yz709-FHCI-sup2

Question 1

[CDs design analysis]

Use Cognitive Dimensions to compare between two programming languages: Scratch (https://scratch.mit.edu/) and Dart (https://dart.dev/).

This analysis should include:

- A sketch of the interfaces highlighting the relevant UI elements
- A brief discussion of the relationship between the *notation* and the *environment* for manipulating the notation for each of the two programming languages
- A brief discussion of the notational activities that users typically engage in (which activities are most important/frequent for each programming language?)
- Then select one activity that is typical for both Scratch and Dart and compare between the two programming languages by briefly discussing:
    - Which are the relevant dimensions for this activity
    - How well each of the relevant dimensions supports the activity
    - Any trade-offs between dimensions for this activity
    - Why you think the designers might have made these design choices

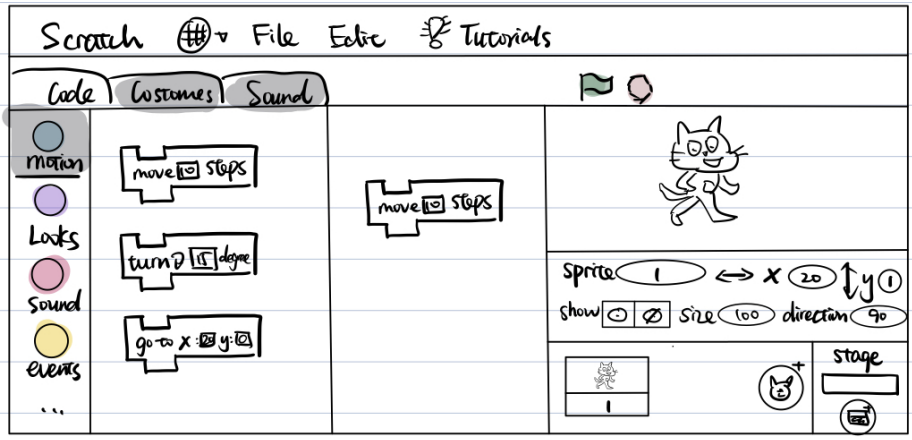
- The relationship between the notation and the environment:
    - Usability is a function of notation and the environment; for an environment designed for a specific set of users, the notation should be easily interpretable by the target users; in that way, we could improve the system's usability.
    - Scratch is an interactive development system designed for children to learn the idea of programming, so all lines of code are transformed into code blocks, colour themes are implemented to attract children's attention.
    - Dart is a language for developing fast apps on any platform, used for multi-platform development. Its target users are programmers (usually with experience in other languages), so the interface is as simple as possible and aims to help programmers concentrate on their codes rather than being distracted by colourful themes and animations.

- Notational activities users typically engage in:
    - In Scratch, incrementation is usually done by dragging and appending code blocks on an existing block structure to build a program. Children will also be engaged in transcription to transform their handwritten logic into the Scratch program. Most kids would typically do an exploratory design when learning Scratch because they would like to know what can be done by trying out all possible code blocks in the main menu. Search is another activity that children would do very often. They might search for a block name from the main menu by browsing the categories or use `control + F` to search for a function or a variable name as it is difficult to find after the program gets to a reasonable size. Modification replaces all duplicated code blocks with a modular function code block.
    - In Dart, incrementation is usually done by adding new functions or lines of codes to implement a program. Collaboration between programmers is crucial for developing an application. Transcription, where programmers transform a system design idea into a set of programs, is one of the common use cases of Dart. A search would often be done in programming once the program gets to a complex structure; modification replaces all duplicated lines of code using a function.
- Take the activity of modification (e.g., encapsulate a function that moves the character by x horizontally and y vertically to replace all similar lines of codes in the previous program) as an example:
    - Relevant dimensions include viscosity, abstraction, secondary notations, and progress evaluation.
        - A higher level of abstraction leads to a higher mental load, which requires secondary notations to aid in comprehension, but this leads to worse diffuseness as the language becomes more verbose.
        - A worse viscosity leads to worse error-proneness because it is harder and more workload for programmers to make one change and thus more likely to make mistakes.
    - Secondary notation is much richer in Dart than Scratch because we can use comments to indicate the objective of the function, while in Scratch, it is not possible and requires users to remember the aim of the function code block.
        - Designers made this decision in Dart because it is a collaborative tool, and comments aid in comprehension. But in Scratch, children usually

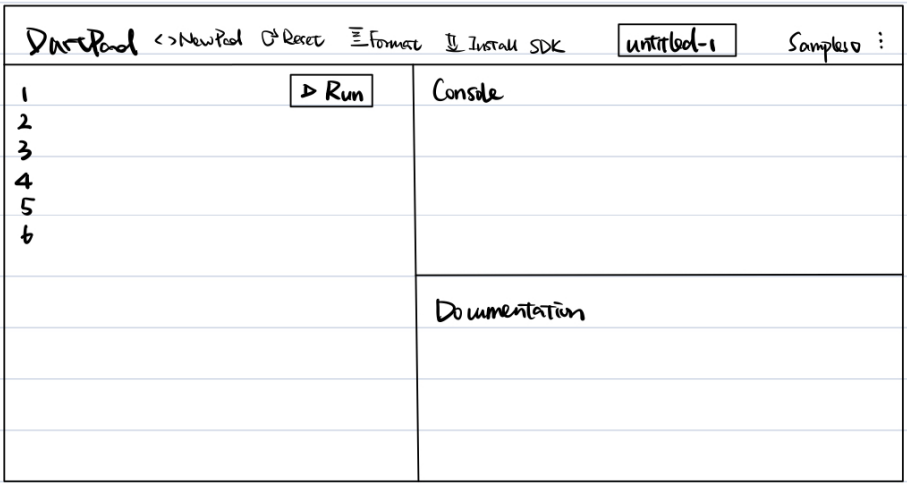
work on a project alone and are more likely to implement simple programs, so comments are not that essential.

- A higher level of abstraction is needed for Dart than Scratch because we need to generalise the lines of codes, but in Scratch, we only need to use the same code blocks to construct a similar code structure.
    - Designers made this decision in Dart because the idea of abstraction is essential for programmers to understand to proceed to learn more advanced concepts, but in Scratch, it is too hard for children to understand the idea of abstraction, so designers tried to use the different shapes of code blocks to help children concatenate them.
- Scratch would do a better job in progress evaluation because we have a screen that acts as a debugger to check the progress. We could replace one set of code blocks to see whether we have achieved the desired results and replace that with all duplicated code blocks once we have expected results. However, in Dart, we can only see the results in the console and need to trigger the debugger to see the whole process.
    - Designers made this decision in Dart because it is the programmers' job to write error-prone codes. They may more often write a complex program that only the final results are understandable; the intermediate progress is too complicated and verbose. However, in Scratch, children usually write programs to manipulate the motion of the spirit, which means the intermediate motion of the spirit is what interests children.
- The viscosity in Scratch is worse than Dart because all the code blocks have to be deleted individually before replacing them a call to a separate function code block; but in Dart, less work have to be done as we only need to replace all relevant lines of code by a new line using a find-replace tool.
    - Designers made this decision in Dart because it is one of the common use cases where programmers have to encapsulate a function to replace duplicated lines of code, but in Scratch, the learning process is more important than cleaning the program.

Scratch



Dart



Comments:

Question 2

2. CDs design intervention

Propose a way in which the design of one of the programming languages above might be modified that would have an effect on one of the *dimensions* you discussed above for the selected *activity*. Consider any trade-offs that might result and discuss whether the proposed modification changes other dimensions for that activity or for other activities.

How would you evaluate whether the proposed design intervention has the impact you expect?

- Suppose we add a set of comment blocks in Scratch as a secondary notation, then for the selected activity (i.e., encapsulate a function that moves the character by x horizontally and y vertically to replace all similar lines of codes in the previous program). In that case, we could comment for the function code block to indicate its objectives and comment above each function invocation to indicate we are referring to the specific function code block.
- This aid in comprehension because children do not need to go through each code block of the program to see what it would do, as comments could help them get a sense of the functionality if they are written clearly. In addition, this also indicates that fewer mental operations are required to understand a piece of the code block. However, comments take spaces and may lead to a more verbose language as more symbols are needed.
- This change will benefit all kinds of incrementation, transcription, collaboration, and exploratory design activities because all these activities involve understanding codes lines. This task gets more complicated when the program gets complex. But this comment block adds another layer of complexity for search activities, especially if we want to find a variable name using `control + F`. The comments might be picked up and mixed with the variables, making visibility worse. Users are much harder to locate the elements, and more cognitive work needs to be done.
- Evaluation: we could use a summative, empirical evaluation method to collect quantitative data, specifically, using controlled experiments.
    - (1) Gets a set of function blocks with comments in and the other without but keeps all other contents the same. (2) Select a group of children with similar familiarity with Scratch, each of them are assigned a mixture of function blocks, some with comments and some without. They have to do multiple choice questions by selecting the correct objective of each function code block; the selection process is timed individually for each multiple-choice question. (3) We could then analyse each function block's time taken to compare the average time for commented function blocks and

uncommented ones to get an idea of whether comments help in comprehension.

💡 Comments:

Question 3

3. Designing different types of systems

Discuss the differences and similarities between designing a professional sound editor and designing a fitness app (max 1500 words).

You can start by differentiating between which kind of system (as listed in the lecture titles) best describes each application, or you can use one (or more) of the theories discussed in the course to compare between them. Your essay should also include a brief discussion comparing the context of use, purpose of the two applications, which frameworks/theories would be most useful in the design process for each of these (and why), and which formative and summative research methods would work best for evaluating the design of each system (and why).

The professional sound editor is an efficient system, while the fitness app is an interactive system.

The target customers are not the same. The professional sound editor software is designed for professional audio editors, musicians and social media influencers such as YouTubers. Hence, customers' main incentives for using a professional sound editor software are to boost their efficiency and embed any features easily inside their videos or soundtracks. The software would primarily be used among a team of editors, working together to produce the best quality soundtracks. They are experts on various sound-related features and would like to learn about all advanced techniques to enhance their soundtrack. Hence the software needs to provide a list of advanced and professional tools (e.g., music rebalance, guitar de-noise, and dynamic de-hum).

On the other hand, the fitness app is designed for the general public to promote exercise and healthy lifestyles. Customers may want to find a workout community on the fitness app or let the fitness app track their activities to help build their exercise habits. Hence, the contents of the fitness app should be designed to suit a wide range of customers, with different ages, jobs and knowledge about fitness. Live interactions will be very desirable; comments, posts, and reactions to posts would be suitable for user interactions with peers within the virtual workout community.

Because of the different purposes of the two applications, they require a different user interface design to enhance usability, and various sets of factors need to be considered when designing the user interface. For the professional sound editor software, a good performance, high flexibility to compose and edit, high similarity to other mainstream methods to avoid explicit learning is more important than enhancing user engagement, visual attention and animations. Hence, minimalism is always applied when designing such a user interface. On the contrary, the fitness application generates revenue from advertisements and in-app purchases; the size of the customer base is the primary concern. It should be designed by putting user engagement first. In addition, it also has to lower computational demands to allow the application to run on portable devices.

Since the design of the professional sound editor software concentrates on efficiency, the Keystroke level model can be used to analyse the speed with which an expert user can complete a task without errors in a user interface. As the professional sound editor software needs to embed a list of advanced techniques, users need to find the desired tool as fast as possible. So the categorisation and the naming of the tools would need to be designed with care; experiments using KLM could be carried out to analyse whether a particular arrangement of the tools is reasonable and would help users accomplish a task efficiently.

For the fitness application, ethnography research could be carried out to gain a deep understanding of users' context, everyday life, habits, and customs. In that way, we should design the fitness application to help people be active and social on the platform and help them develop good exercise habits to improve their quality of life. The ethnography research workshops should be held parallel with the design workshops in an iterative fashion. We first need to understand people's current experience with online exercising sessions and recent healthy lifestyle trends. More specifically, how much money and time they would spend on fitness, and whether their current lifestyle is an obstacle to carrying out off-line exercising in gyms. In order to design a product successfully, we also need to know whether the fitness app would help support customers achieve their objectives, and we need to figure out what kind of unique selling points are there to differentiate our fitness application from other competitors. In the second round, we should understand how users would relate to novel scenarios brought by our innovative designs. For instance, if we invite well-known fitness KOLs to our platform, whether their fans from other platforms would be attracted to download our platform even if the contents the fitness KOLs uploaded are similar.

For the professional sound editor software, cognitive walkthrough could be a possible analytical evaluation in the formative evaluation stage. Since there are a set of common activities, we could ask a tester (not the user) to ask questions about a specific user journey when carrying out an activity (e.g., how to remove guitar noises from the soundtrack). This help evaluate the product usability more efficiently and gets a better understanding before the actual design phase.

When analysing the professional sound editor software after a working product, we could use a quantitative and empirical summative evaluation method - A/B testing. It carries out a statistical evaluation to get an insight into the effects of a particular change in the software, which could inspire future refinement and design ideas.

The fitness application requires a different formative evaluation method, such as interviews with a group of customer representatives who have different backgrounds, ages and previous experience with a fitness application. We could set open research questions to help understand customers' expectations with the fitness application and fulfil these user requirements in our product design.

A possible summative evaluation for the fitness application could be proxy measures such as the number of days the customers actively using the product and the average time spent on live streams and pre-recorded exercise sessions. This measurement has less latency, and the data is much easier to gather.

💡 Comments:

Question 4(y2018p7q8)

This question relates to the design of control software for home automation. There is a market opportunity arising from increasing deployment of software-controlled light bulbs and fittings. As householders acquire larger numbers of such products from different suppliers, they will wish to create lighting schemes, coordinating lights located in particular areas of the house or in individual rooms, where contrasting schemes might be designed for various times of day or activities.

You are asked to consider ways in which HCI research can be conducted to compare the relative advantages of two technical approaches to this opportunity. The first approach is to use machine learning algorithms to infer the schemes from user actions. The second approach is to provide a programming language with which users can define the schemes.

(a) For *each* of the two technical approaches, describe a usability problem that you might expect to arise that is specific to the lighting scheme application, and a design strategy that could be taken to reduce the impact of this problem, noting any trade-offs that may result. [8 marks]

- For the first approach which uses machine learning algorithms to infer the schemes from user actions, a usability problem is that it takes time to train the ML model and at the starting few months, the ML model will not be automated enough and the lighting scheme may frequently output wrong actions. If this training phase is too long, users would have a bad experience and be annoyed by the dumb lighting scheme they may propose replacing with another manual scheme.
    - Provide a set of previous user actions to pre-train the model, i.e., use a supervised learning technique first to generate a reasonable ML model. This ML model would then be placed in the new environment to adjust its lighting scheme according to new user actions. This reduced the new environment's training phase and provided a better user experience and usability in the starting few months. However, collecting previous user actions is a time-consuming task, and we need to make sure the collected sample dataset is a good representative of all future user actions.
- For the second approach, which provides a programming language with which users can define the scheme, a usability problem is that the defined scheme is fixed and would not be able to evolve itself; users need to manually do the changes for all different rooms with different lighting requirements. A detailed and time-consuming system design phase needs to be carried out to ensure a coordinative light scheme with products from other suppliers.
    - Design a system that is generalised enough that later changes consume little effort. This would place a huge mental load on the initial system design and may need to introduce some compromises. For instance, rooms will be categorised into several groups and each with the same settings, we could simply change the category assigned to the room whenever we want to change the room from one lighting scheme to another. But this categorisation process requires a lot of effort to ensure it would adapt to possible future requirements.

(*b*) Propose a formative empirical research method that could be used to compare the relative desirability of the two technical approaches, from the perspective of the home lives of potential customers. Your proposal should include a description of the practical steps involved in carrying out the research, the data that will be collected, the way that it will be analysed, and the kind of recommendations that you expect to be able to make. [6 marks]

- We could provide a survey or a questionnaire containing both open and closed questions to the potential customers. In the survey, we should describe both systems, list a set of advantages and shortcomings of each design, and allow users to think about further effects. For instance, a closed question could be the range of the onsite training time user would accept, while an open question could be "which system would you prefer and why".
- Collecting data via questionnaires are fast, and we can easily get a large sample. Once we gathered all quantitative data, we could carry out a statistical comparison of quantitative measures, e.g., whether the mode onsite training time is possible.
- For qualitative data, we could use Grounded theory. Read the data and look for interesting categories, writing memos to capture insight and organised themes using axial coding. A pilot study might be needed to test questionnaires on the qualitative data to the correctness of the survey.
- From the data evaluation, we could get an idea of customer preference and why they would prefer one compared to the other one, which gives us evidence to recommend one system instead of the other one.

(*c*) Your company has deployed alpha-release versions of both systems. Early feedback from users suggests that there is a problem when a new lighting scheme must be defined quickly, for example when guests arrive for a party. Propose a summative empirical research method that could be used to compare the speed of the two technical approaches, in order to find which is faster. Your proposal should include a description of the practical steps involved in carrying out the research, the data that will be collected, the way that it will be analysed, and the kind of recommendations that you expect to be able to make. [6 marks]

- Controlled experiments could be used to compare two systems. We could invite a group of participants with similar familiarity with the two systems (or no familiarity with both), asking for their consent before collecting data. They will be assigned a set of tasks that all involve setting up a new lighting scheme quickly

(e.g., when guests arrive for a party, or when a storm would come in the morning and go around the evening time). Then they are provided by the lighting schemes and need to implement on both systems individually. This implementation phase is timed, and this data set is collected for evaluation. We could use analytics and metrics in A/B testing, assume a null hypothesis that the systems require the same amount of time, and see whether statistically the results are significant.

- If the results are significant, then we could make a recommendation for the faster one.

Comments: