**Focus Group for Requirement Gathering of Sorting Algorithms Animation Software**

Position:

Project Room 07, Library (Year 1 and Year 2);

Project Room 14, Library (Year 3)

Time:

November 7, 2020 (Year 1 and Year 2);

November 12, 2020 (Year 3)

To help prepare the questionnaire and acquire basic requirements in users’ aspects, we did a focus group research.

Computer science students from three grades, year 1 to year 3, were invited to our focus group meetings. We held three focus groups respectively. Each focus group had six students. The purpose of the focus groups was to receive feedback for our questionnaire and get inspiration for some new ideas about our software.

Brief participants information and an overview of those focus groups are presented in the following. Meeting process and discussion results will be included.

**Participants Background & Recruitment**

As mentioned above, there are three focus groups divided by participants’ grades (from year 1 to year 3). One of our group members was a department master of UNNC student association SESA (Science & Engineering Student Association), We found all participants from year 1, year 2 groups through SESA, we carefully ensured that participants in the same group were unfamiliar with each other. Also, they did not know our team members. It ensures the objectivity of the outcome of the discussion. Year 3 group participants were from junior students of CS major. All participants knew the purpose of the focus group and volunteered to attend.

All participants had signed relevant information documents at the beginning of the whole process.

**Process**

In the beginning, we introduced participants our software’s purpose and basic functions at a theoretical level. Then, for participants’ further understanding of our product, we presented some similar existing software and prototypes and showed functions which we may also include. After giving them a general idea of our software, here comes to the free discussion part, the discussion based on some questions we prepared in advance.

**Discussions**

**Year 1 group**

Year 1 students have less related knowledge to sorting algorithm than we expected. That makes them hard to give expected responses to some of the questions we prepared. Because writing pseudo-code is still a bit hard for participants, using pseudo-code blocks may help them to learn the idea of writing sorting algorithm better. Participants also mentioned that they hope the software can provide a time complexity comparison of different types of sorting algorithm. Participants preferred the software with a clean interface instead of an implementation of game mode. Due to the lack of relevant background knowledge of participants, the discussion result of this focus group is less than expected but still gave us some valuable advice.

**Year 2 group**

Year 2 students’ discussion was out of our expectation. Many innovative and creative ideas were raised in this focus group.

**User guide**

We planned to have a user guide to give users a quick look at how to use this software, explaining each the usage of buttons. When talked about this feature, participants quickly responded that the user guide is necessary. One participant added that user guide should not contain too much text, or users may not want to read them.

**Fun and game likely VS functionality**

When asked which type of software they would like to use, more interesting or more functionality, participants showed a strong preference for functional software. They would like the software to have a single purpose, without those distracting things.

**Method of selecting modules**

We introduced two methods of selecting modules to participants. The first method was that users cannot access the next sorting algorithm’s learning module unless they finished the one before. The order of modules is designed by us, from easy to difficult. The other method was free choosing. It seems that participants would like to freely choose an algorithm module instead of “unlocking” the next module by finishing the current one.

**Displaying code**

We asked participants’ attitude about pseudo-code and source code. Their responses were helpful. They thought pseudo-code is enough since it gives a logical idea of sorting algorithms. Besides, pseudo-code is more friendly to beginners, which is our target user group. Source code can be presented, they claimed, and for language, C, Java and Python are preferred.

**Multiple language support**

When asked if the multi-language function would be helpful, the participants responded positively. They indicated that there might be some Y1 students who are not that good at English, a teaching software which only uses English may decrease its educational effectiveness.

**Quick export of brief notes**

Students responded positively to the idea of exporting a brief note. They thought this function is useful, but not that essential.

**Platform preference**

The participants were asked which platform they prefer. Following options were provided: PC, mobile app, WeChat mini program and website. The result is that the PC was more popular than others. Participants also said that PC would be a more suitable platform for notes’ quick export function. Participants showed great interest in a mobile app at first for its portability, but they overturned this idea in a short time since they realized that a mobile screen is too small to see the animation model.

**Additional ideas**

**Showing algorithms’ complexity**

One participant suggested that we could add a function showing each sorting algorithm’s efficiency, i.e. its time complexity.

**Breakpoints**

Participants mentioned that using breakpoints in pseudo-code or example code might be helpful for understanding code. With breakpoints, user can skip those code they already understand and quicken the learning process.

**Using sound**

One of the participants showed us a video – many blocks with specific sound were sorted after sorting algorithm and made a fluency sound. He also recommended us to use sound to help make the learning process easier for users. He reckoned this way of presenting is quite perceptual.

**Modules division**

One request was raised by our participants that is adding an exercise module. For the exercise module, it could contain some exercises like some multiple-choice. They also described software in their mind: there were two modules, one for learning and one for practice. The former part is just sorting algorithm’s animation, like what we designed. The latter part is consolidation and practice.

**Progress bar**

After presenting and explaining our preliminary prototype, participants showed great interest in the progress bar we designed. They thought that a display of where they were in the program would be helpful. Besides, they suggested us to design history and reset function.

**Year 3 group**

Year 3 students’ discussion focuses more on correctness and provides other suggestions as well.

**Difficult parts of learning sorting algorithms**

Our participants were confused about the recursive steps of some sorting algorithms. Loops were also difficult to understand. One of the solutions is that demonstrate each step. Participants all agreed that animation would be an intuitive way of understanding the sorting process.

**Problems in learning correctness**

Correctness is hard to understand at the beginning. Participants explained that they did not expect that correctness would have a connection to mathematics and predicate logic. They hoped us to show animations of correctness logically and provide sufficient examples. However, it is difficult for year one students to understand predicate logic.

**Way to understand correctness easier**

Participants suggested that providing stages letting users get used to the concept step by step would help. For example, termination and correct output for any legal input are two fundamental ideas of correctness. These two steps can be animated separately to illustrate fundamental ideas. One of the participants raised an idea that was using truth table to show the mapping relationship between outputs and legal inputs. Moreover, people can understand a concept easier by counterexample of incorrect algorithms.

**How to make it interesting**

The last topic was a module that allows users to build algorithms by dragging pseudo-code blocks provided, and the software will generate corresponding code later. One of the participants suggested that using cute and unexpected things to attract users. He suggested that an animated yellow duck character can be displayed at a corner of the software to denote whether the pseudo-code users build is correct. If it is incorrect, the duck will die. This duck can also be used as a listener to record users’ understanding of algorithms when users speak to it.