

AE2GRP Final Group Report

ANIMATION OF SORTING ALGORITHMS

April 28, 2017

Group ID: 5
Group Members:

Jiaying SUN	6515778
Kan LIU	6515770
Muyi JIANG	6513225
Yangyu GAO	6515761
Zhefeng ZHOU	6515792
Zhe REN	6515775

Supervisor: Heshan DU

Contents

1	Introduction	1
1.1	Project Introduction	1
1.2	Team Introduction	1
1.3	Introduction of Sorting Algorithm	2
2	Background Research	3
2.1	Existing Similar Systems and Market Research	3
2.2	Technical Research	7
2.3	Use Cases	9
3	Requirement Specification	10
3.1	Functional Requirements	10
3.2	Non-functional Requirements	11
4	Design	11
4.1	UML Model	11
4.1.1	Use Case Diagram	12
4.1.2	Sequence Diagram	14
4.2	User Interface Design	16
4.2.1	Single Algorithm Page	16
4.2.2	Comparing Part	18
4.2.3	Menu Bar	19
5	Implementation	20
5.1	Key Implementation Decisions	20
5.2	Overview of the developed source code hierarchy	23
5.3	Visualising Sorting Algorithms	23
5.4	Implementing Graphical User Interface	25
5.4.1	Single Algorithm Page	25
5.4.2	Comparison Page	26
5.4.3	Menu bar	28
5.4.4	Help box	29
5.4.5	Preference	29
5.4.6	Guideline	30
6	Testing	30
6.1	Black Box Testing	30
6.2	Evaluation and Discussion	33
6.2.1	User Interface Testing	33
6.2.2	Future Work	33
7	Reflective Comments	34
7.1	Problems Encountered and Solutions	34

7.2	What we have learned	35
8	Conclusion	35
9	Appendix	37
9.1	Meeting Minutes	37
9.2	Sorting Algorithms Pseudocode	53
9.3	User Manual	58
9.4	User Interface Design Update	84
10	References	89

1 Introduction

1.1 Project Introduction

Most beginners of computer science need to learn sorting algorithms. A sorting algorithm is a sequence of steps which provides a solution to sorting problems. Sorting is often used in our daily life. For instance, if teachers need to enter students' exam marks by student IDs to make the process of entering marks easier, they can use sorting algorithms to sort answer sheet first. Sorting algorithms are an essential part of algorithms and data structures. By learning sorting algorithms, people can save time when solving sorting problems. However, sorting algorithms are challenging to understand by only reading the descriptions in pseudo-code or programming languages. It is necessary to find some methods to enable more efficient ways of learning sorting algorithms.

The problem that this project aims to solve is: "How to develop a software tool to help people learn sorting algorithms more efficiently."

To solve this problem, we find that a visualization is a useful tool. It describes the execution of sorting algorithms as a continuous sequence of graphical images. Michael claims that animations help people remember algorithms better because they explain a dynamic, evolving process clearly [1]. Lawrence, Badre, and Stasko utilized animations to help teach undergraduates about minimum spanning tree algorithms in 1994 [2]. A significant benefit was found when students could interact with the algorithm animations in a laboratory context. In this project, we will develop a creative open-sourced software to visualize sorting algorithms. This software should help people understand the correctness and efficiency of the sorting algorithms, facilitate learning and increase the interest of people.

1.2 Team Introduction

Our team consists of five Computer Science undergraduates from University of Nottingham Ningbo China. All the five students are in Part I.

Table 1: Team Role

Name	Role
Zhefeng Zhou	Leader, User Experience Designer
Yangyu Gao	Quality Assurance Manager, Animation Expert
Jiaying Sun	Coordinator
Zhe Ren	User Interface Designer
Kan Liu	Editor
Muyi Jiang	Repository Manager, Coordinator

As shown in Table 1, Different roles are allocated to team members. Zhefeng Zhou

is our leader to coordinate work activities and he performs as User Experience (UX) Designer to concern with improving the user experience. Yangyu Gao is the Quality Assurance (QA) Manager, who establishes procedures and quality standards and monitors these against agreed targets. Zhe Ren as the User Interface (UI) Designer designs the user interface of the software. Kan Liu as the Editor produces documents and makes sure all the key actions of our project are recorded. Muyi Jiang is the Repository Manager who manages the repository in GitHub and maintains codes and documentations of our project. In addition, Jiaying Sun and Muyi Jiang are Coordinator.

1.3 Introduction of Sorting Algorithm

In this section, the basic knowledge of six sorting algorithms implemented in the software will be introduced. For each of the six sorting algorithms, the input is an unordered sequence, and the output is an ascending order sequence.

1. Bubble Sort

Bubble sort is a simple sorting algorithm. It repeatedly visits the sequence to be sorted, compares each pair of adjacency elements at a time, swaps them if they are in the wrong order. The work of visiting the sequence is repeated until no swaps are required, which indicates the sequence has been sorted. The time complexity of bubble sort algorithm is $O(n^2)$.

2. Insertion Sort

Insertion sort usually applies to sort small amounts of data. It works by building an ordered sequence, scan in the sorted sequence from the back to the front, find the appropriate location and insert for unordered data. Insertion sorting usually uses in-place sorting, so in the process scanning from the back to front, it needs to repeatedly shift the sorted elements gradually backward, to provide insertion space for the latest elements. The time complexity of insertion sort algorithm is $O(n^2)$.

3. Selection Sort

Selection sort works as follows. First, find the smallest element in the unordered sequence, store it at the beginning of the sorted sequence, and then continue looking for the smallest element from the remaining unordered elements, store it at the end of the sorted sequence. And so on until all the elements are sorted. The time complexity of selection sort algorithm is $O(n^2)$.

4. Quick Sort

Quick sort is a randomized sorting algorithm based on the divide-and-conquer paradigm. "Divide" means that divide sequence into three parts which is L, E, and G by picking a pivot x. The elements in L is less than x, and the elements in E is equal to x, meanwhile, the elements in G is greater than x. "Conquer" means join L, E, and G together. The basic idea of quick sort is, first call the "Divide" operation on the unsorted sequence to get L, E and G parts. Then recursively call

the "Divide" operation on L and G parts until the size of L and G is equal to zero or one. Finally, recursively invoke "Conquer" operation to join L, E, and G parts to obtain an ordered sequence. The time complexity of quick sort algorithm is $O(n \log n)$.

5. Merge Sort

Merge sort is a stable sorting algorithm, and it is an effective sorting algorithm based on merge operations, which applies "Divide and Conquer". The basic idea is to merge the ordered subsequences to obtain a fully ordered sequence. In other words, data in each sub-sequence is ordered first, and then the sub-sequences are ordered. The time complexity of merge sort algorithm is $O(n \log n)$.

6. Heap Sort

Heap sort is implemented by using heap data structure. A Heap is a complete binary tree, and it includes maximum heap and minimum heap. We introduce maximum heap as an example. The requirement for a maximum heap is that the value of each node is not greater than the value of its parent node, which means the maximum value is at the top of the heap. The basic idea of heap sort is, first adjust the unsorted sequence to a maximum heap, next exchange the element of maximum value with the element at the end of the sequence, then repeat the above two steps until all the elements are sorted. The time complexity of heap sort algorithm is $O(n \log n)$.

2 Background Research

This part provides the background information and research of the software we developed. Section 2.1 describes similar systems, address some possible problems in the system design and discuss the market prospect of the software. Section 2.2 discusses which platform and tools are more suitable for this project. Section 2.3 talks about possible use cases.

2.1 Existing Similar Systems and Market Research

There are a few similar systems which provide the visualization of sorting algorithms. Three typical examples the on the web, Visualgo¹, Algomation², Visualizer³ and two examples on the desktop, SortingVisualization⁴ and Sorting Algorithms Visualizer, are described in this section.

Visualgo

¹<https://visualgo.net/sorting>

²<http://www.algomotion.com/andAlgorithm>

³<http://algo-visualizer.jasonpark.me>

⁴<https://github.com/AlaaAlShammaa/SortingVisualization-JavaFx>

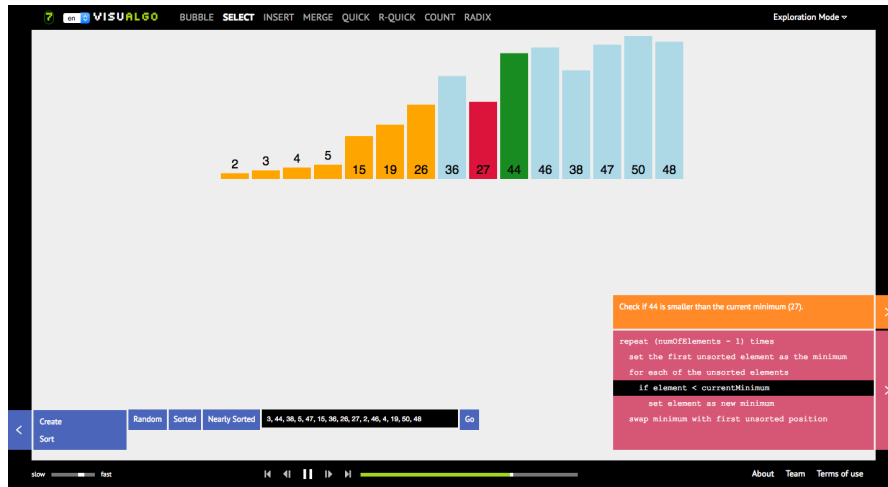


Figure 1: Visualgo

Visualgo was conceptualized in 2011 by Dr. Steven Halim as a tool to help his students to have a better understanding of data structures and algorithms. It is specifically designed for National University of Singapore (NUS) students taking data structure and algorithm classes. This software centers around teaching students the principles of algorithms. As is shown in Figure 1, it contains numerous useful functions. Firstly, the user interface is beautiful as the color of operating block can be changed when users refresh the page. In addition, this system can generate graphical images for random input data or that specified by users. Thirdly, when the animation plays, the pseudo code and explanation of corresponding steps will be shown in the low right corner. This makes users understand an algorithm more easily. However, there are some shortcomings. Some space is not well utilized, such as the space below the animation. The "Start Button" is so small that people might confuse how to start the animation. Similarly, the user guide is in the high right corner which is inconspicuous and users might not know where it is.

Algomation

Algomation is a platform for viewing, creating and sharing any type of algorithm. All algorithms on the site are public and can be viewed and shared by any user of the site. This website is created by Duncan Meech. As the Figure 2 shown, users can search algorithms they want on this website. After selecting an algorithm, the player page will be shown in the Figure 3 which incorporates four main components: Player Visualizer, Player Controls, Player Source code Viewer, and Player Message. The Player Visualizer is in the right part to display the animation. The Player Controls is above the animation part to control the animation which can change the mode and speed of the animation, share to other people, restart the animation and so on. There is a message part in the

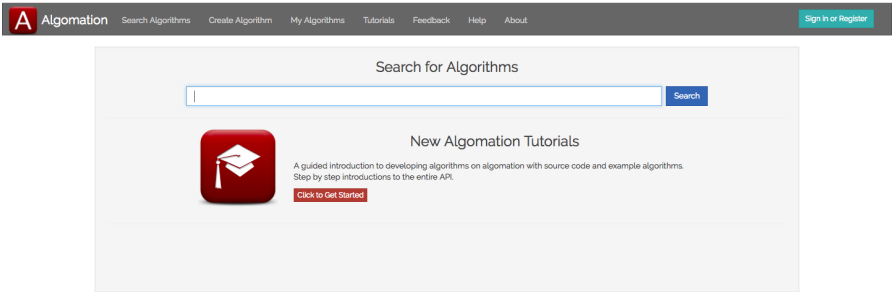


Figure 2: searchAlgomation

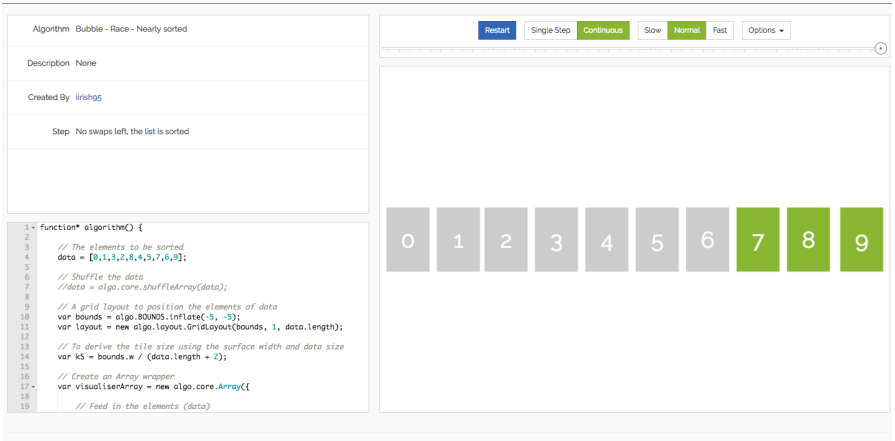


Figure 3: Agomation

upper right hand to describe the message about the visualization such as the author. The last part is in the low right hand to provide the source code. It is more convenient for users to understand the algorithm. There are many useful algorithms with different types of animations in the website. There are also lots of useful functions to help people learning. Moreover, the difference between other software is that people can get a tutorial of developing algorithm animations. They can develop animation to help learn algorithm.

Algorithm Visualizer

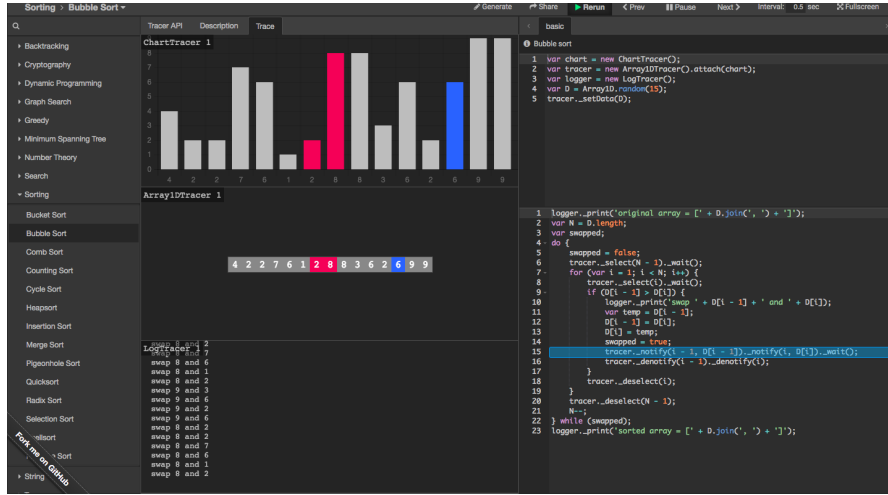


Figure 4: Algorithm Visualizer

Algorithm Visualizer is an open-source software which is developed by Jason Park and others. This software concentrates on visualizing a single sorting algorithm. It is composed of several blocks which are shown in Figure 4. The left side is a list of different algorithms users can choose. The middle part shows two visualizations, ChartTracer 1 and Array1DTracer 1, with the different forms which can meet different needs. LogTracer 1 shows some information to explain each step. On the right side, the whole block displays the JavaScript code of this sorting algorithm. Users can modify the code to change the input of animations. By checking the actual code of animation, the learning interest of users might grow.

SortingVisualization

SortingVisualization is a simple JavaFX application that visualizes the common sorting algorithms through animations. It's an open-source software developed by Alaa Alsham-maa. The software uses JavaFX as the same as our project, the main structure and function are also similar to our software. You can choose 5 different sorting algorithms at the top of the software, and adjust the animation speed by the speed bar on the



Figure 5: SortingVisual

bottom-left corner.

Sorting Algorithms Visualizer

Sorting Algorithm Visualizer is a software based on the Windows system which is shown in Figure 6. Differing from the former two systems, this system focuses on comparing the efficiency of different sorting algorithms. The user interface of this software is succinct. There are also some explanations below each animation. Users can specify inputs by choosing a sorted list, a reversed list or a randomized list. Users also can specify inputs by themselves. However, this software is not very stable and may fail unexpectedly. Moreover, only four sorting algorithms are supported by the system. Users may want to learn more sorting algorithms.

Compared to other similar systems, our system will have three advantages. First, as an open-source software, everyone can use it. Secondly, our system is based on the Windows system and can be used without the Internet. Thirdly, most similar software only focuses on the animation of sorting algorithms rather than comparing the efficiency of them. Our software will not only concentrate on single algorithm but also provide the comparison of sorting algorithms.

2.2 Technical Research

1. GUI Design

There are many frameworks to design GUI for Java. The Abstract Window Toolkit (AWT) is part of the Java Foundation Classes (JFC), the standard API for providing a graphical user interface (GUI) for a Java program. AWT is also the GUI toolkit for a number of Java ME profiles. The swing was developed to provide a more

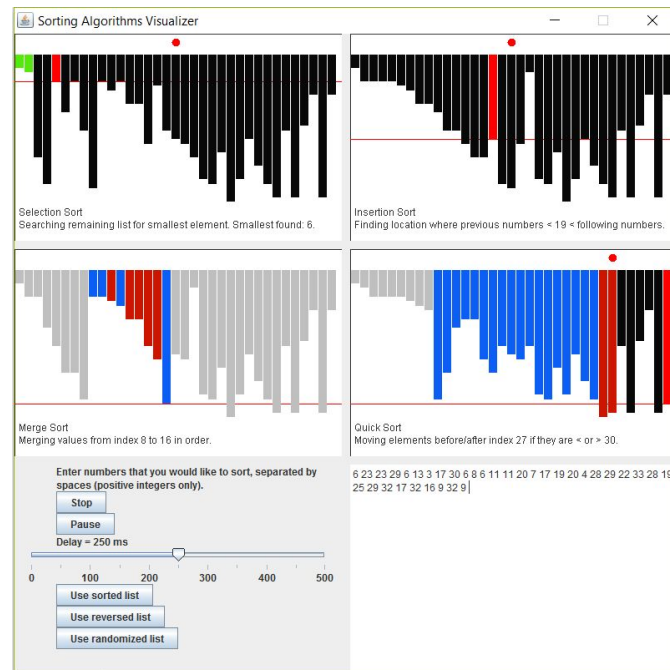


Figure 6: Sorting Algorithms Visualizer

sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT). Swing provides a native look and feel, which emulates the look and feel of several platforms. JavaFX is another software platform for creating and delivering desktop applications, as well as rich internet applications (RIAs) that can run across a wide variety of devices. By using JavaFX, we can also use the web language (HTML, CSS, and JavaScript) indirectly to make our interface more visual.

2. Version control

Git is a distributed version control system, which can keep the history of a file set and can roll back the file to another state (history). GitHub⁵ is a git based web collaboration community. It can work as Git to keep the history of a file and roll back the file state and have a variety of mechanisms for everyone to work together to contribute to the project. In addition, GitHub integrates lots of social features which make users' information and activities visible across open source software projects [3]. Our group decides to use GitHub as a basic tool to develop the software so that we can upload the software project to a repository where everyone can access and work on it, control the project versions, and search some useful open resource distributed in GitHub.

3. Animation

JavaFX has the library to create animation. These codes can be directly used in Java

⁵<https://github.com>

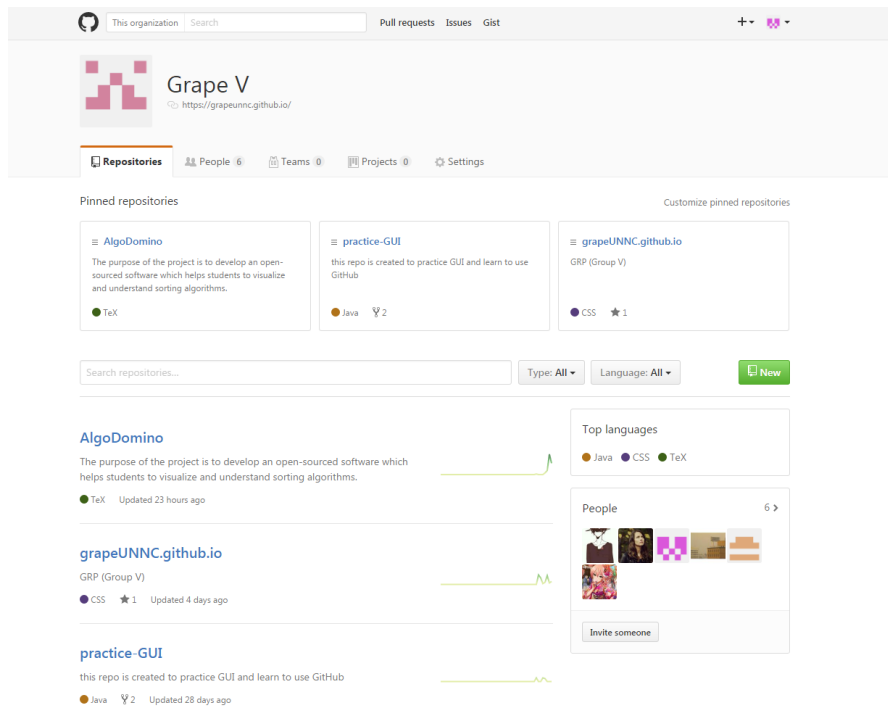


Figure 7: Github page

files and connected with the interface built by JavaFX. Two types of animation are included in JavaFX animation library: Transition animation and Timeline animation. Another most common way to animate nowadays is using JavaScript or CSS. There are many good JavaScript animation libraries on the market, and each of them has specific different features. Each project has its own requirements, but sometimes one library is not sufficient to meet the whole requirements of one project. Some famous and powerful JavaScript libraries are GSAP⁶, Velocity.js⁷ and jQuery⁸.

2.3 Use Cases

It is necessary to find several use cases for our software to plan the goals of the next stages. Here are some possible use cases.

1. Teaching

This software may be used as a convenient tool for teaching. For example, when teachers in class want to teach sorting algorithms, they can use our software to explain the principles. Students would understand them more easily.

⁶<http://greensock.com/gsap>

⁷<http://velocityjs.org/>

⁸<http://jquery.com/>

2. Self-study

Some people may need to learn sorting algorithms. However, they do not have a teacher and they cannot understand sorting algorithms searched from online. This software becomes a good helper to explain knowledge by visualization.

3. Review

People may forget the knowledge they have learned. By using this software, they can revise the sorting algorithms. The comparison of the efficiency of different algorithms would help users learn or reuse the algorithms more quickly.

3 Requirement Specification

This section specifies the requirements for our software. As mentioned before, the purpose of this software is to help users learn principles and efficiency of sorting algorithm. Hence, users should have a basic understanding of what an algorithm is. Typical users include computer science students and those interested in sorting algorithms. The rest of this section presents the specifications of the software. The first part is an overall description of the software and its functional requirements. The second part outlines other related non-functional requirements of the software.

3.1 Functional Requirements

The list provides a description of main functional features. These features are divided into two categories: core features and optional features. Core features are essential to the operation of the software and optional features are additional functionality.

• Core features

1. The software should provide users a guide of how to use core features.
2. The software should allow users to select which sorting algorithms to animate.
3. The software should be able to allow users to define input data by themselves.
4. The software should be able to generate input data randomly.
5. The software should be able to visualize the processes of running a sorting algorithm.
6. The software should allow users to start, stop, slow down, speed up, restart or pick up a point of the animation.
7. The software should be able to show the running sorting algorithm's source code to users.
8. The software should be able to compare the efficiency of different sorting algorithms by showing the animation simultaneously.

9. The software should allow users to select which algorithms to compare.
10. The software should be able to provide the time complexities of sorting algorithms when comparing the efficiency.

- **Optional features**

1. Users should be able to share the animation as video or gif file to some social network websites.
2. The software should allow users to customize the interface, such as the background color of animation, shapes, and colors of basic components used in an animation.
3. Users should be able to download the source code of selected sorting algorithms in several programming languages such as Java, C, Python, JavaScript, and PHP.

3.2 Non-functional Requirements

1. Platform
The software will run on Windows, Mac and Linux with JRE (Java SE Runtime Environment).
2. Capacity
The software allows users to compare two sorting algorithms only.
3. Software Language
All code will be written in Java SE8, JavaFX, and CSS.
4. Online User Documentation and Help System Requirements
The software should allow users to download the source code to local address. All documentation will be made in accordance with requirements pertaining to open source software under the GNU General Purpose License. Additionally, online user documentation will be in the form of Java Platform, Standard Edition 8 API Specification.

4 Design

This section describes the design of the software. Section 4.1 presents two UML models of the design. Section 4.2 shows the user interface design.

4.1 UML Model

The Unified Modeling Language (UML) is a visual, object-oriented, and multipurpose modeling language. It is primarily designed for modeling software systems [4]. From different perspectives of the system, UML defines use case diagram, sequence diagram, class diagram and so on. With UML, we can model the types, properties, and states of

those objects as well as to integrate the corresponding object flows into the activities [4]. In this part, we will analyze use case diagram and sequence diagram for our software according to the requirement specification.

4.1.1 Use Case Diagram

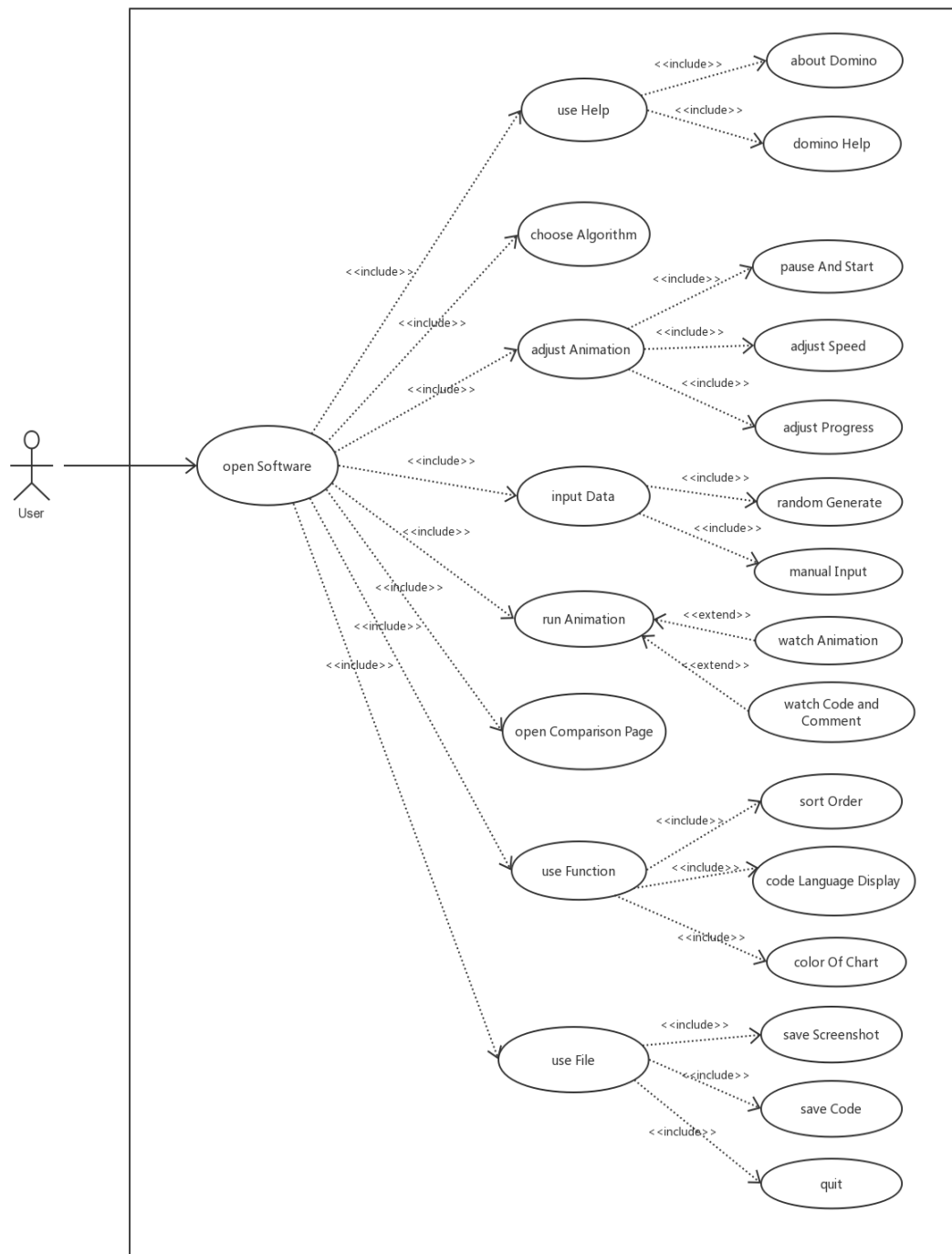


Figure 8: Use case diagram

According to Wegmann and Genilloud, use case diagram is used to represent functions of the system [5]. The advantage of use case diagram is that it can describe the functions of software in a clear and visual way. Figure 8 is a use case diagram drawn on the basis of the functional requirements of our software. From it, we can see that every use case of the user is captured, and most of them can include or extend to different sub use cases. The diagram is designed to describe all the available functions when users open the software. We list some of the available functions as examples, such as users can choose six different sorting algorithms, users can click "start" button to run the animation, users can watch the animation and corresponding algorithm code, users can adjust the speed and the progress of animation and users can use functions in menu bar such as "File", "Function" and "Help".

4.1.2 Sequence Diagram

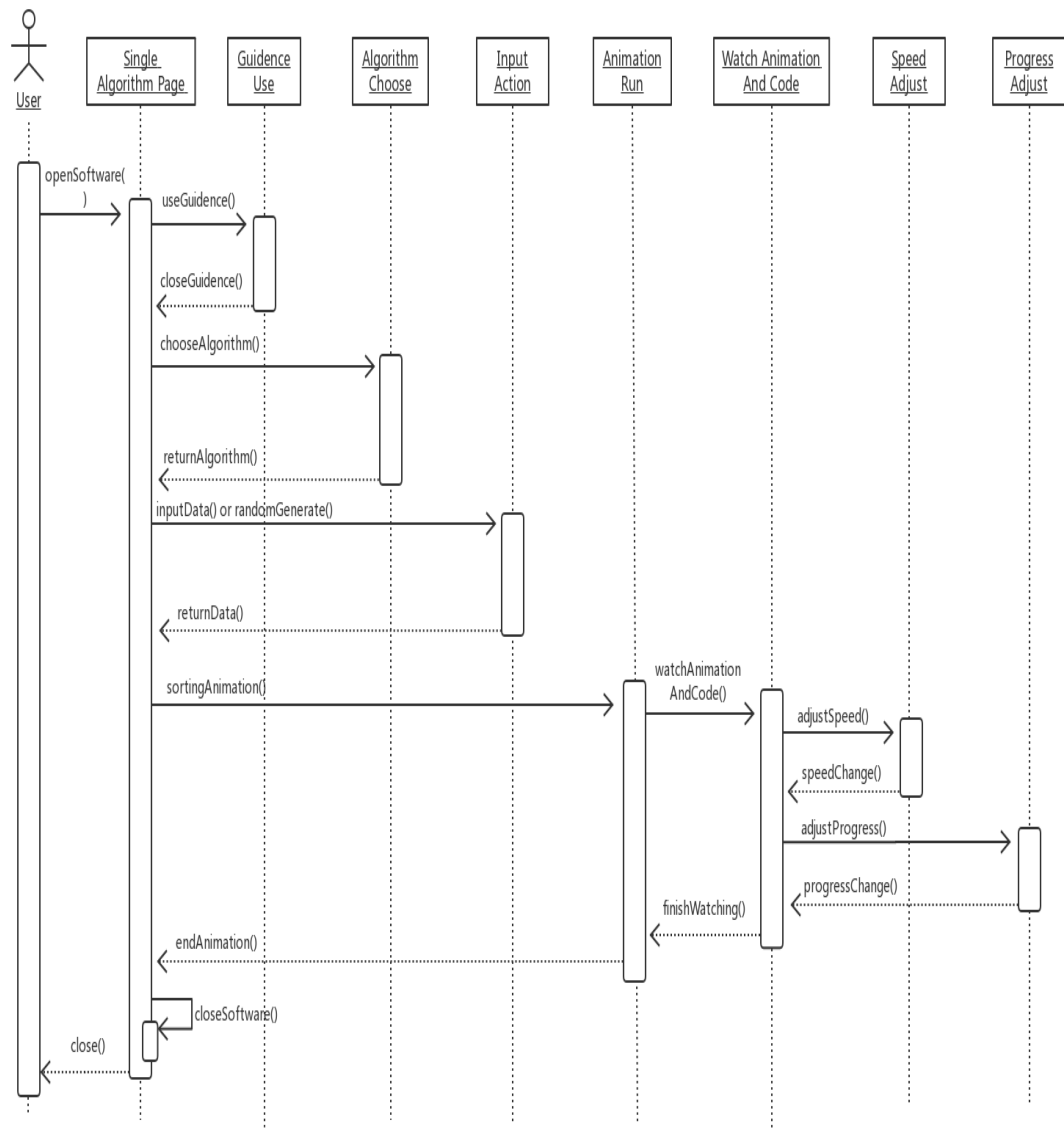


Figure 9: Sequence Diagram

Bell claims that sequence diagram primarily shows the interactions between objects in the sequential order that those events occur [6]. The advantage of the sequence diagram is that it can clearly describe the time line of events happening and the interactions among objects. Figure 9 is a sequence diagram drawn according to the functional requirements of our software. The diagram is designed to describe the sequence of events happening when a user uses the software to watch the animation of one sorting algorithm, first when the user opens the software, he will see the single algorithm page, then he finds the guidance to teach him how to use the software. After the guidance, he selects an algorithm and inputs specific data, next he runs the animation. During the animation, he watches the animation and corresponding algorithm code, meanwhile, he adjusts the speed bar and progress bar of animation. Finally, he closes the software after watching the animation.

4.2 User Interface Design

In our current software, several main functions are added into the user interface. The main part should be a block used for the animation. Since most of the computers users watch videos, I added play/pause button and progress bar at the bottom of the interface. I also added a speed bar to change the speed of the animation. Several toggle buttons are listed at the left-side block to let the users choose the algorithm. For generating the animation, a text field is added to let users input data and a random button to generate data randomly. I also add a **Tab Pane** to swap between single algorithm function and comparing function. The menu bar at the top of the interface includes several extra functions.

4.2.1 Single Algorithm Page

For the single algorithm page of the sorting algorithm animation, the interface is divided into five parts as shown in Figure 10.

1. Algorithm Button Group

The algorithm button group is on the left-hand side of software, and it is used to select sorting algorithms. There are six toggle buttons on behalf of six sorting algorithms, which are Bubble Sort, Insertion Sort, Selection Sort, Quick Sort, Merge Sort and Heap Sort. When users click one of the six buttons, the corresponding sorting algorithm is chosen, and the corresponding algorithm code and the hint will be displayed in algorithm display block.

2. Animation

The main part is the animation. This part contains graphical images generated using several corresponding input or random data. The corresponding data numbers are also printed on the rectangles. When the code in the low right corner is running, the animation changes based on the code of the specified sorting algorithm. The speed of the animation is related to the speed bar in the bottom control block. The

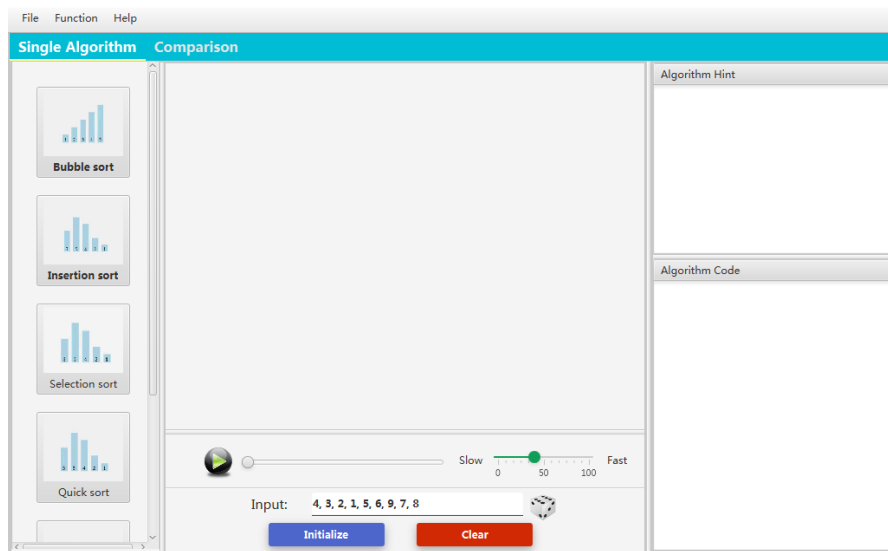


Figure 10: Single Algorithm Page

animation is also controlled by the play button and progress bar.

3. Algorithm Display Block

The algorithm display block is on the right-hand side of software, and it is divided into two parts.

- Algorithm Code:** The lower part is used to show the algorithm code. When users select different sorting algorithms, the algorithm code part will display different codes corresponding to the algorithm users choose. It is convenient for users to the specific code while running the sorting animation.

- Algorithm Hint:** The algorithm hint is shown in the upper part. This function in this part uses hints to explain what the specific code is and how it works. It is similar to the algorithm code part, the only difference is that the function will display the hint of the algorithm rather than code. It can assist users to understand the sorting algorithms in a simple way.

4. Animation Control Block

For the control block, several buttons are involved in controlling the animation.

- Play/Pause Button:** The play button is the basic function to control the animation. When use firstly clicks it, the frame will start to change and the icon of this button will become "Pause" image. When users click it again, the changing frame will be paused and the icon will change to "Play" image.

- Progress Slider:** The progress slider is used to control the progress of animation. It consists of a slide bar and a handle. The slide bar is defined as the timeline of animation. Users can drag the handle on the slide bar to control the animation.
- Speed Bar:** The speed bar is used to control the speed of the animation. Users can drag the slider to make rectangles moving more quickly or slowly.
- Input Field:** The input field is used to fetch the input data so that the software can generate the corresponding images.
- Random Button:** The button with a dice icon is the random button. Users can click it to generate random rectangles with random data.
- Initialize Button:** The initialize button can be used to generate the corresponding image when users want to use input data to implement an algorithm.
- Clear Button:** The clear button is used to delete all content that users typed in the input field.

4.2.2 Comparing Part

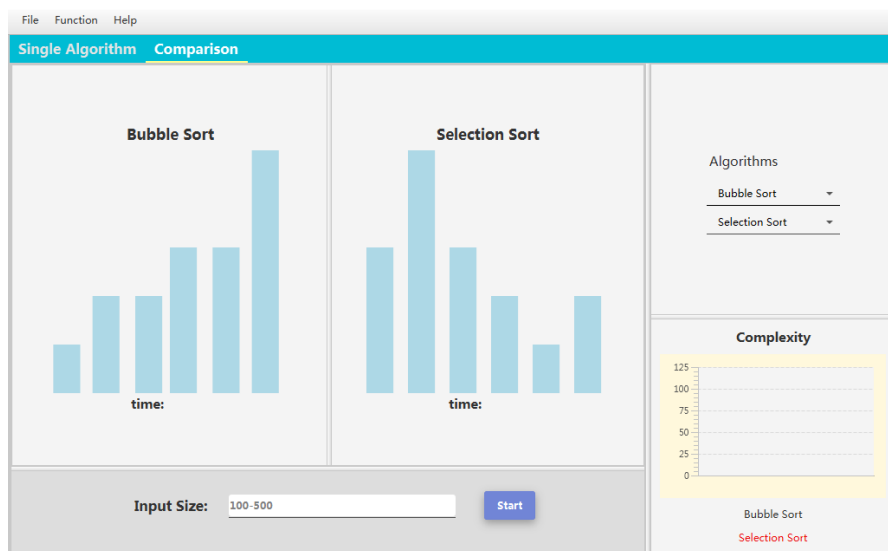


Figure 11: Comparison of the efficiency

The user interface of compare efficiency is shown in Figure 11. Two algorithms both have the same default input and start at the same time after pressing the Start button.

1. Selection Block

Comparing sorting algorithms is another function of our software. In the upper right corner of the interface, there is a selection part where users can select two algorithms to compare. After choosing the algorithms, the animation changes to two single parts which locate in left and right side. Each part is an independent part which has its own animation process.

2. Animation

In each animation part, the execution time shows in the top and the name of the sorting algorithm is shown in the bottom. Below the animation, the control block is similar to the main page. Users can input the number of data in the right corner of this block.

3. Control Block

The control block includes several buttons to control the comparing animation. For the input field, users can input the size of data so that software can generate corresponding random rectangles. When users click the start button, the two comparing frame will animate at the same time.

4. Time Complexity Block

Below the algorithm choosing block, there is a chart shows the time complexity of the two algorithms. After the comparing process, the chart is a clear pattern.

4.2.3 Menu Bar

The software includes three menus, each of which implements basic functions.

1. File

As the name implies, users can use functions in "File" menu bar to perform some operations on files. This menu includes three function items.

●**Save code:** The function is for users to download algorithm code. When users click it, the corresponding sorting algorithm code shows in algorithm display block will be saved in the form of a TXT file. If users don't choose an algorithm before using this function, the software will pop up a warning dialog.

●**Save screenshot:** When users click the function, the current interface will be saved, and the suffix of the screen shot can be .png or .jpg.

●**Quit:** This function is same as the Shutdown function of the software. Users can click it to quit the program.

2. Function

The menu "Function" provides some additional functions related to the algorithm animation.

•**Code Language:** The function is used to select the language of code displayed in the algorithm code block. It provides five different programming languages including "Java", "JavaScript", "C", "C++" and "PHP" for users to choose.

•**Color of Chart:** The function is used to change the color of animation chart. When users click the function item, a new window interface will pop up in which users can choose the color they want. After users select the color, they should click apply button and the color of rectangles will be directly changed.

•**Sort of order:** This function is used change the order that the algorithms sort according to. Two specific choices are included in this menu. Users can choose ascending or descending order to sort the rectangles in the animation block.

3. Help

Users can use the functions in "Help" menu to know more information about the software and the development team.

•**Domino Help:** A guideline interface will show up when users click the function item. In the interface, there are two tabs, the first tab tags the details of each function in single algorithm page, and the second tab also tags the details in comparison page. All of the tags describe what the functions are use for and how to use them.

•**About Domino:** By clicking the "About Domino" function item, a new interface will show up and display the relevant instructions of the software and the development team.

5 Implementation

This section introduces Implementation. Section 5.1 shows the key decisions we made and reasons. Section 5.2 displays the overview of source code hierarchy. Section 5.3 describes how to implement the visualization of sorting algorithms. And section 5.4 shows how we implement the graphical user interface in each part of the software and list some code implementation of functions.

5.1 Key Implementation Decisions

According to the Background research and user interface design, we made some key implementation decisions.

1. Development Methodology

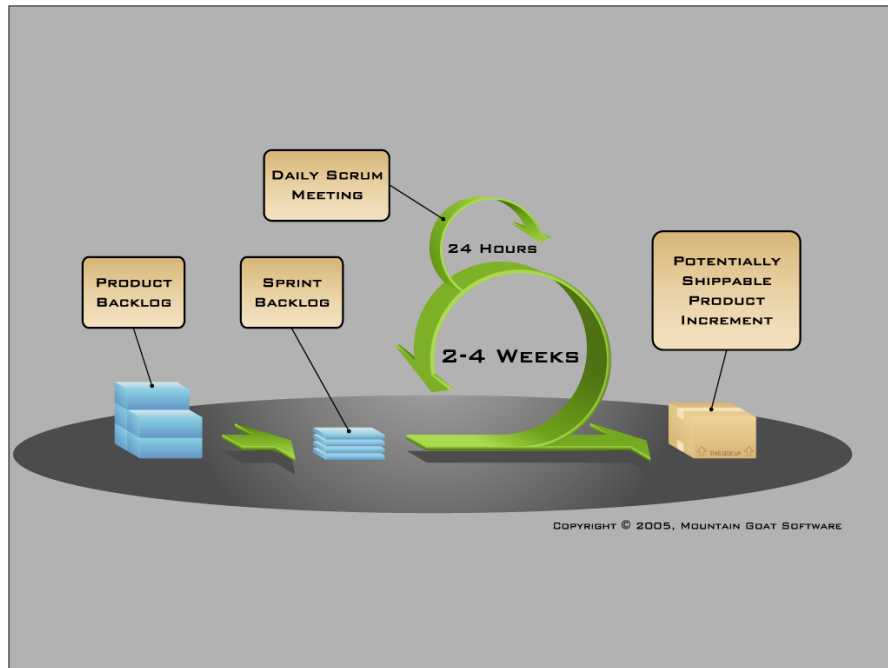


Figure 12: Basic process of Scrum

●**Scrum:** Scrum is an agile software development methodology, and it is an increment, iterative development process. The development process consists of a number of short iteration cycles named sprints (the green curly arrow). The basic process of scrum⁹ is described in Figure 12. In the scrum, product backlog is a prioritized list of requirements used to manage product requirements. The Scrum team in a sprint should select the highest priority requirement from product backlog to develop. Then the selected requirements are analyzed and evaluated at the Scrum meeting to get a task list to be completed in the sprint called sprint backlog. When a sprint finished, the product increment is handed [7]. The process shows that scrum needs the team works as tight, focus on a single goal, understands the priorities and team members are clear about their roles [8]. It is necessary for our team to work as the scrum team to develop the software. We set one sprint for a week, and finish a software requirement list as the product backlog. Our team has a short and everyday meeting to agree on a to-do-list, which will be released in a team collaboration tool named Tower as sprint backlog. In each sprint, our team will have a formal meeting with the supervisor to discuss the requirements selected and hand in product increment.

●**Tower:** Tower is a team collaboration tool which based on agile development. It

⁹<https://www.mountaingoatsoftware.com/agile/scrum/overview>

can be used for Scrum to release to-do-list in each sprint. It is like an online office where you can quickly process tasks, conduct discussions, review project progress, and work with your team at any time. According to Cockburn and Highsmith [9], agile development reduces the cost of exchanging information in a team and improves the team's amicability-members' sense of community. These points are perfectly implemented in Tower. For instance, all the tasks are displayed in the panel, and team members can access all the tasks to know about the details and progress, which is easy to exchange information. Moreover, when a task is finished or assigned, every team member will be reminded, which improves the efficiency of doing tasks. It is obvious that Tower can help a lot to develop the software, so we use it as our team collaboration tool. The link is <https://tower.im/>.

2. Development languages

●**Programming languages:** The programming language is the basic part of our software. According to a website for famous programming language ranking list named TOIBE, Java is the most popular programming language in the world [10]. Therefore, it has a huge amount of open source libraries to make our software better. Moreover, our group members are all familiar with Java. It is easier for us to develop a software. As a result, we decide to use Java as our major programming language. For the Graphical User Interface (GUI) part, we decide to use JavaFX, which is a new version GUI library for Java SE. By using JavaFX, we can also use the web language (HTML, CSS, and JavaScript) indirectly to make our interface more visual.

●**Scenebuilder JavaFX:** According to Jackson, SceneBuilder JavaFX¹⁰ provides a visual layout environment that allows us to quickly generate the main frame of the user interface for Java applications[11]. It supports a graphical interface control to simply drag and drop to a JavaFX scene. When one creates a user interface layout, FXML layout code will be automatically generated. SceneBuilder provides a simple and intuitive user interface that can help developers quickly create an interactive application prototype that connects GUI controls to the application logic.

●**External Library:** There is a Library for JavaFX which have some fancy and colorful components. This library is used in our user interface to help components more beautiful. jfoenix

3. Developing Tools

●**IDE:** We decide to use eclipse. We all familiar with it. It is also convenient for us to test our code using JUnit.

¹⁰<http://www.oracle.com/technetwork/java/javase/downloads/javafxscenebuilder-info-2157684.html>

- GitHub:** According to the technical research, our group decides to use GitHub as a basic tool to develop the software, so that we can upload the software project to one repository where everyone can access and work on it, control the project version in case conflicts happens, and search some useful open resource distributed in GitHub.

4. Platform and Operating systems

For the personal computer(PC) platform, the advantage is the computer has more storage space than other devices to install an application. It also has a larger display screen, which can make animation, the main function of our software, more clear and visual. Therefore, we can design more functions for PC software development.

For different operating system, according to StatCounter Global Stats, from 2015 to 2016, the number of users of the Windows system still possesses a high proportion of PC users [12]. This situation happens not only in China but also in the whole world. Therefore, we realize that most of the users still use the Windows system, so we decide to develop our software based on the Windows operating system.

5.2 Overview of the developed source code hierarchy

We use Model-View-Controller (MVC) architecture in the project. As the name indicates, the Model-View-Controller pattern suggests the division of an application into the following elements:

- **Model:** Contains the data model and all information that identifies the state of the application. It is generally self-consistent and independent of the other elements.
- **View:** Stands on the other side with respect to the model and defines the representation of the data stored in the model. The view is commonly identified as your application's user interface (or GUI) or, in the case of Web applications, the browser Webpage.
- **Controller:** Represents the application logic. Here, it is defined how the user can interact with the application and how user actions are mapped to model changes.

Since it's a project focus on animation, we do not have the model package.

5.3 Visualising Sorting Algorithms

The most important part of the software is the animation of sorting algorithms. The Bubble Sort animation will be as an example to explain how we implement the animations. The basic idea of sorting algorithm animations is adding transitions to the swap function.

The gap of rectangles is fixed at the initial step using `HBox`.

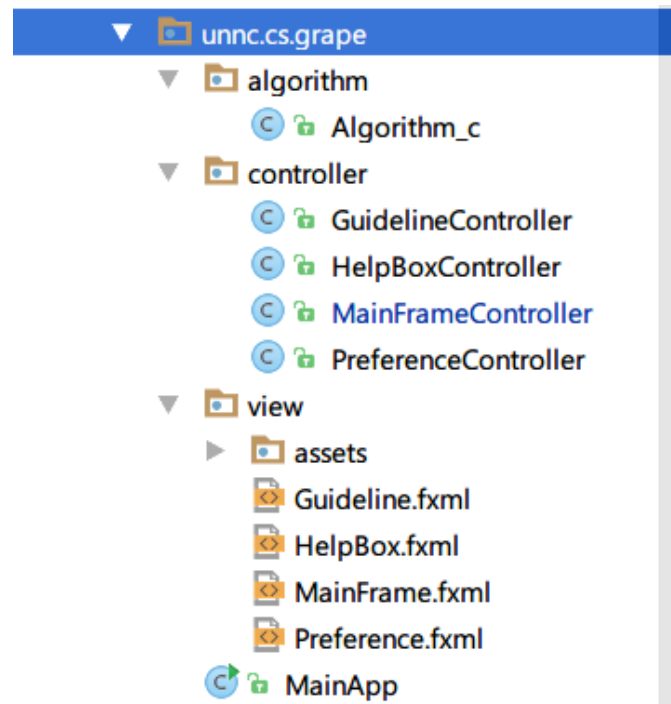


Figure 13: Code hierarchy

```

private SequentialTransition BubbleSort(int arr[], ArrayList<StackPane> list,
double duration) {
    SequentialTransition sq = new SequentialTransition();
    int temp;
    for (int i = 0; i < arr.length - 1; i++) {
        for (int j = 1; j < arr.length - i; j++) {
            if (arr[j] < arr[j - 1]) {
                temp = arr[j - 1];
                arr[j - 1] = arr[j];
                arr[j] = temp;
                sq.getChildren().add(swap(list.get(j - 1), list.get(j),
                    list, duration));
            }
        }
    }
    return sq;
}

```

This is the function of bubble sort animation. Firstly, we use `SequentialTransition()` to make sure each action of the animation should act as a sequential order. The action order is added into the swap part to make rectangles move correctly.

```

private ParallelTransition swap (StackPane l1, StackPane l2,
    ArrayList<StackPane> list, double speed) {
    TranslateTransition t1 = new
        TranslateTransition(Duration.millis(speed), l1);
    TranslateTransition t2 = new
        TranslateTransition(Duration.millis(speed), l2);
    ParallelTransition pl = new ParallelTransition();
    t1.setByX(30);
    t2.setByX(-30);
    pl.getChildren().addAll(t1, t2);
    Collections.swap(list, list.indexOf(l1), list.indexOf(l2));
    return pl;
}

```

As it shown, it is swap () function. Using two TranslateTransition() to move the rectangles which need to be swapped. 30 is the gap size of two adjacent components. Using the method SetByX() of TranslateTransition () to set the moving distance on X-axis. Then add the two transitions into ParallelTransition () to make the two swapping actions (transition) happen at the same time.

5.4 Implementing Graphical User Interface

5.4.1 Single Algorithm Page

The MainFrame file designs the main interface of our software. Different components are used to implement different functions. the menu bar component contains different menus inside. A Tab pane component is used to change the pages between Single Algorithm and Comparison, which makes the page swapping more convenient and efficient. In single algorithm page, our software uses several toggle buttons to choose the algorithms. For each toggle button, a ImageView component is included to show the overview picture of the corresponding algorithm. Each button also has a mouseEnter function and a mouseExit function, which means that when the cursor moves on the toggle button, the gif picture will animate, and be static if the cursor leaves. The animation block uses a hbox component to generate the image. The progress bar and speed bar are both made by slider components. Both of the initialize button and clear button use JFXbutton components. The play/pause button and random button also use button components and imageView components inside. The input field consists of a JFXTextField. On the right side of the interface, two textArea components are used for the Algorithm Hint block and Algorithm Code block. In comparison page, the software uses two JFXcomboBox to choose the algorithms, two hbox to show the animation and one LineChart component to show the complexity diagram. Below will show how the time slider works.

```

public void timeChange() {
    timeSlider.valueProperty().addListener(e -> {
        if (timeSlider.isValueChanging() && timeSlider.getValue() != 0) {
            st.pause();
            st.playFrom(st.getTotalDuration().multiply(timeSlider.getValue()
                / 100));
            if (playbutton.getId() == "replaybutton") {
                st.pause();
            }
        }
    });
}

private void updateProgress() {
    Platform.runLater(() -> {
        Duration currentTime = st.getCurrentTime();
        Duration duration = st.getTotalDuration();
        if (!currentTime.equals(duration)) {
            timeSlider.setValue(currentTime.divide(duration.toMillis()).toMillis()
                * 100);
        } else {
            timeSlider.setDisable(true);
            inputString.setDisable(true);
            st.stop();
        }
    });
}

```

The `timeChange` method is the On-Drag function of time slider, it add a `Listener` on `valueProperty`, when the slider's value changes, it will pause the animation and play from the new value from the slider. The `updateProgress` aims to move the slider with the animation. Every time when the animation's current time changed, it will also calculate a new value and set the slider to the proper position. To make it work, you should add a `Listener` to the animation's `CurrentTimeProperty`.

5.4.2 Comparison Page

The interface of Comparison part is also designed by `MainFrame` file. This part is used to show the efficiency of two different sorting algorithms. The interface is divided into 4 parts. The first part is the animation display part, which has two windows, each window has one `HBox` and two `Labels`.

The second part is the input part which has an input area and a `START` button. After inputting a fixed format input and clicking the `START` button, two `HBox` generate with the same frame and start to run different animations of what users have chosen, the time of the animation will also be shown under the `HBox`.

- "Get Input" function

Get input size from the text area and generate a random sequence of integers given the size.

```
private int[] getInput() {
    .....
    for (int i = 0; i < input_c.length; i++) {
        randomNum = ThreadLocalRandom.current().nextInt(1, in[0]);
        input_c[i] = randomNum;
    }
    .....
    return input_c;
}
```

- "Start Compare" function

First generate rectangles in every HBox, according to the algorithms chose by user, play the animation.

```
public void startCompare() {
    ...
    // generate rectangles
    ArrayList<Animation> l1 = new ArrayList<Animation>();
    ArrayList<Animation> l2 = new ArrayList<Animation>();
    ...
    showTime(l1, duration_c, time_left);
    ...
    showTime(l2, duration_c, time_right);
    SequentialTransition sq = new SequentialTransition();
    sq = playTwoAnimate(l1, l2);
    sq.play();
}
```

The third part is the algorithm choosing part, which contains two ComboBox to let the user choose different algorithms, when user choose the algorithms, the algorithms' name will appear on the animation display area.

- "Choose Algorithm" function

Choose two sorting algorithms at the right part of the frame, a label of algorithm name will be shown.

```
public void chooseAlgo() {
    compareAlgo =
        combo.getSelectionModel().getSelectedItem().getAccessibleText();
    label.setText(combo1.getSelectionModel().getSelectedItem().getText());
    AlgoComplex.setText(combo.getSelectionModel().getSelectedItem().getText());
}
```

The last part is "Complexity" part, which will generate a chart to show the complexity of two algorithms which has been chosen by users.

5.4.3 Menu bar

The user interface design of menu bar is implemented in `MainFrame.fxml`. In the file, `MenuBar` is used to place three `Menu` which are "File", "Function" and "Help". In `Menu` "File" and "Help", there are several `MenuItem` representing different functions. In `Menu` "Function", there is another `Menu` in which group of `RadioMenuItem` is used to indicate users can only choose one option. Each function in menu bar is implemented in `MainFrameController.class`, in the following part, two important functions in menu bar will be described how to implement them, the first one is "Save Code" function in "File" menu, and the second one is "Code Language" function in "Function" menu.

- "Save Code" function

When users click the function, method `handleSaveCode` in `MainFrameController.class` will be invoked first.

```
@FXML
private void handleSaveCode() {
    mainapp.SaveAlgorithmCode(selectAlgo, languageSelect);
}
```

Method `SaveAlgorithmCode` is then invoked as the above code shows. Parameter "selectAlgo" means the algorithm users select, and "languageSelect" means the code language users choose. This method is implemented in `MainApp.class`.

```
public void SaveAlgorithmCode(String algo, String language) {
    .....
    FileChooser.ExtensionFilter("TXT files (*.txt)", "*.txt");
    filechooser.getExtensionFilters().add(extFilter);
    File file = fileChooser.showSaveDialog(primaryStage);
    String code=null;
    .....
    String fileName = "./code/" + algo + language + ".txt";
    .....
    code = new String(Files.readAllBytes(Paths.get(fileName)),
        StandardCharsets.UTF_8);
    .....
    fileWriter = new FileWriter(file);
    fileWriter.write(code);
}
```

When method `SaveAlgorithmCode` is invoked, first, it will show a dialog for users to type a name and choose a location to save, and the default extension is TXT file. Then the corresponding algorithm code which users choose will be read from

the files in `code` package, and the code will be written to a `fileWriter`. Finally, the code will be saved when users click "Save" button in the dialog.

- "Code Language" function

The function provides five programming languages to choose, and we use "C language" as an example. When users click "C", the method `handleC` implemented in `MainFrameController.class` will be called.

```
@FXML
private void handleC() {
    languageSelect = "C";
    displayCode(languageSelect, selectAlgo);
}
```

In method `handleC`, variable `languageSelect` will be assigned to the value "C" which is the language users choose, then the method `displayCode` will be invoked. The parameters "languageSelect" and "selectAlgo" have the same meaning as in "Save Code" function.

```
private void displayCode(String language, String algo) {
    String fileName = "./code/" + algo + language + ".txt";
    try {
        codeDisplay.setText(new
            String(Files.readAllBytes(Paths.get(fileName)),
                StandardCharsets.UTF_8));
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

When method `displayCode` is invoked, first it will read the corresponding algorithm code from the files in `code` package, then it will use `setText` method, which can refresh the content of `TextArea`, to display the code in algorithm display block.

5.4.4 Help box

The `HelpBox` is implemented in `HelpBoxController.class`, which shows the information of our software and developer. It use an `imageView` component to show our logo and a button ok to confirm and quit the dialog. Other word information is included in the `Text` component.

5.4.5 Preference

The `Preference` is implemented in `PreferenceController.class`, which shows the interface of color choosing function. It uses a `JFXColorPicker` component to choose the color for generated image and a button `Apply` to confirm and quit the dialog.

5.4.6 Guideline

The `Guideline` is implemented in `GuidelineController.class`, which designs the guide page of our software. It uses a `TabPane` component to change the guide pages between Single Algorithm and Comparison. It also has a button ok to confirm and quit the dialog.

6 Testing

In this chapter, the testing of the application will be involved. For all the function requirements, we test them by using Black Box Testing. In addition, the User Interface Testing is used to test the user interface (UI) of the software. Otherwise, the future works are shown in the following part, which includes the introduction of the incomplete functions and the functions that can be achieved in the future.

6.1 Black Box Testing

To check whether the application has corresponded with the functional requirement, the black box testing has been used. Black box testing is a testing way which only examines application's functionality, it does not peer into the internal part of the application otherwise. By doing the black box testing, the following errors could be found:

- Incorrect or missing functions
- Interface errors
- Initialization and termination errors

The table below shows the test process and the result of this application's black box testing.

System Test Sheet			
Project Title	AlgoDomino		
Objective	To test whether the system meets the functional requirements.		
Method	Black Box Testing		
Test Environment	CPU: Intel (R) Core (TM) CPU i5-4210M @ 2.60GHz RAM: 8.00 GB Operating System: Microsoft Windows 10 Professional 2017		
Test Progress			
Requirement to the test	Action	Expected Result	Y/N
The software should provide users a guide of how to use core features.	Help - Domino Help	A window pops up which shows the guide of this software	Y
The software should allow users to select which sorting algorithms to animate.	Click Single Algorithm (under the menu bar)- Click the sort button	After clicking, the button generates a blue border, and the background color changed	Y
The software should be able to allow users to define input data by themselves.	Input a sequence of integers following the provided format	The text area shows what user has input	Y
	Click CLEAR button	Empty the input area	Y
The software should be able to generate input data randomly.	Click RANDOM button (represent by a "dice")	A sequence of numbers appear in the input text area	Y
The software should be able to visualize the processes of running a sorting algorithm.	Click INITIALIZE button	The animation board generates a sequence of rectangles with text of numbers, which the rectangles' height and the numbers are connected with the input	Y
The software should allow users to start, stop, slow down, speed up, restart or pickup a point of the animation.	Click START/PAUSE button	Animation starts to run, the START/PAUSE button change to from START to PAUSE, the input area cannot input integers at this moment, the time slider start to move	Y
	Click START/PAUSE button	Animation paused, the START/PAUSE button change from PAUSE to START, the time slider stop to move	Y

	Drag SPEED SLIDER to change the speed of the animation	The speed of the animation changed	Y
	Drag TIME SLIDER to change	The animation changed	Y
	Click NEXT STEP button (left side of the START/PAUSE button)	The animation moves to next movement	N
	Click LAST STEP button (right side of the START/PAUSE button)	The animation moves to the previous movement	N
The software should be able to show the running sorting algorithm's source Java code to users.	Click START/PAUSE button	The code shows on the Algorithm Code part, which is in the right part of the interface	Y
The software should be able to match specific line of running sorting algorithm's source code and its explanation with animation.	Click START/PAUSE button	In the Algorithm Code part, the specific line of code is highlighted	N
		In the Algorithm Hint part (above the Algorithm Code part), the explanation of the sorting algorithm appeared	Y
The software should allow users to select which algorithms to compare.	Click Comparison button (between the Single Algorithm button) - Click to choose two algorithms (right side of the interface)	Two algorithms has been chosen, two labels show the name of the algorithms on the animation window and complexity window (under the algorithm choosing part)	Y
The software should compare the efficiency of different sorting algorithms by showing the animation simultaneously.	Input integer's size following the input area's default format - Click START button	The animation window generates two groups of rectangles and start to run different animations depends on what user choose	Y
		The algorithm's running time will be shown under each single animation window	Y

The software should be able to provide the time complexities of sorting algorithms when comparing the efficiency.	Click START button	A chart will be shown in the Complexity window (right part of the main interface) according to the times of these two algorithms	Y
The software should allow users to customize the interface, such as the background color of animation, shapes, and colors of basic components used in an animation.	Click Function button - Click Color of Chart - Select a color - Click Apply button	The color of rectangles are changed	Y
Users should be able to download the source code of selected sorting algorithms in several programming languages.	Click File button on the top bar - Click Save Code	Save the code of the algorithm	Y
Users should be able to save the screen-shot of the current interface	Click File button - Click Save Screen-Shoot	Save the algorithm animation as screen-shot	Y

6.2 Evaluation and Discussion

6.2.1 User Interface Testing

Because the black box test can not test the functions without code, so we need to change the way of testing the GUI part.

When open the software, the screen of the software is validated, nothing lost and miss. For the navigation, all the button have a react after clicking it. When input something in the input area, the text area shows the input smoothly. When using the software, all the manipulations are reacting without delay.

6.2.2 Future Work

To enhance the usability of the software, there are also some things we would like to do. Firstly, the User Interface (UI) design have lifting space. We make the window of the software as fixed size, it can be improved to resizable, and the inner windows' size will also be changed automatically. In addition, the shape of components of animation need to be changed by users in the future, which can fit the user center model and make users feel more comfortable to see the animation.

In terms of skill improvement, we want to increase the number of algorithms for the animation part which means people can learn more algorithms from our software. More-

over, the highlight of the code which needs to synchronize with the sort animation is not achieved because of the limitation of the coding language. If there are some technicals to solve it, it is better for users to learn and we are willing to make it.

Finally, it is more convenient for users to have multiple platforms to operate the system. People can learn sorting algorithms everywhere by smartphone or pad. So create a website which contends all the functions of this project is a good choice.

7 Reflective Comments

7.1 Problems Encountered and Solutions

1. Technical problem

Since this is the first time we try to develop a software, we are lack of some vital skills for this project. Therefore, we try to find some upperclassmen to ask how to plan our project. Then we find the agile development method is suitable for us. Based on this method, we have an appropriate plan of the project.

In addition, we do not learn GUI before. It is difficult for us to make animations. We choose JavaFX as our main language. However, there is a little tutorial of JavaFX. It is very hard for us to code at the first stage. Then we are more familiar with this language by checking API and learning from other people.

2. Group roles

As a new team builds up, we do not know each other. It is difficult for a team leader to know teammates' strengths and weaknesses in a short time. Thus we do not have a clear group role for everyone originally, and it causes some trouble in assigning tasks. At the third week of the first semester, we distributed our roles of the project. It is very useful for us to distribute tasks.

3. Task overdue

Sometimes tasks were overdue for some reasons, and it delays the progress of the project. Therefore, to prevent this problem, we try to use a team collaboration software named tower to manage and distribute our tasks. It can record tasks, their implementers and due days. If someone does not finish the task in time, it would remind us. It is extremely efficient to solve this problem. Our members can get motivated by this method.

4. Missing stakeholders

Our project does not have any stakeholders; we need to figure out all requirements by ourselves. To make the requirements clearer, we find some people to ask for suggestions. The supervisor also gives us many useful suggestions based on her experience.

5. Inappropriate Time Schedule

It is very difficult to make a perfect plan. Although we plan for our project, some problems break the plan. Firstly, there are some functions we cannot achieve in time. It is very time-consuming to fix bugs. secondly, at the former stage, we do not concentrate on the most important part. We try to add more functions for the software, but ignore to make the better quality of animations.

7.2 What we have learned

Firstly, it is very happy to have the chance to work with a group. We have learned how to do a project more efficiently. We are more familiar with the process of software engineering. It is a good experience for future projects. We also know how to use tools to help us such as using GitHub to track code.

Secondly, the skill of coding is increasing. Although we all did not use JavaFX before, we can use it make a software in a short time. It is very amazing based on our level because we did not have the experience to work with a software engineering project before.

Thirdly, we know the importance of collaboration. If the team members do not help each other, we will not finish the project. Our members are all pleased to share their knowledge to help each other.

8 Conclusion

Overall, we think the project was a success, as it solves the problem asked and meets all the main requirements.

The project had originally aimed to implement the functionality for the user to be able to change shapes of animation and more interesting functions, however as the project went on, it became ever more apparent how hard this feature would be to implement and the time is limited, instead of half-implementing it, it was decided to focus on making the basic features of the system stronger.

In summary, this project achieved the aims that we were required. We believe that the software can help students study the sorting algorithms easier.

9 Appendix

9.1 Meeting Minutes

Meeting: 26 Sep. 2016

Student: Zhefeng ZHOU, Yangyu GAO, Mui JIANG, Jiaying SUN, Kan LIU, Zhe REN

Supervisors: Heshan DU

Aim:

To discuss how to have a head start of our project.

Agenda:

1. Background research
2. Java GUI programming
3. Algorithm in prototype

Discussion:

1. Background search

We discussed two ways to do background search, one is issuing questionnaire and other one is searching similar apps via Internet and checking their features.

2. Java GUI tutorial

We discussed how to make a head start of Java GUI programming, and decided to start from the textbook and produce simple demo to get familiar with related packages.

3. Algorithm in prototype

We discussed which sorting algorithm to start with when developing the prototype. Possible options include bubble sort, selection sort and insertion sort, well learn a little bit and discuss it before 30 Sep. 2016.

Action Points

1. Set up the website for the project
2. Design and issue the questionnaire, and send a draft to Heshan
3. Start to prepare the first version of requirement specification
4. Produce simple java programs with GUI
5. Decide which algorithm would be used in the prototype, learn and implement it in java code
6. Start to learn Latex, get the software download and installed on the computer

Next Meeting: 10 Oct. 2016

Meeting: 10 Oct. 2016

Student: Zhefeng ZHOU, Yangyu GAO, Muiji JIANG, Jiaying SUN, Kan LIU, Zhe REN

Supervisors: Heshan DU

Aim:

To discuss methods to specify our jobs of this project.

Agenda:

1. Website template
2. Draft of questionnaire
3. Requirement specification
4. Report

Discussion:

1. Website template

We planed to use a web page template to create our website. This template contains a timeline as a clue, which is convenient to record the progress of our project.

2. Draft of questionnaire

Firstly, we discussed the context of the questionnaire template. Some questions of it are not suitable, so we decided to make questions more specific and focus on smaller group like classmates.

Then, we discussed research checklist. There are some problems need to be revised, such as the relevance of secondary data sets. In addition, we knew there is some jobs of making a questionnaire, such as attaching information sheet and assigning identified id number, etc.

3. Requirement specification

We discussed which sorting algorithm to start with when developing the prototype. Possible options include bubble sort, selection sort and insertion sort, well learn a little bit and discuss it before 30 Sep. 2016.

4. Report

We discussed that when we should start to write interim report. We decided to accomplish a draft of interim report two weeks earlier than the deadline to have enough time to revise.

Action Points

1. Prepare website
2. Try to solve the problems of questionnaire
3. Prepare draft of requirement specification
4. Keep learning Java GUI programming and LaTeX
5. Consider the date of writing report

Next Meeting: 17 Oct. 2016

Meeting: 17 Oct. 2016

Student: Zhefeng ZHOU, Yangyu GAO, Muiji JIANG, Jiaying SUN, Kan LIU, Zhe REN

Supervisors: Heshan DU

Aim:

To discuss methods to specify our jobs of this project.

Agenda:

1. Website fix
2. Questionnaire cancelation
3. Requirement specification
4. Software platform
5. Dissertation view

Discussion:

1. Website fix

We finished our website page temple and there remained some small changes to be made. We would complete it before the next formal meeting.

2. Questionnaire cancelation

We decided to cancel the questionnaire due to the fact that its not suitable for our project.

3. Requirement specification

We viewed part of the first version of our requirement specification. Some details would be added and changes would be made during our software development process.

4. Software platform

We havent decided which platform to show our software on and we discussed previous year students choices of software platforms. Most of them chose windows only. The decision would be made after our group meeting this week.

5. Dissertation view

We picked and scanned some dissertations that are helpful for the preparation of our report.

Action Points

1. Make some changes of our website
2. Accomplish requirement specification as the project develops
3. Decide the platform of project and prepare for it.
4. Study the related dissertation and prepare for our report.
5. Keep learning Java GUI programming and LaTeX

Next Meeting: 24 Oct. 2016

Meeting: Meeting: 7 Nov. 2016

Student: Zhefeng ZHOU, Yangyu GAO, Muiji JIANG, Jiaying SUN, Kan LIU, Zhe REN

Supervisors: Heshan DU

Aim:

To discuss methods to specify our jobs of this project.

Agenda:

1. Requirement Specification V3
2. Website design
3. GUI design
4. Suggestion on system architecture

Discussion:

1. Requirement Specification V3

We decided to modify the Introduction part to make the requirement specification fit in interim report.

We decided to make some user stories and UML design, then put them in a motivation part in interim report rather than requirement specification part.

2. Website design

We decided to improve the home page of website in the following two parts

1) The homepage should introduce the software and its main function. 2) The homepage should explain why our group is called Grape. We decided that the function of comment is not necessary.

3. GUI design

We decided to improve the mainframe part

1) The choosing algorithm button should show the full name of sort algorithm. 2) The input box should prompt users the format of input. 3) The progress bar should be under the bar graph. 4) We will decide where to put the Random generate button.

We decided to improve the comparison part

1) We will reduce the number of choosing algorithm button to 2. 2) We will show the complexity of two algorithms in one coordinate system. 3) We will consider how to design the Stop button. 4) We will consider the relation between input size and complexity in coordinate system.

4. Suggestion on system architecture

We decided to focus on one algorithm and try to implement its main function.

Action Points

1. Improve requirement specification
2. Improve website homepage
3. Improve GUI design
4. Prepare the interim report
5. Learn the user interaction model
6. Make some user stories and UML design

Next Meeting: 14 Nov. 2016

Meeting: 14 Nov. 2016

Student: Zhefeng ZHOU, Yangyu GAO, Muiji JIANG, Jiaying SUN, Kan LIU, Zhe REN

Supervisors: Heshan DU

Aim:

To discuss the progress of our project

Agenda:

1. Stakeholder issue
2. Website fix
3. GUI feedback
4. Interim report structure

Discussion:

1. Stakeholder issue

Since we cannot find classmates and supervisors as our stakeholders, we decided to find tutors or PhD students. Our supervisor recommended Paul as our stakeholder. We should prepare consent form and information sheet for each stakeholder. In addition, we decided that the number of stakeholders is 3.

2. Discuss the interface improvement

We have added some components to our website, such as the introduction of our project and roles. Then, we decided to make the role of our project more specific.

3. GUI feedback

Firstly, We talked about the GUI of the mainframe. The supervisor suggested us to combine the speed bar and progress bar. So we would consider this later. We also decided to make the data input part without comma when users input data and make some changes of button and code parts size.

Secondly, we discussed the efficiency frame of GUI. Some details of words should be changed.

In addition, we decided to use javascript to improve our GUI.

4. Interim report structures

We wanted to add Motivation to the Background Research section. Then we discussed the location of it. We also decided to change the section name from Prototypes to Implementation.

If we want to add interviews of stakeholders to the appendix of our report, we need to prepare consent form to get permissions of our stakeholders.

Action Points

1. Improve the interface
2. Decide choosing stakeholder or test user group
3. Implement the function of tool bars

Next Meeting: 21 Nov. 2016

Meeting: 21 Nov. 2016

Student: Zhefeng ZHOU, Yangyu GAO, Muiji JIANG, Jiaying SUN, Kan LIU, Zhe REN

Supervisors: Heshan DU

Aim:

To discuss the progress of our project

Agenda:

1. Stakeholder issue
2. GUI feedback
3. Interim report
4. UML feedback

Discussion:

1. Stakeholder issue

We have talk about this question with Danny, a PHD student in our school, but he is too busy to do some other works. Therefore, we would like to ask some other people about the stakeholder, such as professor Eugen, tutor Manish and some other PHD students. Since Paul has agreed to be our stakeholder, we only need to find two stakeholders more.

2. GUI feedback

Since there are too many coursework in last week, we didnt do too much about report. But we already have a specific division of labor. We will start to write the draft of report during this week.

3. Interim report

Since there are too many coursework in last week, we didnt do too much about report. But we already have a specific division of labor. We will start to write the draft of report during this week.

4. UML feedback

We have made a UML diagram draft, but it still has some problems in small parts. The meanings of different arrows are not very clear.

Action Points

1. Deal with issues of stakeholders
2. Improve GUI
3. Prepare the draft of interim report
4. Improve UML model
5. learn javascript more

Next Meeting: 15 Dec. 2016

Meeting: 15 Dec. 2016

Student: Zhefeng ZHOU, Yangyu GAO, Muiji JIANG, Jiaying SUN, Kan LIU, Zhe REN

Supervisors: Heshan DU

Aim:

To discuss the interim report

Agenda:

Discuss and improve the first version of interim report

Discussion:

1. Common academic writing mistakes

We should avoid using emotional word like "excellent" and "beautiful".

We should avoid using abbreviation like "can't", "it's" and "don't".

We should avoid using personal pronoun like "you", using "a" or "the" instead.

2. Common mistakes in formatting

We should use footnotes to provide a link for pictures.

We should use citation at the last of sentence and place a space before citation.

We should not use nouns "user", "people" and "student" to indicate one kind of person. Choose one to use.

We should not use nouns "application", "software" and "system" to indicate the project. Choose one to use.

3. The detail of content that needs to change

The detail is described in the paper version of interim report which Heshan gave to us after she commented on it .

Action Points

1. Improve the interim report 2. Finish and hand in the interim report before December 16th

Next Meeting: No next meeting

Meeting: 21 Feb. 2016

Student: Zhefeng ZHOU, Yangyu GAO, Muiji JIANG, Jiaying SUN, Kan LIU, Zhe REN

Supervisors: Heshan DU

Aim:

To check the progress and make a plan for this semester

Agenda:

1. Discuss having a test user group
2. Discuss the interface improvement
3. Make a plan for this semester

Discussion:

1. Discuss having a test user group

Last semester we decide to find several PhD students or tutors to be our stakeholders. Now we think having a test user group is also a good way to improve our project, so we will decide to choose one of them as soon as possible.

2. Discuss the interface improvement

The tool bar should be more detailed and specific, for example, the meaning of bars—File, Run and Edit are ambiguous, users cannot easily know which part the bars are used for. And the Select bar should be corresponding to Single algorithm visualization page, meanwhile the Compare bar should be corresponding to Comparison between algorithms page. In addition, we need to implement the specific function of every tool bar.

The part to select sorting algorithms should be simple to operate and obvious, so we decide to put it on the left side of main page.

The speed control bar is a little bit short, so we can make it like a vertical control bar used to control volume, or adjust the bar to be longer.

3. Make a plan for this semester

According to what we have done last semester, now we will focus on the PC version of our project, and we will try to make the software run normally before April. As for the near-term target, we will try to make several sorting algorithms' animation besides bubble sort.

Action Points

1. Improve the interface
2. Decide choosing stakeholder or test user group
3. Implement the function of tool bars

Next Meeting: 28 Feb. 2016

Meeting: 28 Feb. 2016

Student: Zhefeng ZHOU, Yangyu GAO, Muiji JIANG, Jiaying SUN, Kan LIU, Zhe REN

Supervisors: Heshan DU

Aim:

To check what we have done last week

Agenda:

1. Discuss the problem of the animation
2. Show the improvement of the user interface

Discussion:

1. Discuss the problem of the animation

Now there is a problem that the rectangles of the animation cannot work with the timeline (progress bar). Only two rectangles can move with timeline.

We discuss how to solve the problem and look for some possible methods to try. Then we decide to fix this problem this week.

2. Show the improvement of the user interface

We improve the start button like the format of the PowerPoint. To make the choice button of selecting the algorithms become the single buttons and move them to the right-hand place.

In addition, we also change the position of the input part

Action Points

1. Fix the problem of the animations timeline
2. Improve the user interface
3. Combine the animation and user interface and make some buttons work

Next Meeting: 7 Mar. 2016

Meeting: 7 Mar. 2016

Student: Zhefeng ZHOU, Yangyu GAO, Muiji JIANG, Jiaying SUN, Kan LIU, Zhe REN

Supervisors: Heshan DU

Aim:

To check the process of our project

Agenda:

1. The changes of user interface
2. The animation code
3. Combination of the user interface and animation

Discussion:

1. The changes of user interface

We use a new Javafx library to make some UI elements more fancy and colorful

2. The animation code

The insertion sort and bubble sort animations are achieved.

We decide to use transition rather than timeline in our code, because the timeline function cannot achieve playing animation continually.

3. Combination of the user interface and animation

The achieved algorithm animations are combined to the user interface. There is also some weakness like the position of the objects. It should be improved later.

Action Points

1. Improve the interface
2. Decide choosing stakeholder or test user group
3. Implement the function of tool bars

Next Meeting: 14 Mar. 2016

Meeting: 14 Mar. 2016

Student: Zhefeng ZHOU, Yangyu GAO, Muiji JIANG, Jiaying SUN, Kan LIU, Zhe REN

Supervisors: Heshan DU

Aim:

To check the process of our project

Agenda:

1. The changes of user interface
2. Some useful functions
3. The improvement of the algorithm animations

Discussion:

1. The changes of user interface

The play button has been changed.

The timeline of animation is deleted.

The edit function is also deleted.

2. Some useful functions

The functions of user guide and about us are added to the tool bar.

The color of rectangles in the animation can be changed by users choice. However, there is bug. When users choose the dark color, the numbers in the rectangles can not see clearly. It is necessary to find some methods to fix it.

3. The improvement of the algorithm animations

The selection sort animation has been added.

The height and size of rectangles of the animation have been adjusted.

Other algorithms animations are working by group members.

Action Points

1. Try to achieve more algorithm animations
2. Fix some bugs in the toolbar
3. Change the design of switching two interfaces

Next Meeting: 21 Mar. 2016

Meeting: 21 Mar. 2016

Student: Zhefeng ZHOU, Yangyu GAO, Mui JIANG, Jiaying SUN, Kan LIU, Zhe REN

Supervisors: Heshan DU

Aim:

To check the process of our project

Agenda:

1. The changes of user interface
2. The input function
3. The Merge sort animation

Discussion:

1. The changes of user interface

We use a new Javafx library to make some UI elements more fancy and colorful

2. The animation code

The insertion sort and bubble sort animations are achieved.

We decide to use transition rather than timeline in our code, because the timeline function cannot achieve playing animation continually.

3. Combination of the user interface and animation

The achieved algorithm animations are combined to the user interface. There is also some weakness like the position of the objects. It should be improved later.

Action Points

1. Improve the interface
2. Decide choosing stakeholder or test user group
3. Implement the function of tool bars

Next Meeting: 28 Mar. 2016

Meeting: 28 Mar. 2016

Student: Zhefeng ZHOU, Yangyu GAO, Muiji JIANG, Jiaying SUN, Kan LIU, Zhe REN

Supervisors: Heshan DU

Aim:

To check the process of our project

Agenda:

1. Fix some bugs of input function
2. Change the UI element of user interface
3. Heap sort and quick sort animation

Discussion:

1. Fix some bugs of input function

User can make random input by click the random button.

The input bug still has not been achieved. We will continue to do it.

2. Change the UI element of user interface

The play button has been changed more fancy.

3. Heap sort and quick sort animation

The Heap sort animation and quick sort animation have been achieved.

Action Points

1. Continue to achieve merge sort animation
2. Improve the input function
3. Fix the bug of size and height of animation
4. Add more related functions of animatin

Next Meeting: 11 Apr. 2016

Meeting: 11 Apr. 2016

Student: Zhefeng ZHOU, Yangyu GAO, Mui JIANG, Jiaying SUN, Kan LIU, Zhe REN

Supervisors: Heshan DU

Aim:

To check the process of our project

Agenda:

1. Discuss some problems of our process
2. Discuss the functions of our software
3. Discuss the final report outline

Discussion:

1. Discuss some problems of our process

There are some problems in the animation part. It is necessary to improve this part to make the animations more understandable.

There are about one month before the open day, so we should plan our time carefully.

Some possible bugs need to be considered, such as the speed bar.

2. Discuss the functions of our software

Highlighting code function should be added

We still have time to implement the efficiency part.

3. Discuss the final report outline

The outline is good. There are some sections need to be combined or edited.

The first draft of the final report need to be finished before the next Thursday.

Action Points

1. Fix some bugs of our software
2. Add some important functions to the software
3. Focus on the animations of algorithms
4. Start to write the first draft of the final report

Next Meeting: 18 Apr. 2016

Meeting: 18 Apr. 2016

Student: Zhefeng ZHOU, Yangyu GAO, Muiji JIANG, Jiaying SUN, Kan LIU,
Zhe REN

Supervisors: Heshan DU

Aim:

To check the process of our project

Agenda:

1. Fix some bugs of the software
2. Check final report outline

Discussion:

1. Fix some bugs of the software

We have fixed some bugs of the software, such as the

2. Check final report outline

The play button has been changed more fancy.

Action Points

1. Continue to achieve some functions
2. Write the first draft of the report

Next Meeting: 25 Apr. 2016

Meeting: 11 Apr. 2016

Student: Zhefeng ZHOU, Yangyu GAO, Muiji JIANG, Jiaying SUN, Kan LIU, Zhe REN

Supervisors: Heshan DU

Aim:

To check the progress of the GRP software and discuss the draft of the final report

Agenda:

1. Discuss some problems from interim report
2. Talk about the comparison part of the software
3. Discuss the draft of the final report

Discussion:

1. Discuss some problems from interim report

There are some parts of the feedback of the interim report we do not understand. Therefore, we discuss it and will change it later.

2. Talk about the problems of the software

The time of the comparison dose not achieve

We change the size of the input to prevent the large number of inputs.

3. Discuss the draft of the final report Move the algorithm part to the introduction part.

The design part of initial design and changes of updated design should be combined.

The second section of implementation part should be added more details.

Using the bubble sort code as an example to display how the it be implemented.

Visualizing sorting algorithm part. provide the main structure of the source code.

Refer to the .java

Section 5.3 Provide the reference of the library.

Section 5.4 Provide the source code hierarchy. (drawing the diagram)

Add readme to the code

Add the discussion the reason of why it does not changed.

Optional timeline

Add the footnote or describe of the () description of testing

Action Points

1. Improve the software
2. Improve the draft of final report

Next Meeting: No meeting

9.2 Sorting Algorithms Pseudocode

Algorithm 1 Bubble Sort

Input: *Array, size*

Output: Sorted array

```
1: function BUBBLESORT(Array, size)
2:    $i \leftarrow 0$ 
3:    $j \leftarrow 0$ 
4:   for  $i = 0 \rightarrow size - 1$  do
5:     for  $j = 0 \rightarrow size - 1 - i$  do
6:       if  $Array[j] < array[j + 1]$  then
7:         SWAP( $Array[j], Array[j + 1]$ )
8:       end if
9:     end for
10:  end for
11: end function
```

Algorithm 2 Insertion Sort

Input: *Array, size*

Output: Sorted array

```
1: function INSERTIONSORT(Array, size)
2:   for  $i = 1 \rightarrow size$  do
3:      $j \leftarrow i$ 
4:     while  $j > 0$  and  $Array[j - 1] > Array[j]$  do
5:       SWAP( $Array[j], Array[j - 1]$ )
6:        $j \leftarrow j - 1$ 
7:     end while
8:   end for
9: end function
```

Algorithm 3 Selection Sort

Input: $Array, size$ **Output:** Sorted array

```
1: function SELECTIONSORT( $Array, size$ )
2:   for  $i = 0 \rightarrow size - 1$  do
3:      $index \leftarrow i$ 
4:     for  $j = i + 1 \rightarrow size$  do
5:       if  $Array[index] > Array[j]$  then
6:          $index \leftarrow j$ 
7:       end if
8:     end for
9:     SWAP( $Array[index], Array[i]$ )
10:  end for
11: end function
```

Algorithm 4 Quick Sort

Input: *Array, size***Output:** Sorted array

```
1: function QUICKSORT(Array, size)
2:   var lessList, pivotList, greaterList
3:   if size ≤ 1 then
4:     return Array
5:   else
6:     select a pivot value pivot q
7:     for each x in q except the pivot element do
8:       if x < pivot then
9:         add x to lessList
10:      end if
11:      if x ≥ pivot then
12:        add x to greaterList
13:      end if
14:      add pivot to pivotList
15:    end for
16:    return concatenate(QUICKSORT(lessList), pivotList, QUICKSORT(greaterList))
17:  end if
18: end function
```

Algorithm 5 Merge Sort

Input: $Array, n$ **Output:** Sorted array

```
1: function MERGESORT( $Array, left, right$ )
2:    $result \leftarrow 0$ 
3:   if  $left < right$  then
4:      $middle \leftarrow (left + right)/2$ 
5:      $result \leftarrow result + \text{MERGE SORT}(Array, left, middle)$ 
6:      $result \leftarrow result + \text{MERGE SORT}(Array, middle, right)$ 
7:      $result \leftarrow result + \text{MERGER}(Array, left, middle, right)$ 
8:   end if
9:   return  $result$ 
10: end function
11:
12: function MERGER( $Array, left, middle, right$ )
13:    $i \leftarrow left$ 
14:    $j \leftarrow middle$ 
15:    $k \leftarrow 0$ 
16:    $result \leftarrow 0$ 
17:   while  $i < middle$  and  $j < right$  do
18:     if  $Array[i] < Array[j]$  then
19:        $B[k++] \leftarrow Array[i++]$ 
20:     else
21:        $B[k++] \leftarrow Array[j++]$ 
22:        $result \leftarrow result + (middle - i)$ 
23:     end if
24:   end while
25:   while  $i < middle$  do
26:      $B[k++] \leftarrow Array[i++]$ 
27:   end while
28:   while  $j < right$  do
29:      $B[k++] \leftarrow Array[j++]$ 
30:   end while
31:   for  $i = 0 \rightarrow k - 1$  do
32:      $Array[left + i] \leftarrow B[i]$ 
33:   end for
34:   return  $result$ 
35: end function
```

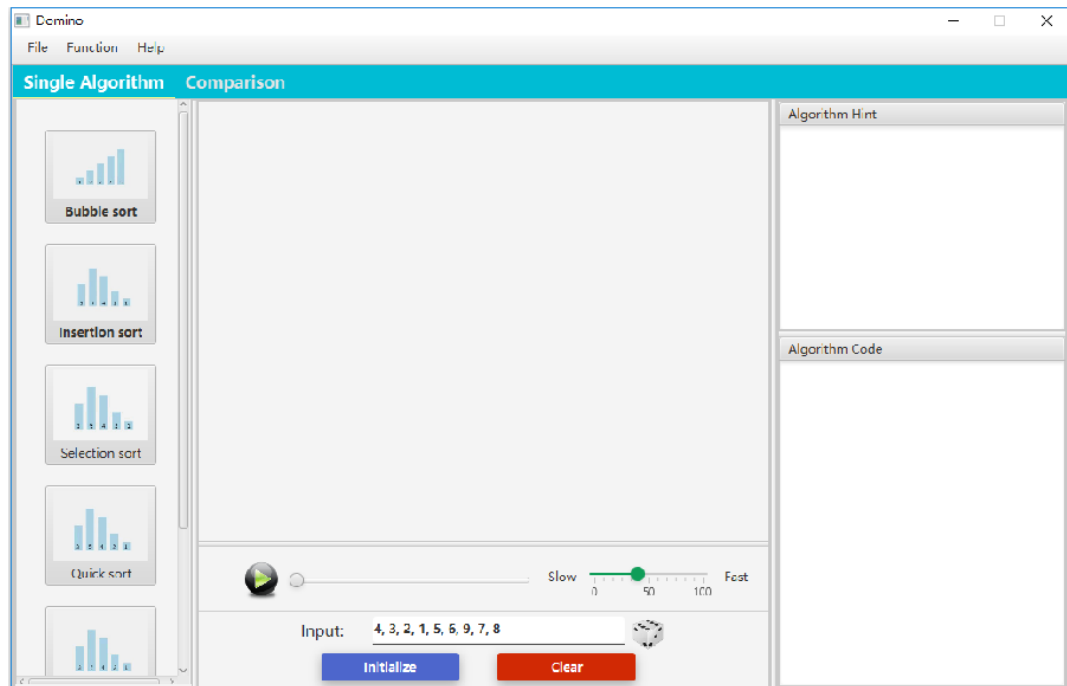
Algorithm 6 Heap Sort

Input: *Array, size***Output:** Sorted array

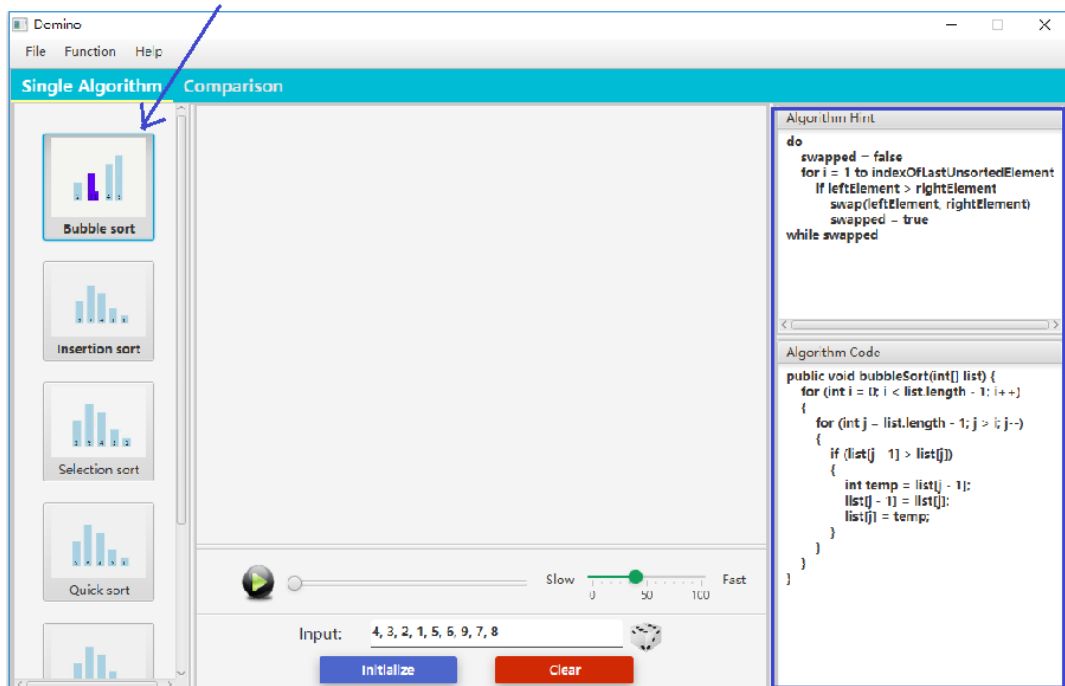
```
1: function HEAPSORT(Array, size)
2:   for  $i = size/2 \rightarrow 0$  do
3:     HEAPIFY(Array, i, size - 1)
4:   end for
5:   for  $i = size - 1 \rightarrow 1$  do
6:     SWAP(Array[ $i$ ], Array[0])
7:     HEAPIFY(Array, 0,  $i$ )
8:   end for
9: end function
10:
11: function HEAPIFY(Array, parent, size)
12:    $temp \leftarrow Array[parent]$ 
13:    $child \leftarrow 2 * parent + 1$ 
14:   while  $child < size$  do
15:     if  $child + 1 < length$  and  $Array[child] < Array[child + 1]$  then
16:        $child \leftarrow child + 1$ 
17:     end if
18:     if  $temp \geq Array[child]$  then
19:       break
20:     end if
21:      $Array[parent] \leftarrow Array[child]$ 
22:      $parent \leftarrow child$ 
23:      $child \leftarrow 2 * child + 1$ 
24:   end while
25:    $Array[parent] \leftarrow temp$ 
26: end function
```

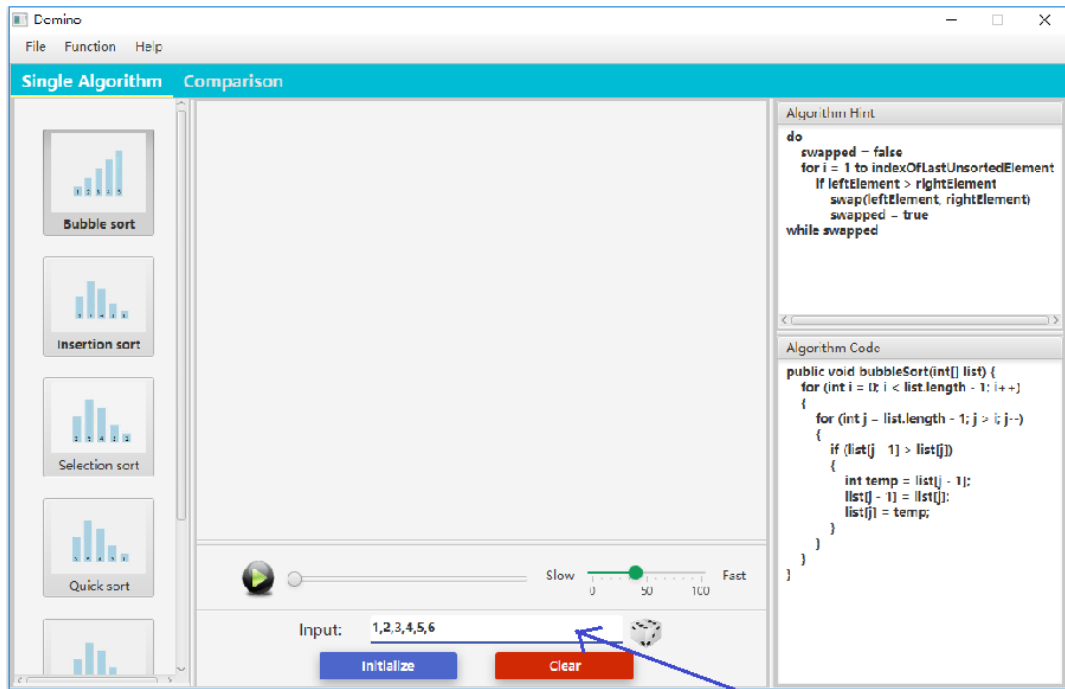
9.3 User Manual

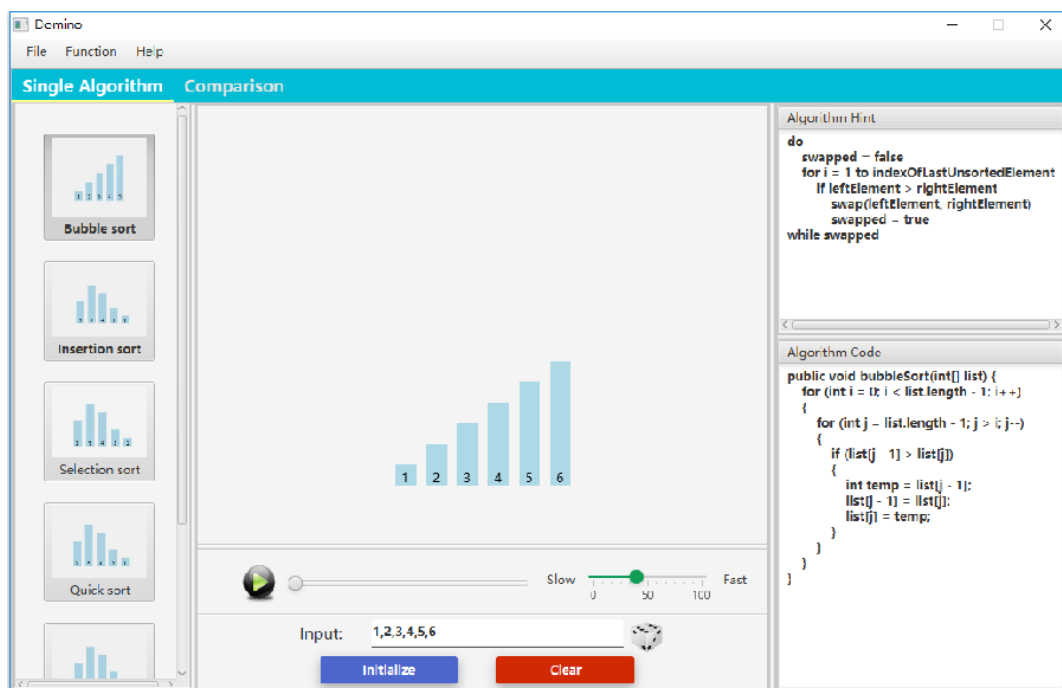
Single Algorithm Page



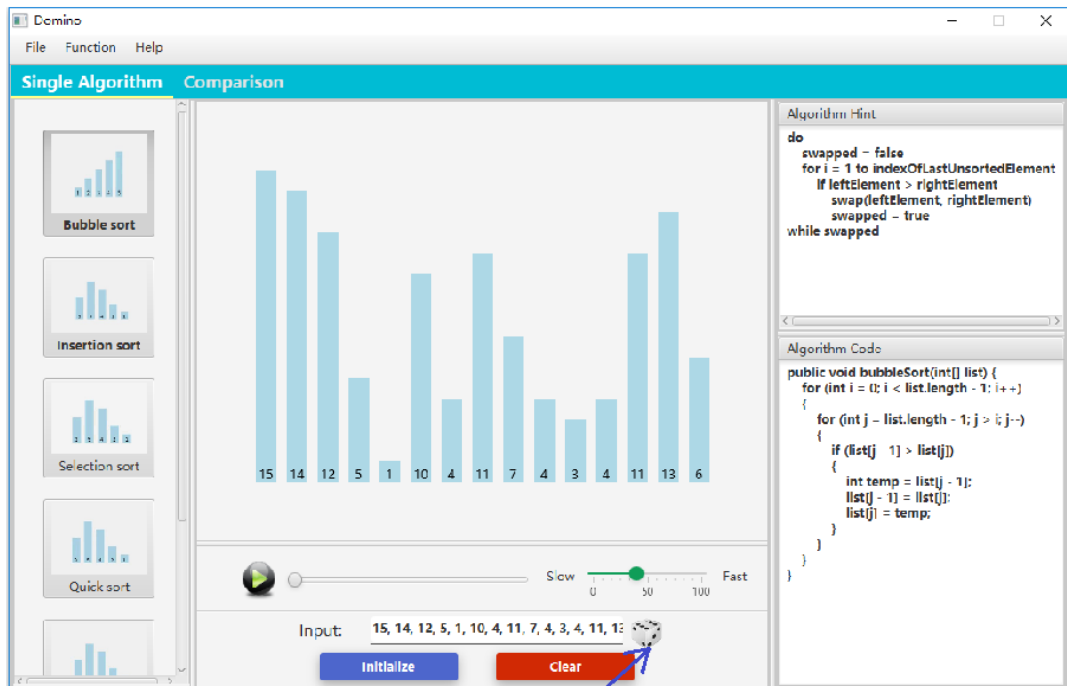
Select "Bubble sort" algorithm and corresponding code and hint display



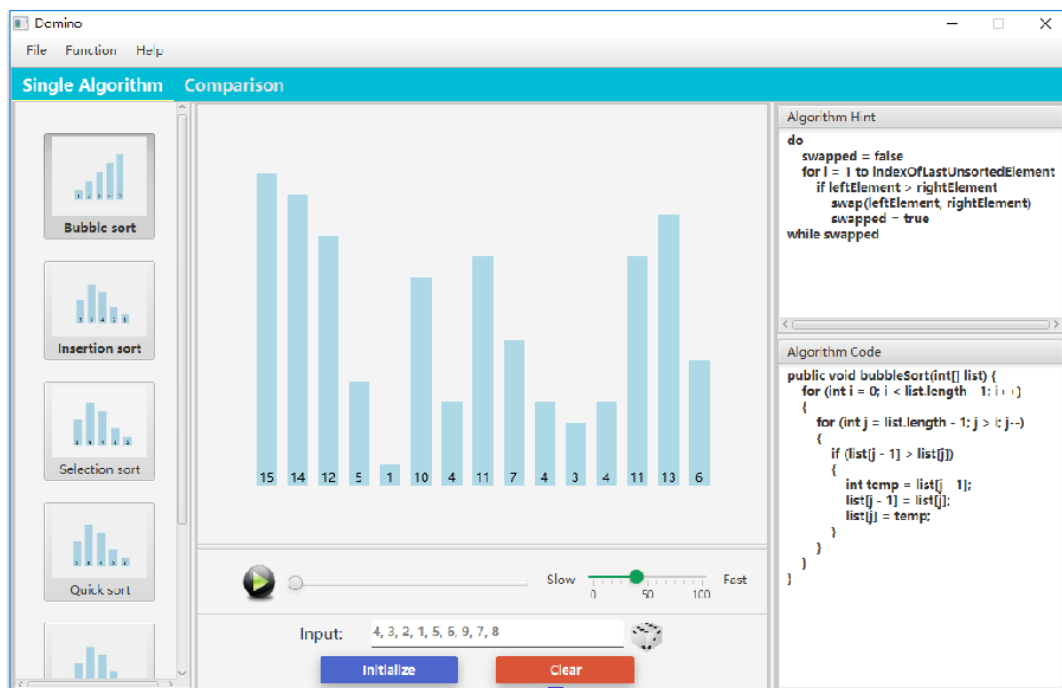




Click "Initialize" button to generate corresponding input chart

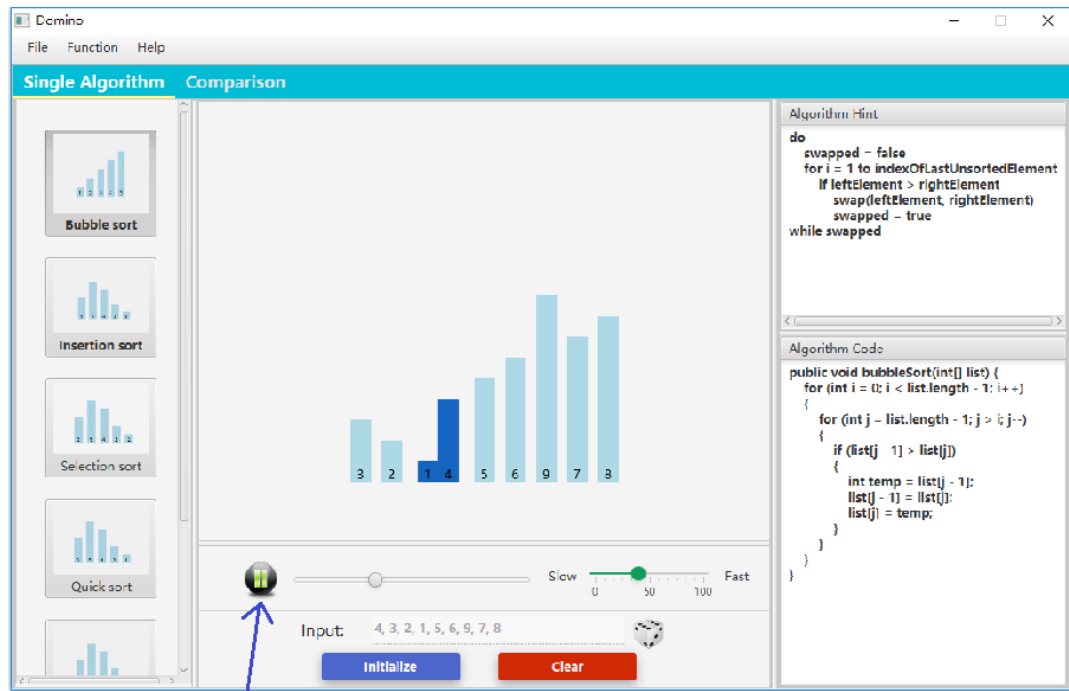


Or click "Random" button to generate and initialize random input

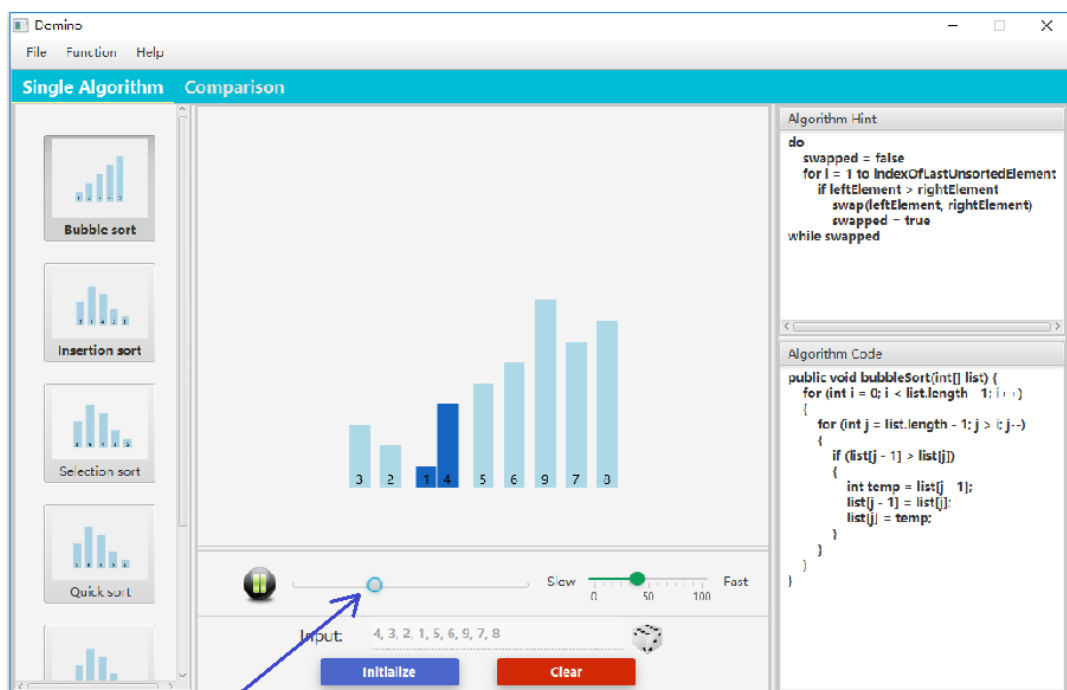


"Clear" button is used to clear the value in input field

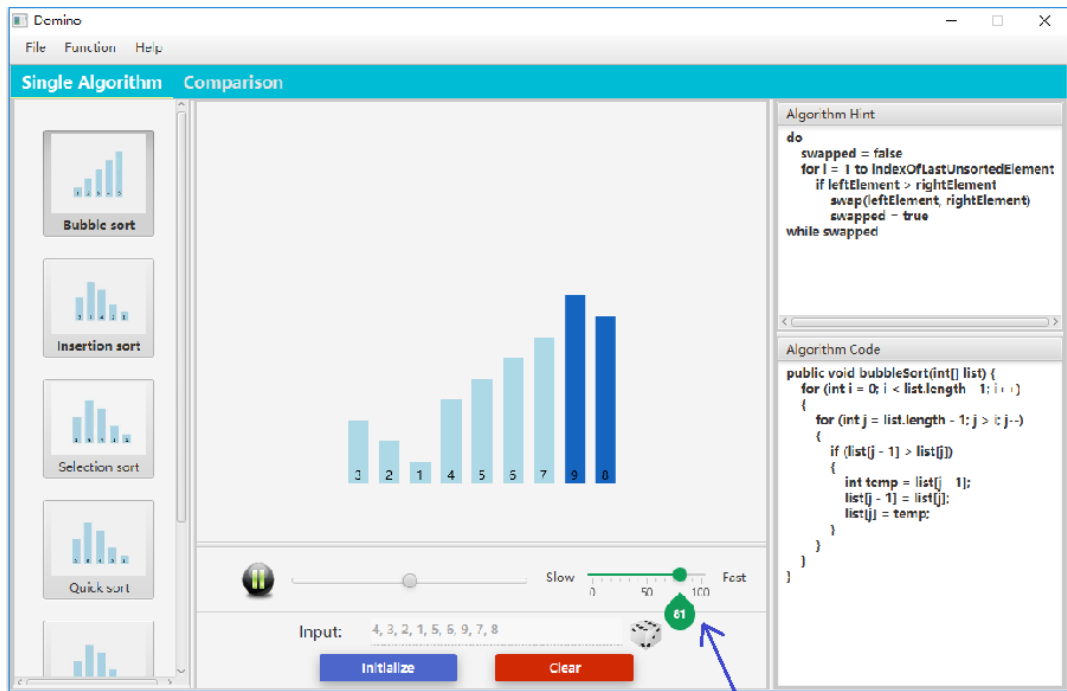
We use the default value as input to run the animation



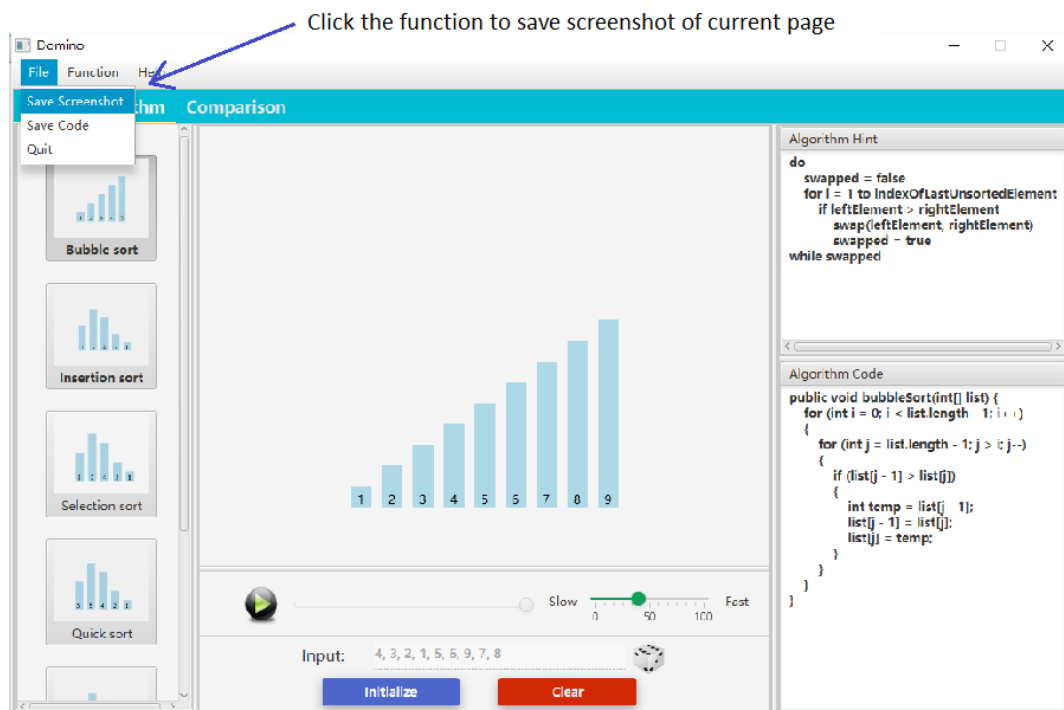
Click "Start" button to start or pause the animation



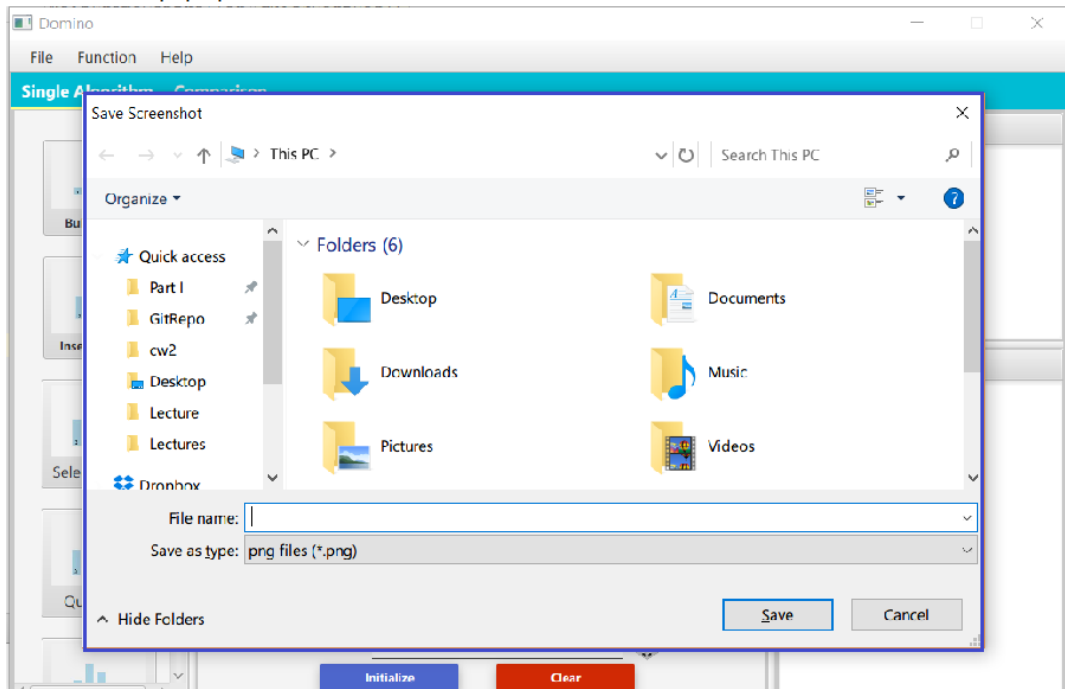
Adjust the bar to adjust the progress of animation

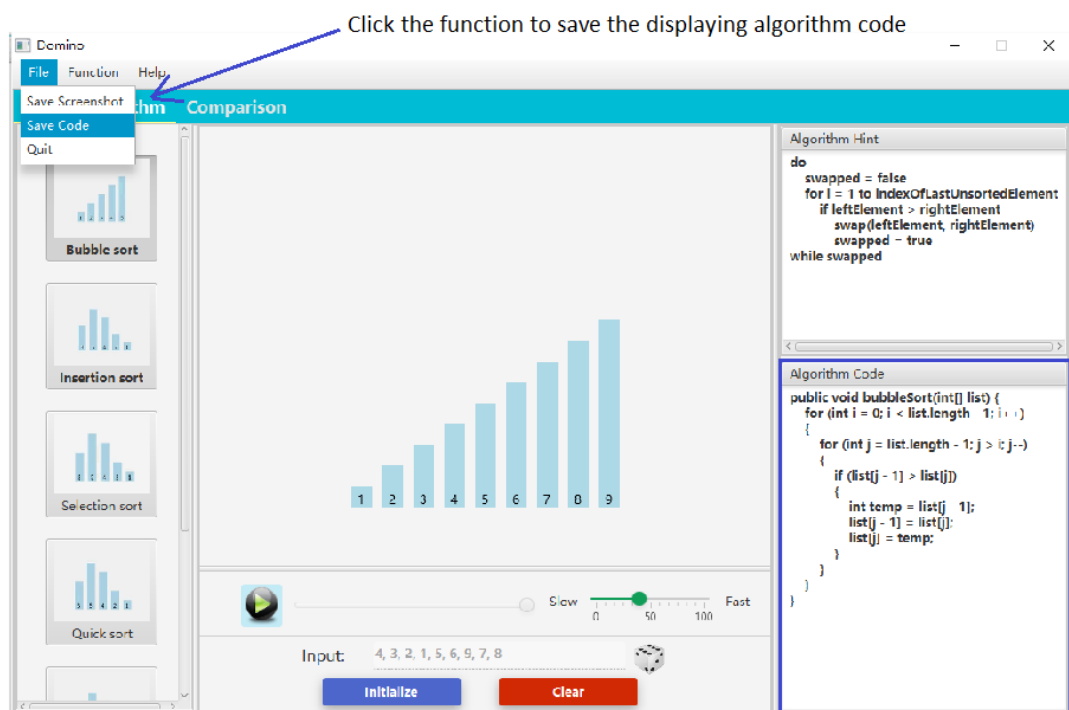


Adjust the bar to adjust the speed of animation

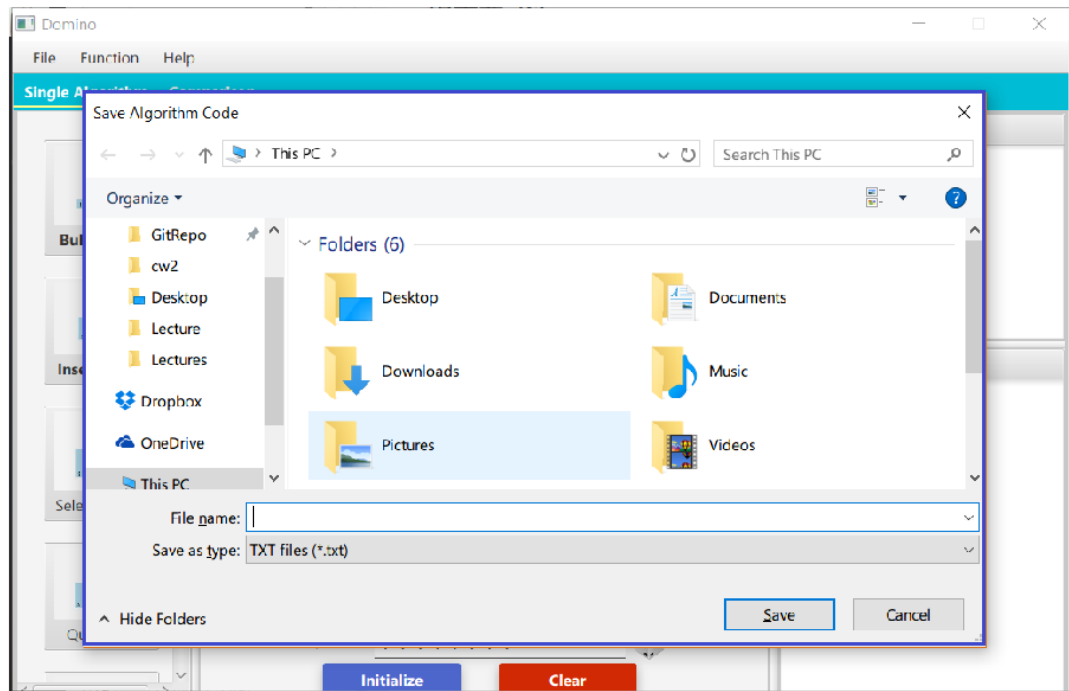


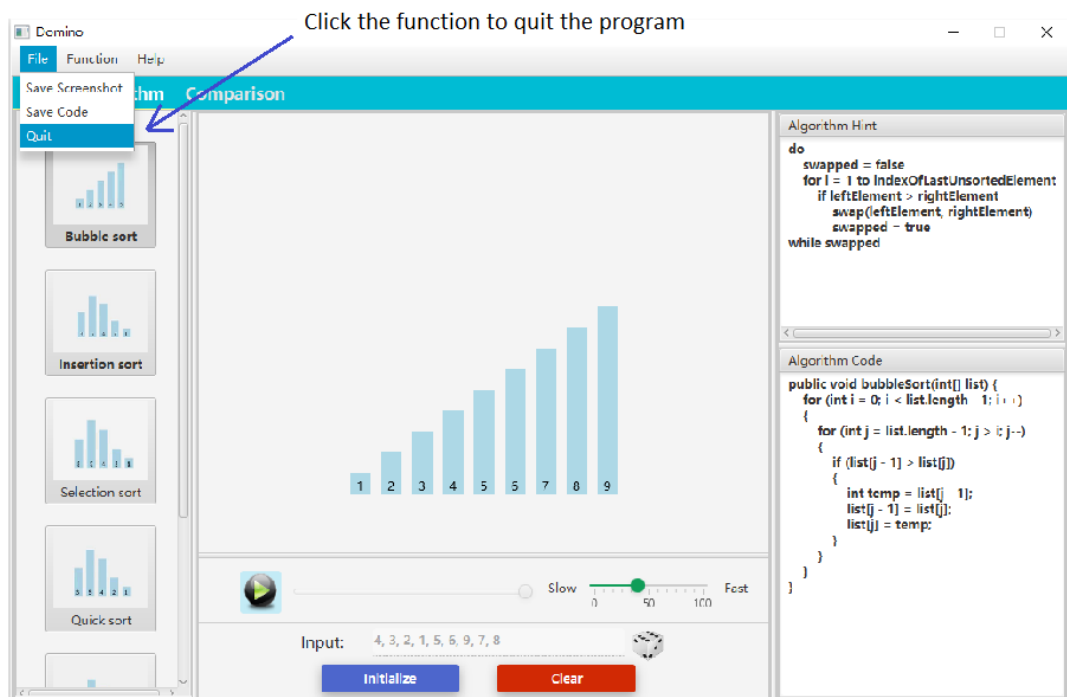
The window is pop up for uses to save screenshot

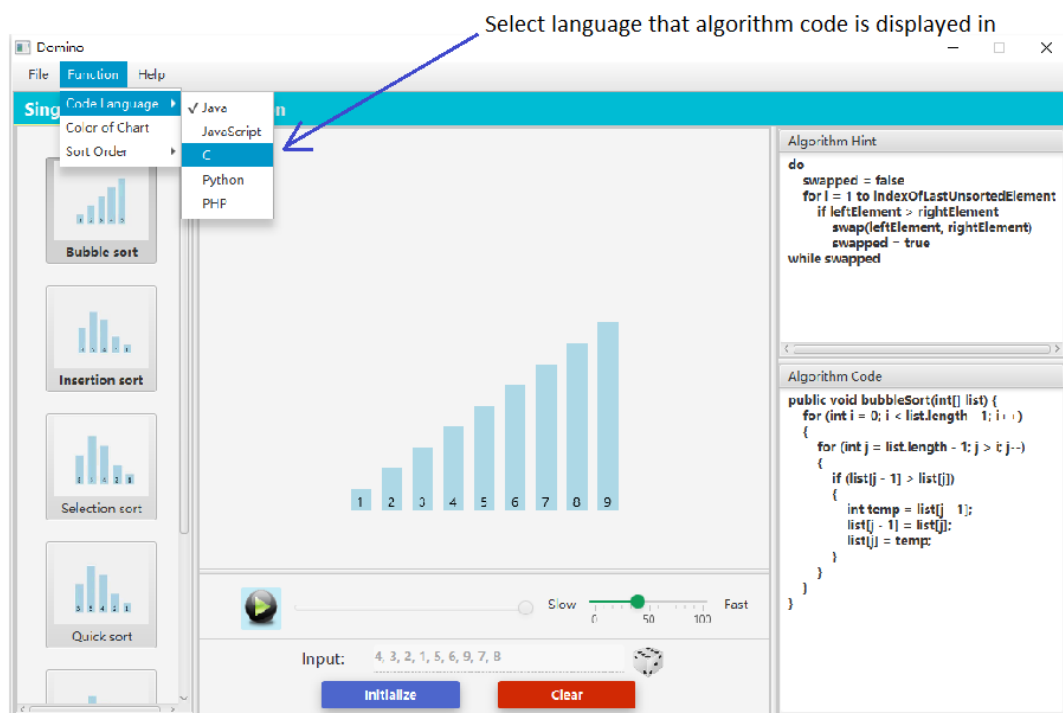


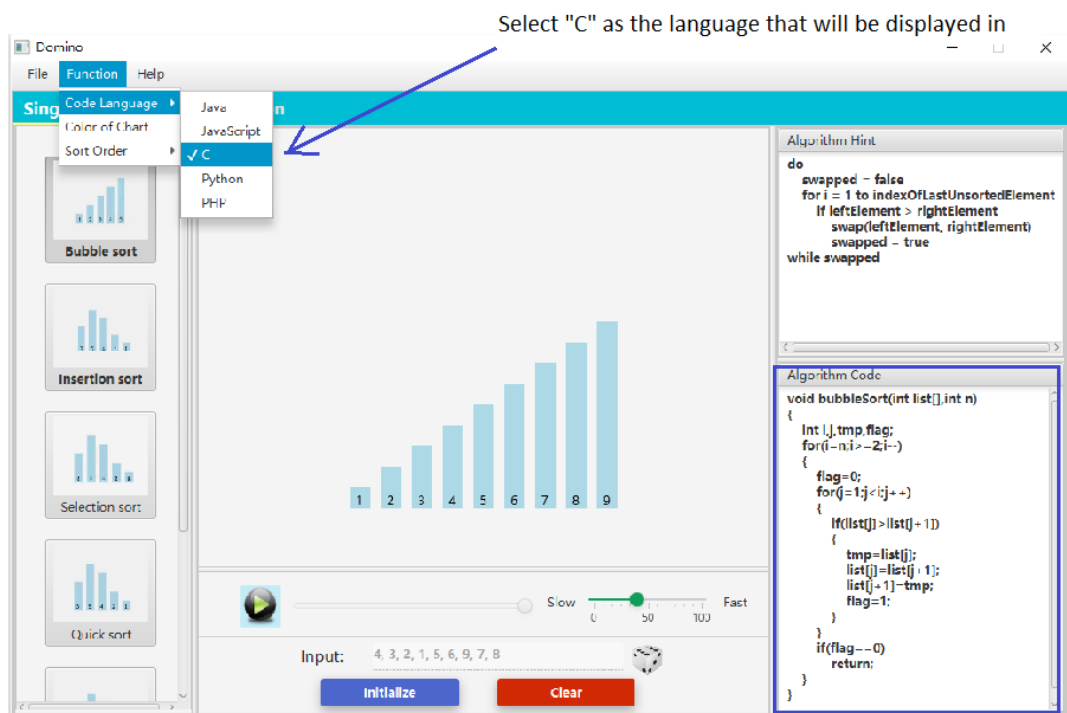


The window is pop up for users to save code









Click the function to change the color of chart

The screenshot shows the Demino software interface. The main window has a menu bar with 'File', 'Function', and 'Help'. The 'Function' menu is open, showing 'Code Language', 'Color of Chart', and 'Sort Order'. The 'Color of Chart' option is highlighted. The main area displays a bar chart titled 'Comparison' with 9 bars of increasing height, labeled 1 through 9. The bars are currently light blue. Below the chart is a speed slider ranging from 'Slow' to 'Fast' with a green indicator. The input field shows 'Input: 4, 3, 2, 1, 5, 6, 9, 7, 0'. There are 'Initialize' and 'Clear' buttons. On the right, the 'Algorithm Hint' section shows a bubble sort algorithm, and the 'Algorithm Code' section shows the corresponding Java code.

Demino

File Function Help

Comparison

Code Language

Color of Chart

Sort Order

Bubble sort

Insertion sort

Selection sort

Quick sort

Algorithm Hint

```
do
  swapped = false
  for i = 1 to indexOfLastUnsortedElement
    if leftElement > rightElement
      swap(leftElement, rightElement)
      swapped = true
  while swapped
```

Algorithm Code

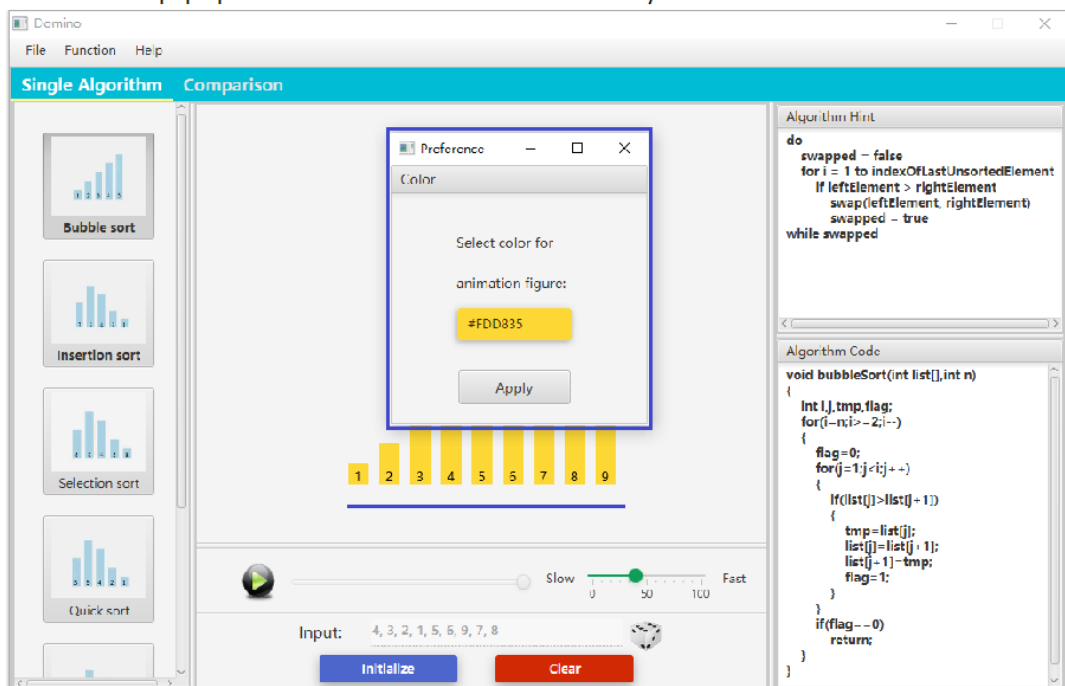
```
public void bubbleSort(int[] list) {
  for (int l = 0; l < list.length - 1; l++)
  {
    for (int j = list.length - 1; j > l; j--)
    {
      if (list[j - 1] > list[j])
      {
        int temp = list[j - 1];
        list[j - 1] = list[j];
        list[j] = temp;
      }
    }
  }
}
```

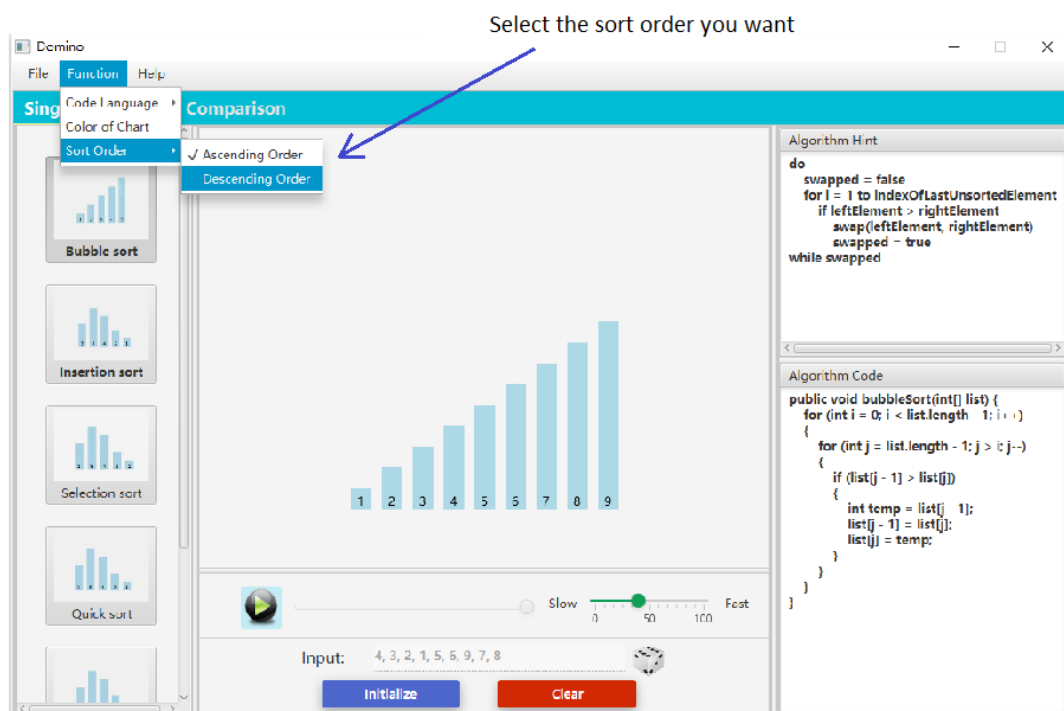
Slow Fast

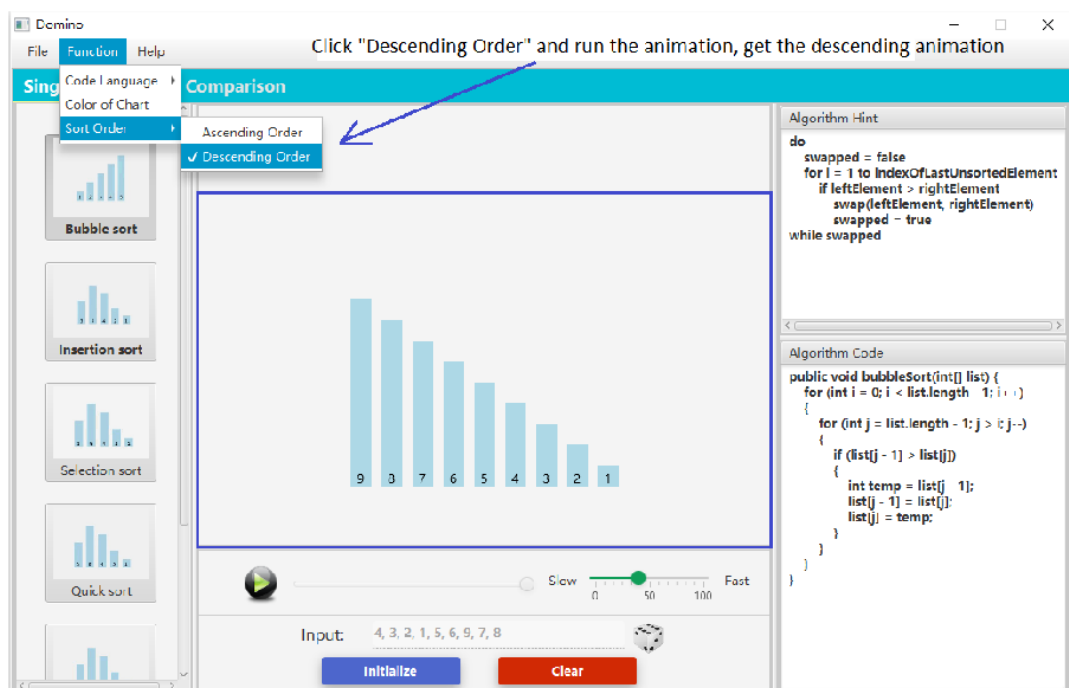
Input: 4, 3, 2, 1, 5, 6, 9, 7, 0

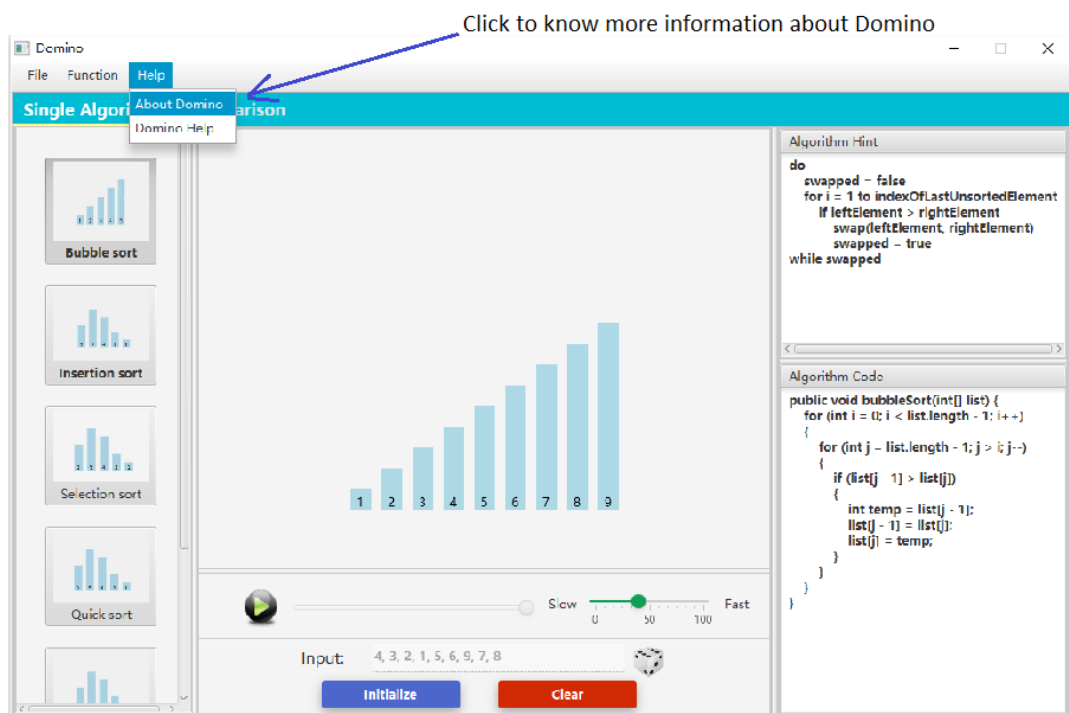
Initialize Clear

The window is pop up for users to choose the color of chart they want

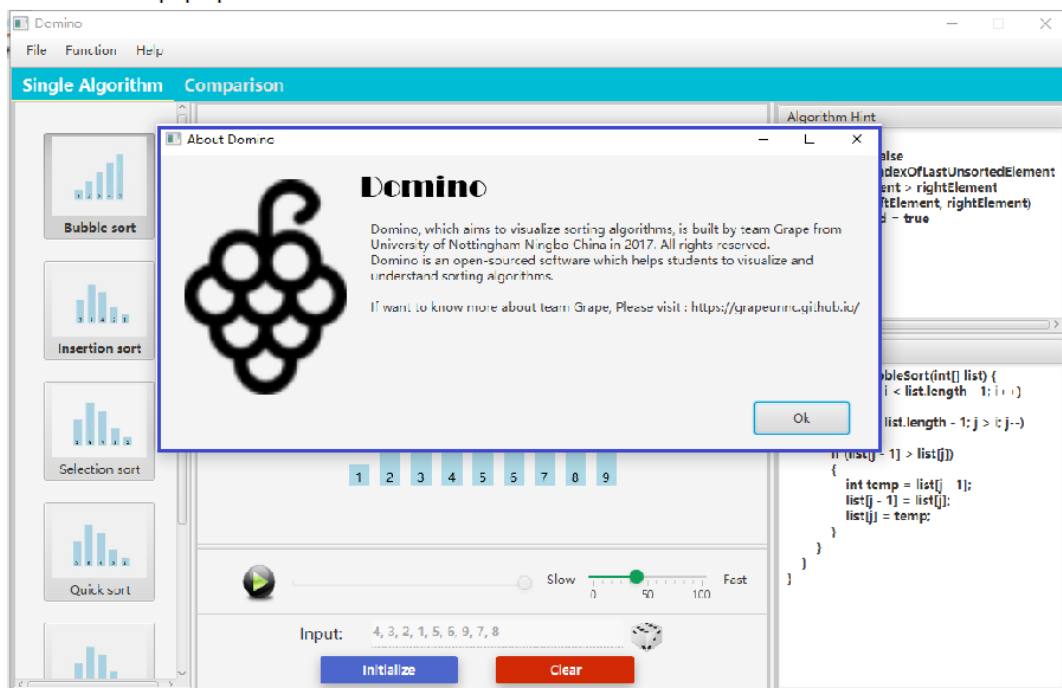




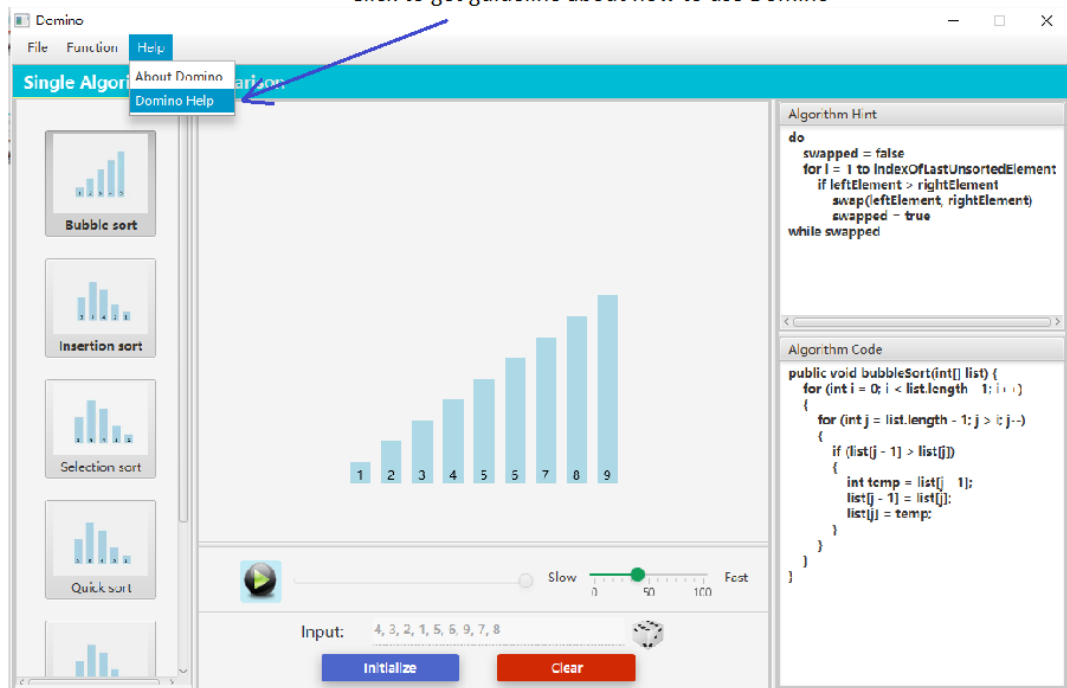




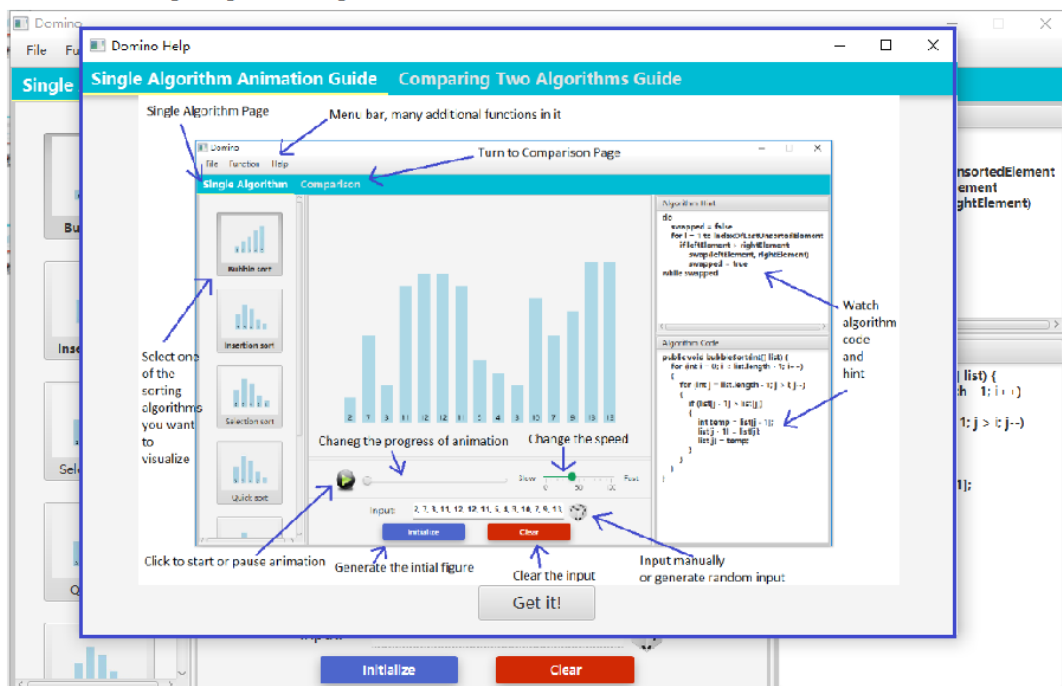
The window is pop up to show information about Domino



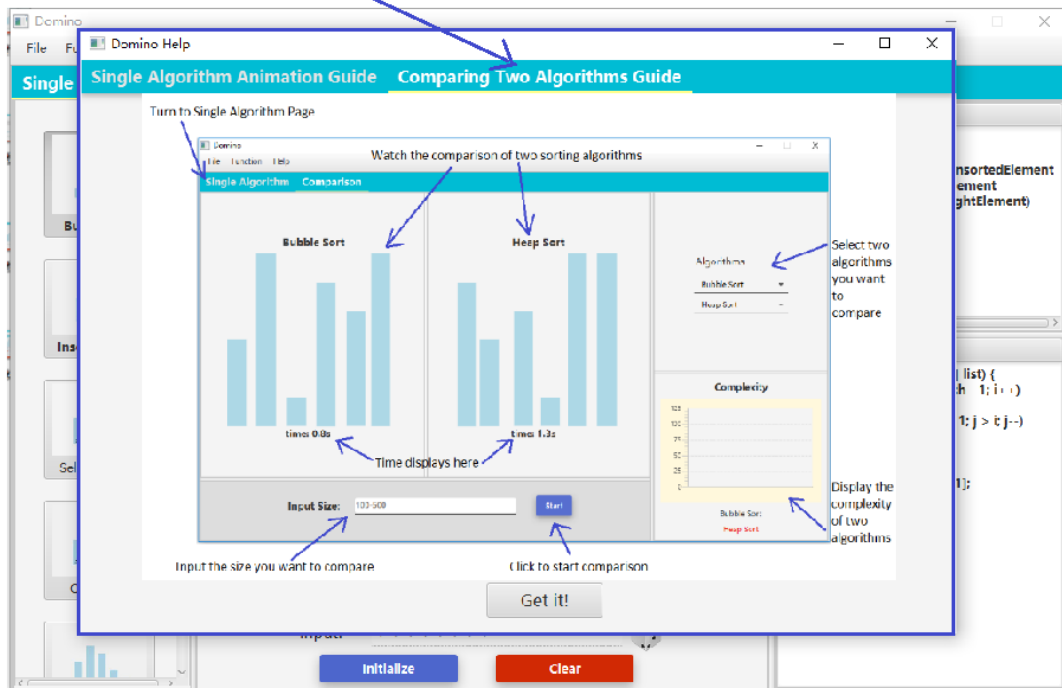
Click to get guideline about how to use Domino



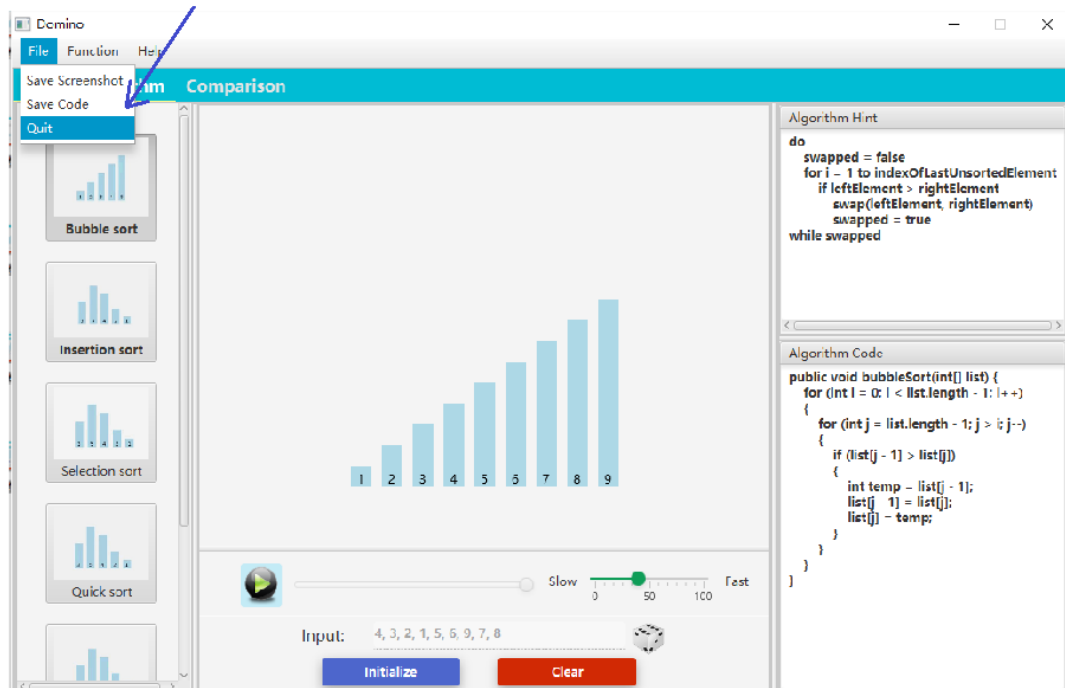
Guideline of Single Algorithm Page



Guideline of Comparison Page



Quit the program



9.4 User Interface Design Update

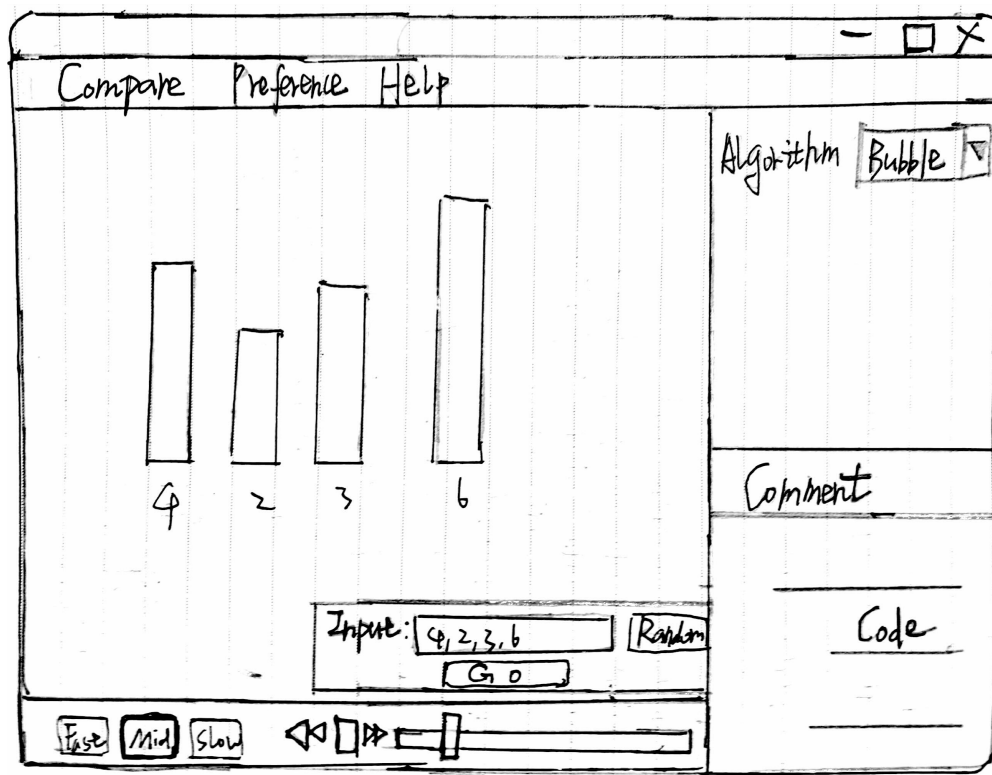


Figure 14: Initial UI Design Scratch

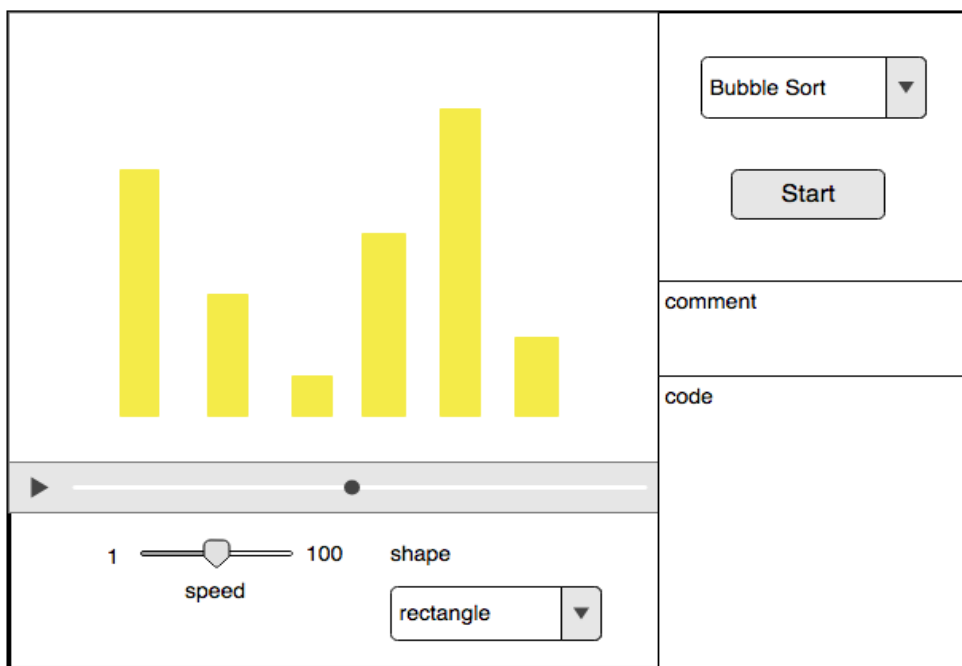


Figure 15: Initial UI Design

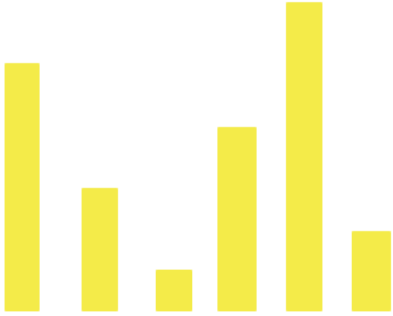

<div>Bubble sort</div> <div>Insertion sort</div> <div>Selection sort</div> <div>Quick sort</div> <div>Merge sort</div> <div>Heap sort</div>		comment
		code
	<div> <div>▶</div> <div>low  high</div> </div>	
	<div>Input <input type="text" value="4, 3, 2, 1, 5, 6, 9, 7, 8"/></div> <div> <div>Random</div> <div>Initialize</div> </div>	

Figure 16: updated UI Design

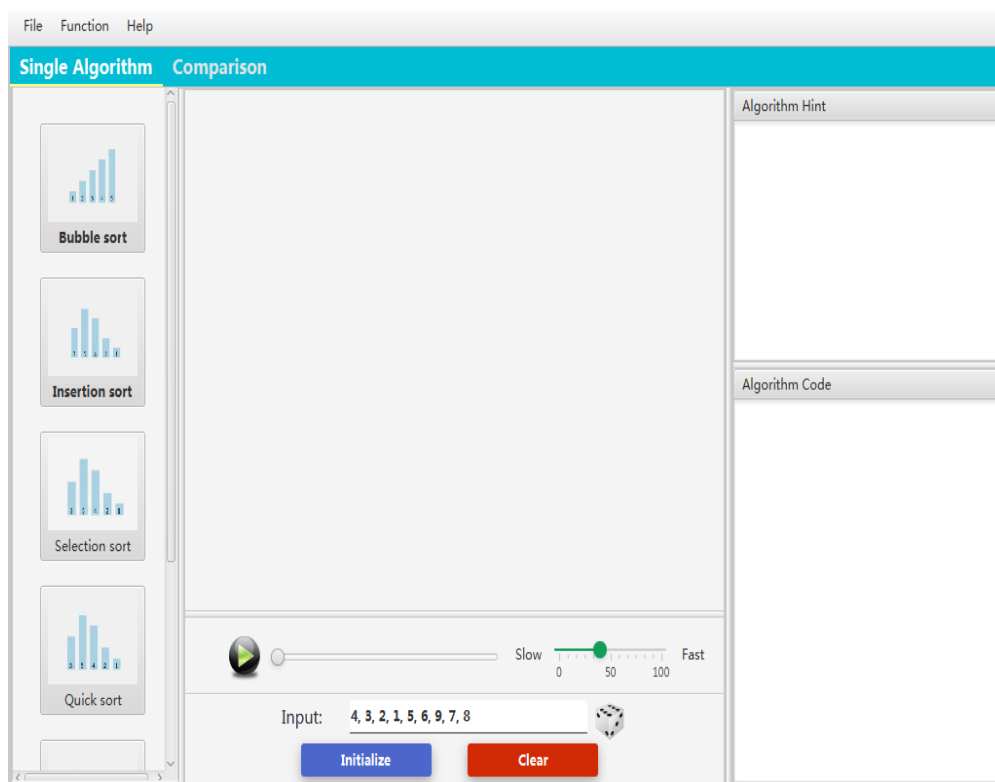


Figure 17: Current Interface

10 References

- [1] D. Byrne Michael, Catrambone Richard, and T. Stasko John. Do algorithm animations aid learning? Report P-2, School of Psychology Georgia Institute of Technology and College of Computing Georgia Institute of Technology Atlanta, 1996. <https://smartech.gatech.edu/bitstream/handle/1853/3504/96-18.pdf?sequence=1&isAllowed=y>.
- [2] M.T.H. Chi, N. De Leeuw, C. Mei-Hung, and C Levancher. Eliciting self-explanations improves understanding. *Cognitive Science*, 18:439–477, 1994.
- [3] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. Social coding in github: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1277–1286. ACM, 2012.
- [4] Engels Gregor, Frster Alexer, Heckel Reiko, and Thne Sebastian. *5 Process Modeling using UML*. John Wiley & Sons, Inc., 2005.
- [5] Alain Wegmann and Guy Genilloud. *The Role of Roles in Use Case Diagrams*. Springer Berlin Heidelberg, 2000.
- [6] Donald Bell. Uml’s sequence diagram. *IBM. [Online] IBM*, (2), 2005.
- [7] Scrum overview for agile software development. <https://www.mountaingoatsoftware.com/agile/scrum/overview>. Accessed: 2016-11-26.
- [8] Linda Rising and Norman S Janoff. The scrum software development process for small teams. *IEEE software*, 17(4):26, 2000.
- [9] Alistair Cockburn and Jim Highsmith. Agile software development, the people factor. *Computer*, 34(11):131–133, 2001.
- [10] TIOBE. Tiobe index for november 2016. Technical Report +31 40 400 2800, 2016 TIOBE software BV, Victory House II, Company Number 2089, Esp 401, 5633 AJ Eindhoven, The Netherlands, 2016. <http://www.tiobe.com/tiobe-index/>.
- [11] Wallace Jackson. An introduction to javafx 8: Exploring the capabilities of the java 8 multimedia engine. In *Beginning Java 8 Games Development*, pages 75–99. Springer, 2014.
- [12] StatCounter Global Stats. Top 7 desktop, tablet & console oss in china from oct 2015 to oct 2016. Technical report, StatCounter, globalstats@statcounter.com, 2016. <http://gs.statcounter.com/#os-CN-monthly-201510-201610-bar>.