the University of Sheffield.*
- *I understand that I can withdraw from the research/study, with or without notice, at any time. I understand that I do not have to give any reasons for why I no longer want to take part and there will be no adverse consequences if I choose to withdraw.*

How my information will be used during and after the project:

- *I understand my personal details will not be revealed to people outside the project.*
- *I understand and agree that my words may be quoted in publications, reports, web pages, and other research outputs. I understand that I will not be named in these outputs unless I specifically request this.*
- *I understand and agree that other authorised researchers will have access to this data only if they agree to preserve the confidentiality of the information as requested in this form.*
- *I understand and agree that other authorised researchers may use my data in publications, reports, web pages, and other research outputs, only if they agree to preserve the confidentiality of the information as requested in this form.*
- *I give permission for the personal data that I provide to be deposited in the University of Sheffield Google Drive so it can be used for future research and learning.*
- *I agree to assign the copyright I hold in any materials generated as part of this project to The University of Sheffield.*

1. I agree to participate in the research project under the conditions described above. *

Mark only one oval.

- ◯ YES, I confirm that I have read and understand the Participant Information Sheet and voluntarily AGREE to participate in the project.
- ◯ NO, I do not wish to participate.

academic institution

2. Which academic institution do you belong to? *

Mark only one oval.

- ◯ The University of Sheffield
- ◯ Other

Survey continue Q1

3. Q1 What is your subjects? *

Mark only one oval.

- ◯ BEng/MEng Software Engineering. *Skip to question 6*
- ◯ BSc/MSc Computer Science (AI & Computer Science). *Skip to question 6*
- ◯ Other Engineering subjects. *Skip to question 4*
- ◯ Other Science subjects *Skip to question 4*
- ◯ Other *Skip to question 4*

Survey continue Q2

4. Do you have any programming experience (Regardless of proficiency)? *

Mark only one oval.

- ◯ Yes *Skip to question 5*
- ◯ No

Survey continue Q3

5. Did you have any experience of studying data structures and algorithms before ? *

Mark only one oval.

- ◯ Yes, I have known or learned about data structures and algorithms before. *Skip to question 6*
- ◯ No, but I am currently learning data structures and algorithms through some modules. *Skip to question 6*
- ◯ No, but maybe I need to learn related knewledge in future.
- ◯ Know nothing about data structures and algorithms, don 't plan/need to study them.

Survey continue Q4

6. How much did you know data structures and algorithms? *

Mark only one oval.

- ◯ Beginner *Skip to question 7*
- ◯ Imtermediate *Skip to question 7*
- ◯ Advanced *Skip to question 7*
- ◯ I am not sure *Skip to question 7*

Survey continue Q12

7. What difficulties did you encounter in learning data structures and algorithms? *

Tick all that apply.

- [ ] It seems like a "gap" between real-world programming and their working princples.
- [ ] It is abstract
- [ ] Hard to practice.
- [ ] There are currently not many scenarios where I can practice them.
- [ ] I often forget points and have to review them even if I fully understand how they work.
- [ ] No difficulties.
- [ ] Other: _____

Skip to question 8

Survey continue Q5

8. Recall: When I was a beginner (or I am now), sometimes I had a clear understanding of how an algorithm works from the lectures, but was confused about the technical details of how they are represented and behaved in real progroms. *

Mark only one oval.

- ◯ Strongly agree *Skip to question 9*
- ◯ Agree *Skip to question 9*
- ◯ Neutral *Skip to question 9*
- ◯ Disagree *Skip to question 9*
- ◯ Strongly Disagree *Skip to question 9*

Survey continue Q6

9. Recall: When I was a beginner (or I am now), sometimes I had a clear understanding of how an algorithm works from the lectures, but still felt confused about how to practice them(eg. using the array /ArrayList to construct a binary tree). *

Mark only one oval.

- Strongly agree *Skip to question 10*
- Agree *Skip to question 10*
- Neutral *Skip to question 10*
- Disagree *Skip to question 13*
- Strongly Disagree *Skip to question 13*

Survey continue Q7

10. Recall: When I was a beginner (or I am now),I had the experience of learning the process of an algorithm works from videos(eg. From Youtube), but still felt confused about how to code it with bugs free. *

Mark only one oval.

- Strongly agree *Skip to question 11*
- Agree *Skip to question 11*
- Neutral *Skip to question 11*
- Disagree *Skip to question 11*
- Strongly Disagree *Skip to question 11*

Survey continue Q8

11. I have the experience of using the visiualization tool(or watching youtube video) to understand the working processes of algorithms, but still felt at a loss about how to code them in real program(eg. solve a sorting problem by Java/Python). *

Mark only one oval.

- ◯ Strongly agree *Skip to question 12*
- ◯ Agree *Skip to question 12*
- ◯ Neutral *Skip to question 12*
- ◯ Disagree *Skip to question 12*
- ◯ Strongly Disagree *Skip to question 12*

Survey continue Q9

12. Choose from the options below that you think are good ways to learn about data structures and algorithms: *

Tick all that apply.

- ☐ Watch videos on Youtube.
- ☐ Watch Uni lectures.
- ☐ Practice on site (eg. LeetCode) after learning.
- ☐ Choose an visualization tool of data structure and algorithms to aid your learning.
- ☐ Other: _____

Skip to question 14

Survey continue Q11

13. Tell us the reason why you disgree/strongly disgree previous two questions? *

_____

Skip to question 12

Survey continue Q10

14. Choose from the options below that you think is the *best* way to learn about data structures and algorithms: *

Mark only one oval.

- ◯ Watch videos on Youtube.
- ◯ Watch Uni lectures.
- ◯ Practice on site (eg. LeetCode) after learning.
- ◯ Choose an visualization tool of data structure and algorithms to aid your learning.
- ◯ Other: _____

Appendix D

README Document

User-Defined Data Structures and Algorithms Visualization Teaching Tool for Beginners

Installation & Usage：

This project is based on JDK 14 and can be run using native Java commands like `javac`. Please follow the steps below to set up and run the project:

1. **Install JDK 14:**

If you don't have JDK 14 installed on your system, please download and install it from the official website. Follow the installation instructions provided for your specific operating system.

2. **Compile the project:**

To compile the project, navigate to the `src/main` directory and use the `javac` command to compile the main class file:

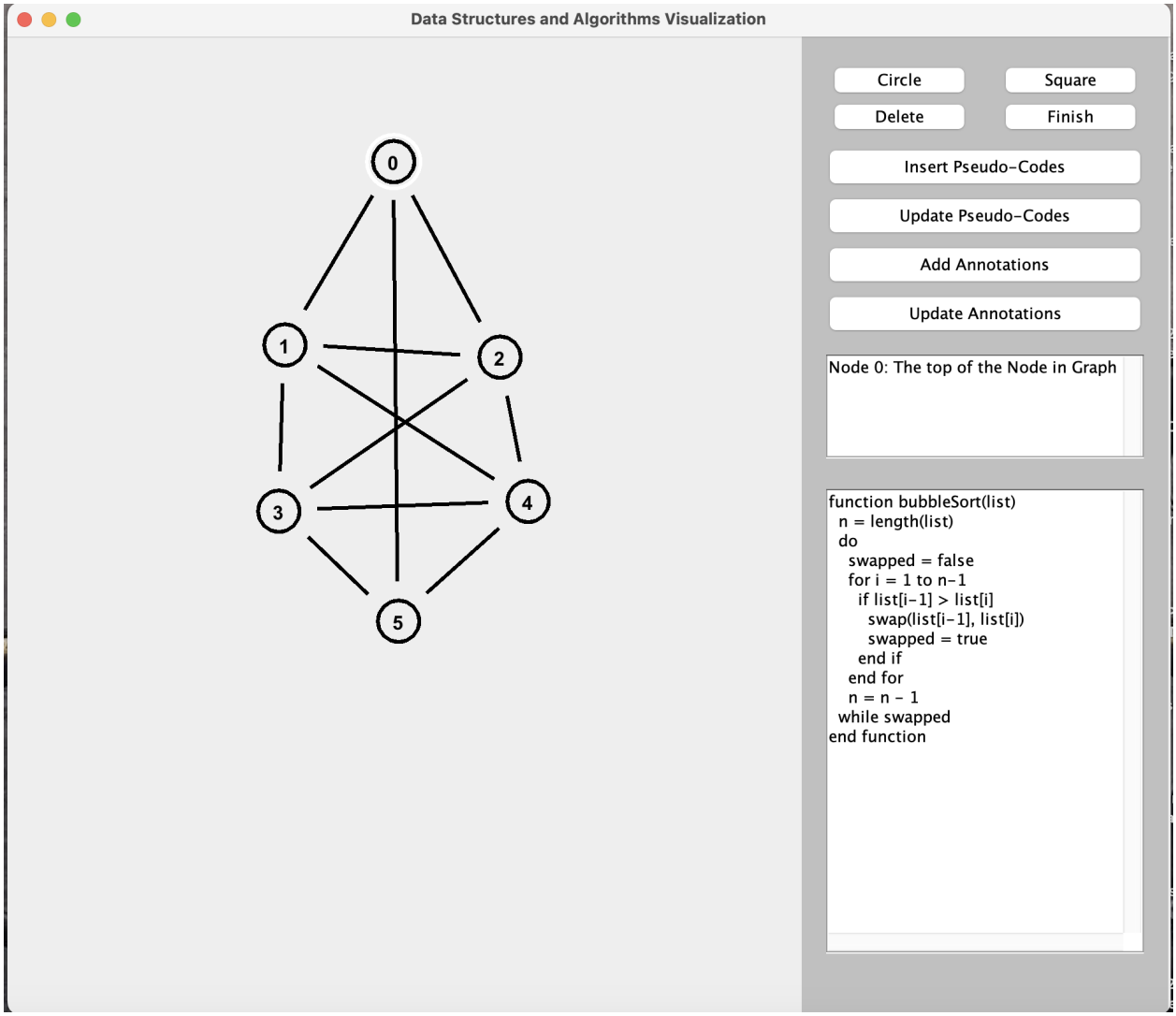
```
cd src/main
javac Run.java
```

3. **Run the project:**

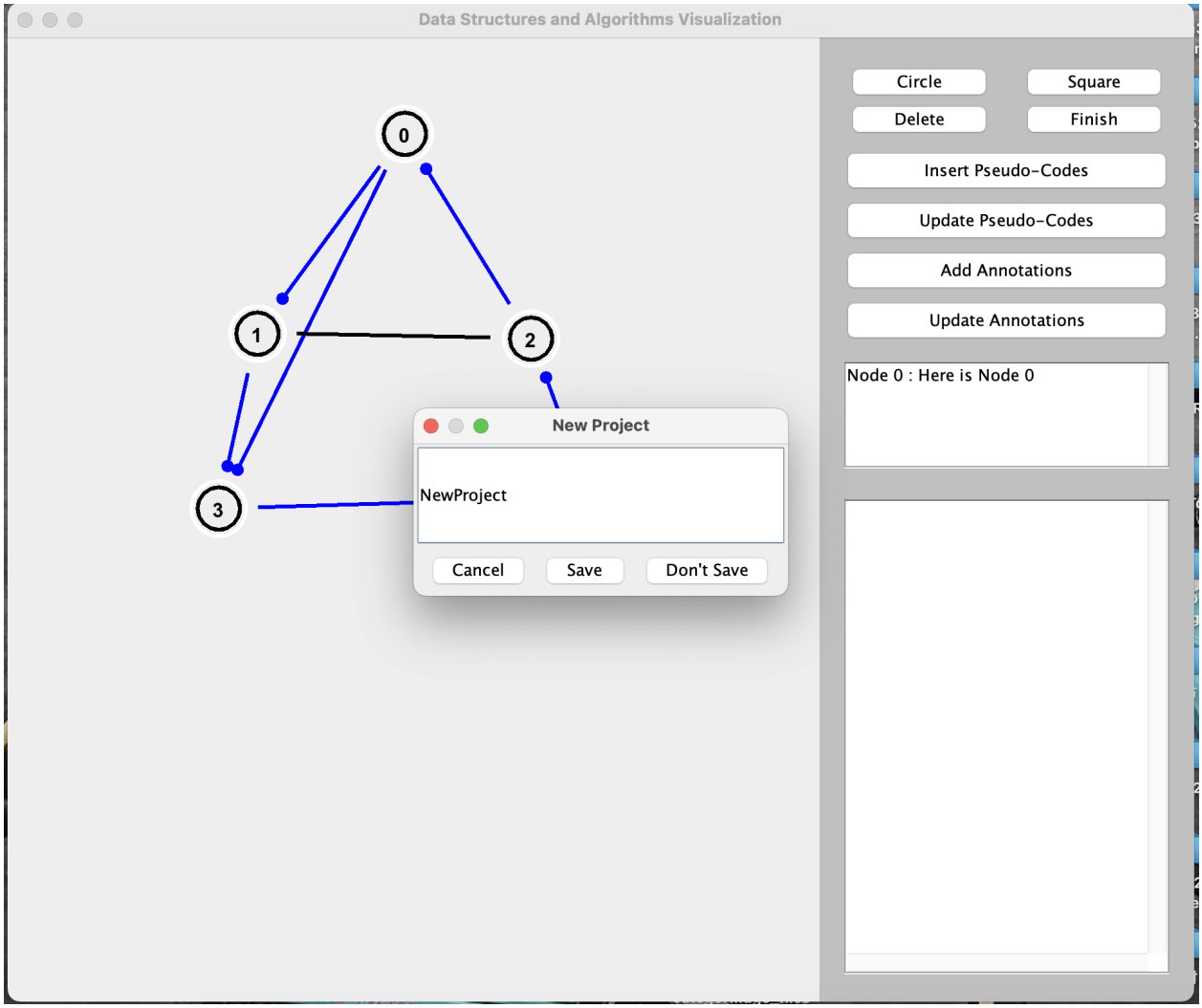
```
cd src/main
java Run.java
```

4. **Use The Application:**

- Use the Canvas UI to design the data structure eg. A tree or graph, and insert the source code or pseudo-code to system, you are also allowed to add annotations by buttons in controll panel:



- After finish the design, click "Finish" button to name and create the new project folder, the system will automatically upload the graphics you created and launch the animation player UI to display it.



- To use the animation profile to create the animation, navigate to the `src/user/your project folder` directory and use the animation library commands in the `src/user/your project folder/AnimationController.java` file to define the animation steps.
- Please Make Sure to Attend The Online Training !
- ….
- When you hope to run the `AnimationController.java` to launch the the Animation Player UI, you need to compile the `AnimationController.java` :

```
cd src/user/your project folder
javac AnimationController.java
```

- Then, run it to launch and preview your animation:

```
cd src/user/your project folder
java AnimationController.java
```

Appendix E

User Testing Questionnaire Copy with Consent Form and Information Sheet

User Testing Quiz

* Indicates required question

Information Sheet

Information Sheet

Invitation:

You are being invited to take part in a research project. Before you decide whether or not to participate, it is important for you to understand why the research is being done and what it will involve. Please take time to read the following information carefully and discuss it with others if you wish. Ask us if there is anything that is not clear or if you would like more information. Take time to decide whether or not you wish to take part. Thank you for reading this.

What is the project's purpose?

It is well known that data structures plus algorithms equal programs, so they are crucial for computer science learners, however, learning algorithms and data structures can be challenging for beginners. We are conducting a project using an object-oriented programming language designed to better assist first-year students in learning data structures and algorithms. The University of Sheffield is made up of students with different backgrounds, some of whom may not have a superior computer background in high school. This means that they may find it difficult when faced with the abstract algorithms and data structures.

Therefore, this project is focused on designing and building a Data Structures and Algorithms Visualisation Tool for Beginners in the Department of Computer Science.

Why have I been chosen?

Although this project is focused on designing and building a Data Structures and Algorithms Visualisation Tool for Beginners in the Department of Computer Science at the University of Sheffield, All students in the UK who have the related experience will provide the positive effect to this project and make the final design better if they are happy to fill up the questionnaire.

Do I have to take part?

It is up to you to decide whether or not to take part. If you do decide to take part you will be given this information sheet to keep (and be asked to sign a consent form) and you can still withdraw at any time* without any negative consequences. You do not have to give a reason. If you wish to withdraw from the research, please contact the person in charge of the project: Hongyu Wang(hwang135@sheffield.ac.uk).

Please note that by choosing to participate in this research, this will not create a legally binding agreement, nor is it intended to create an employment relationship between you and the University of Sheffield.

What will happen to me if I take part? What do I have to do?

- You will participate in and complete the survey by filling out a Google questionnaire with around 10 to 14 multiple/single choice questions online.

- The types of questions you will answer broadly fall into two categories: 1. Your experience in learning data structures and algorithms. 2. Problem-solving methods you have tried during the learning process.

- You may take around 5 to 8 mins to finish all the questions from the questionnaire.

- You need to think and answer each question as truthfully as possible based on your own situation.

- According to the UK's **Information Commissioner's Office (ICO)**, your views on the questions in the questionnaire will be classified as **personal data** and they will be collected in this survey.

What are the possible disadvantages and risks of taking part?

We are not aware of any risks associated with taking part in the project.

Will my taking part in this project be kept confidential?

All the information that we collect about you during the course of the research will be kept strictly confidential and will only be accessible to members of the research team. You will not be able to be identified in any reports or publications unless you have given your explicit consent for this. If you agree to us sharing the information you provide with other researchers (e.g. by making it available in a data archive) then your personal details will not be included unless you explicitly request this.

What is the legal basis for processing my personal data?

According to data protection legislation, we are required to inform you that the legal basis we are applying in order to process your personal data is that 'processing is necessary for the performance of a task carried out in the public interest' (Article 6(1)(e)).

Further information can be found in the University's Privacy Notice: https://www.sheffield.ac.uk/govern/data-protection/privacy/general.

What will happen to the data collected, and the results of the research project?

Due to the nature of this research it is very likely that other researchers may find the data collected to be useful in answering future research questions. We will ask for your explicit consent for your data to be shared in this way.

Who is organising and funding the research?

The project was initiated and funded by the Department of Computing at the University of Sheffield as a dissertation project for third-year undergraduates.

The person in charge of the project: Hongyu Wang (hwang135@sheffield.ac.uk)

Supervisor: Dr Mike Stannett(m.stannett@sheffield.ac.uk)

No external organisation/third party funds the project, except the University of Sheffield.

Who is the Data Controller?

The University of Sheffield will act as the Data Controller for this study. This means that the University is responsible for looking after your information and using it properly.

Who has ethically reviewed the project?

This project has been ethically approved via the University of Sheffield's Ethics Review Procedure, as administered by the department of computer science.

What if something goes wrong and I wish to complain about the research or report a concern or incident?

If you are dissatisfied with any aspect of the research and wish to make a complaint, please contact Hongyu Wang (hwang135@sheffield.ac.uk) in the first instance.

If you feel your complaint has not been handled in a satisfactory way you can contact the Head of the Department of Computer Science Professor Guy Brown(g.j.brown@sheffield.ac.uk).

If the complaint relates to how your personal data has been handled, you can find information about how to raise a complaint in the University's Privacy Notice: https://www.sheffield.ac.uk/govern/data-protection/privacy/general.

Contact for further information

The person in charge of the project: Hongyu Wang (hwang135@sheffield.ac.uk).

Supervisor: Dr Mike Stannett(m.stannett@sheffield.ac.uk).

The Head of the Department of Computer Science: Professor Guy Brown(g.j.brown@sheffield.ac.uk).

*** Participants can withdraw from any on-going or future data collection, but the data collection stage of survey will end 14 days after the questionnaire released and all data will be collated and anonymized (not identifiable), so data cannot be removed from the study/project beyond this deadline.**

Consent Form

Taking part in the project:

- *I have read and understood the project information sheet dated 18/10/2022 or the project has been fully explained to me. (If you answer No to this question please do not proceed with this consent form until you are fully aware of what your participation in the project will mean)*
- *I have been given the opportunity to ask questions about the project.*
- *I agree to take part in the project. I understand that taking part in the project will include completing a questionnaire.*
- *I understand that by choosing to participate as a volunteer in this research, this does not create a legally binding agreement nor is it intended to create an employment relationship with the University of Sheffield.*
- *I understand that I can withdraw from the research/study, with or without notice, at any time. I understand that I do not have to give any reasons for why I no longer want to take part and there will be no adverse consequences if I choose to withdraw.*

How my information will be used during and after the project:

- *I understand my personal details will not be revealed to people outside the project.*
- *I understand and agree that my words may be quoted in publications, reports, web pages, and other research outputs. I understand that I will not be named in these outputs unless I specifically request this.*
- *I understand and agree that other authorised researchers will have access to this data only if they agree to preserve the confidentiality of the information as requested in this form.*
- *I understand and agree that other authorised researchers may use my data in publications, reports, web pages, and other research outputs, only if they agree to preserve the confidentiality of the information as requested in this form.*
- *I give permission for the personal data that I provide to be deposited in the University of Sheffield Google Drive so it can be used for future research and learning.*
- *I agree to assign the copyright I hold in any materials generated as part of this project to The University of Sheffield.*

1. I agree to participate in the research project under the conditions described above. *

Mark only one oval.

- ◯ YES, I confirm that I have read and understand the Participant Information Sheet and voluntarily AGREE to participate in the project.
- ◯ NO, I do not wish to participate.

UI Design

2. **UI Design:** What do you think of the appearance of the app's UI? *

Mark only one oval.

- ◯ Satisfactory
- ◯ Mostly Satisfactory
- ◯ Average
- ◯ Mostly unsatisfactory
- ◯ Very unsatisfactory

UI Design Improvement

3. **UI Design:** How do you think the appearance of the UI should be improved in future versions?

_____

_____

_____

_____

_____

Skip to question 4

Ease of use

4. **Ease of use:** How easy or hard was it for you to figure out the content in each subpanel on Animation Player UI? *

Mark only one oval.

- ◯ Very easy
- ◯ Easy
- ◯ Okay
- ◯ Hard
- ◯ Very hard

5. **Ease of use:** How easy or hard was it for you to figure out how to use the animation library to create animation? *

Mark only one oval.

- ◯ Very easy
- ◯ Easy
- ◯ Okay
- ◯ Hard
- ◯ Very Hard
- ◯ Other: _____

6. **Ease of use:** How easy or hard was it for you to figure out how to design logical data structure on Canvas UI? *

Mark only one oval.

- ◯ Very easy
- ◯ Easy
- ◯ Okay
- ◯ Hard
- ◯ Very Hard
- ◯ Other: _____

Functionality practical

7. **Functionality practical:** Are you satisfied with the software's running speed? *

Mark only one oval.

- Very Satisfied
- Satisfied
- Neutral
- Dissatisfied
- Very Dissatisfied

8. **Functionality practical:** Do you find the installation process of the software easy? *

Mark only one oval.

- Very Easy
- Easy
- Neutral
- Difficult
- Very Difficult

Application features

9. **Application features:** What is your favourite feature? *

Mark only one oval.

- Multi-content display
- Custom animations
- Drawable graphic data structures
- Local storage
- No reliance on the internet
- Other: _____

Ease of use improvement : Design graphic data structure.

10. **Ease of use:** Any comments?

Ease of use improvement: Creating Animation by animation profile and library.

11. **Ease of use:** Any comments?

Ease of use Animation player UI improvement

12. **Ease of use:** Provide us anything make you confused on Animation Player UI?