9. Project description

Please describe the project you are planning to undertake. How would it help you in your scholar development plan? (200 – 500 words).

I am planning to design a programming language named "Qu". Qu is a non-Turing-complete programming language designed to queue objects on a presentation slide. For example, if an user wants to display “image1” first, then the text “text1”, they can express it as the following code:
“<1-> image1
<2-> text1”
which, after rendering to the presentation slide, would behave as expected.

Currently, it takes a lot of time and clicks to control object appearance order in a presentation slide. For Microsoft PowerPoint and Google Slide, a user must click on each object they wish to control, go to the “Animation” menu, scroll to find the desired effect, then click “Apply”. For LaTeX Beamer, image controls are slightly more convenient, but a lot of copy-paste would need to be done in order to achieve custom order. Python Manim module has great results but it demands a lot from the user, since the code library has become more and more complicated to serve the needs of advanced users.

Qu aims to provide the great control that Latex Beamer and Python Manim has, without the extra copy-paste or a steep learning curve. The language is simple, and it serves to do one thing only: queue objects so that they appear in the desired order. This facilitates the presentation process of students and professors alike, making well-ordered presentations less time-consuming.

The programming language has 2 main parts: the language design and a compiler. The design of the language is determined by the context-free grammar that specifies the production rules. The first part of the compiler takes in source code written in Qu, creates a matrix that shows the status (visible = 1, invisible = 0) of each object in each slide. This matrix clearly expresses all the objects and its visibility on each slide, making the process of automating slide making easier.

In terms of scholar development, this project presents a lot of challenges for me to explore and pushes me to become a better engineer. Most of the necessary theoretical knowledge I have learned from CS302 - Theory of Computing class at Fulbright. This project provides practical experiences in language design and compiler building, and I strongly believe that there is always a gap between theory and practice that can be filled by hands-on experience. These skills are highly relevant in modern software development, where designing domain-specific languages and creating compilers or interpreters are increasingly valuable.

Overall, this project would equip me with multiple technical insights and problem-solving abilities necessary for my future endeavors in software engineering.

10. Learning objectives

Please elaborate on what do you aim to learn in this scholar development project (200 – 500 words)

There are several key skills that I would like to practice with this particular project. First, and the most important of all, I'd like to gain practical experience in language design and compiler design. I have learned about the fundamentals of these areas in Theory of Computing, but so far it has been a black box that I don't understand the inner workings of. By creating Qu, I aim to learn how to create a language that is simple, yet powerful enough to serve its specific purpose of queueing objects in a presentation slide. This should require me to explore and consider language features, making the choices and trade-offs between simplicity and convenience, and overall making informed decisions.

Another challenge that I expect to tackle is resolving syntactic sugar in the language. Many languages has shorthand expressions to make writing code more succinct and convenient. Qu also has the shorthand notation to select multiple rows of the same textbox. For example, when writing "text1[0:3]", a user wants to target cell 0, 1, and 2 of object "text1". The compiler must be able to see through this syntactic sugar to understand which exact cells it needs to select.

The third tangible skill I would like to become better at is handling bugs and errors (debugging programs), as well as code documentation. All programs must have bugs, it is inevitable, and I hope by undertaking a demanding project such as this compiler, I would develop better practices to navigate my way out of the compounding error messages. When the first working version of this project is done, I would have to learn how to write proper documentation so that future maintainers or developers have an easier time understanding my code and my intentions.

Overall, throughout this project, I wish to develop my critical thinking and problem-solving skills in general. As with many things, going from theory to practice has a large gap that would not be easy to navigate. I anticipate having a lot of my theoretical understandings to be challenged, and thus I also expect doing a lot of ad-hoc research in order to make sense of the obstacles I face. I also anticipate several "Aha!" moments, when something I did not know before starts to make sense, which is honestly the part that gets me the most excited.

11. Tentative Schedule and Workplan

Note: This is the work schedule that has been carried out the Fall 2023 semester, where this work was done.

- Thursday 14:00-15:00: meeting with instructor, catch up, report work of previous week, discuss work of upcoming week.
- Saturday and Tuesday, 13:00-18:00: work individually on the task of that week.

12. Proposed artefacts for assessment:

In this space, list proposed artefacts for assessment. (Optional) You can suggest certain parameters or criteria to assess your scholar development here.

Note: To keep the deliverables of this project realistic, the first version of the compiler should work with 2 main objects: images and textboxes (one textbox can have multiple lines of text). Other objects that might be incorporated to make this language more useful are (but not limited to) tables, mathematical formulas (LaTeX), code snippets, which is more realistic to reserve for future developers, or future semesters if I decided to apply to work on this project again.

Proposed artefacts for assessment:

- Grammar: A comprehensive document outlining the alphabet, production rules, and sample code for the Qu programming language,
- Compiler implementation: Input is Qu source code, and output is a matrix displaying object visibility for each frame.
- All the source code for the compiler.

Proposed criteria for scholar development assessment:

- Language design: simplicity and clarity of the Qu language.
- Compiler functionality: The compiler should read Qu source code and generate a matrix that accurately represents the order of appearance (and disappearance, if specified by user) of the objects.
- Code organization and documentation: cleanliness, modularity, and readability of the compiler source code.