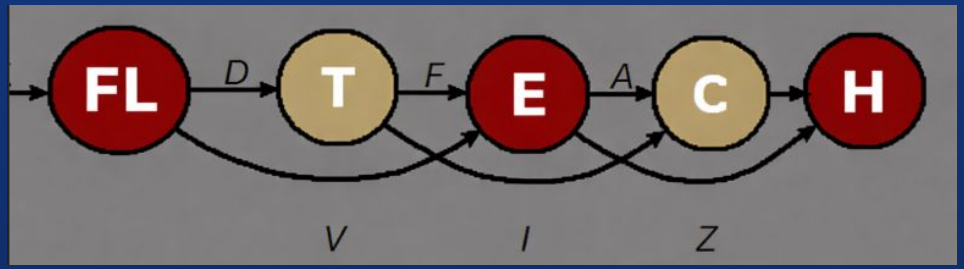
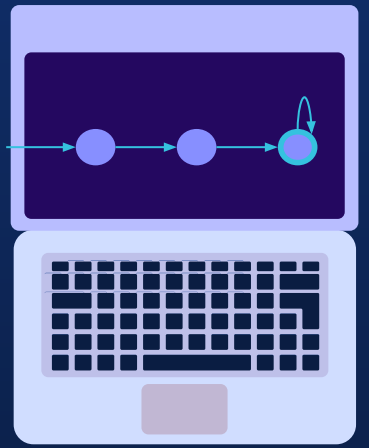
Improved Visualization for Formal Language: **Milestone 6**





https://kmcnear2022.github.io/

Group Members: Chris Pinto-Font, Vincent Borrelli, Andrew Bastien, Keegan McNear

Milestone 6 Goals

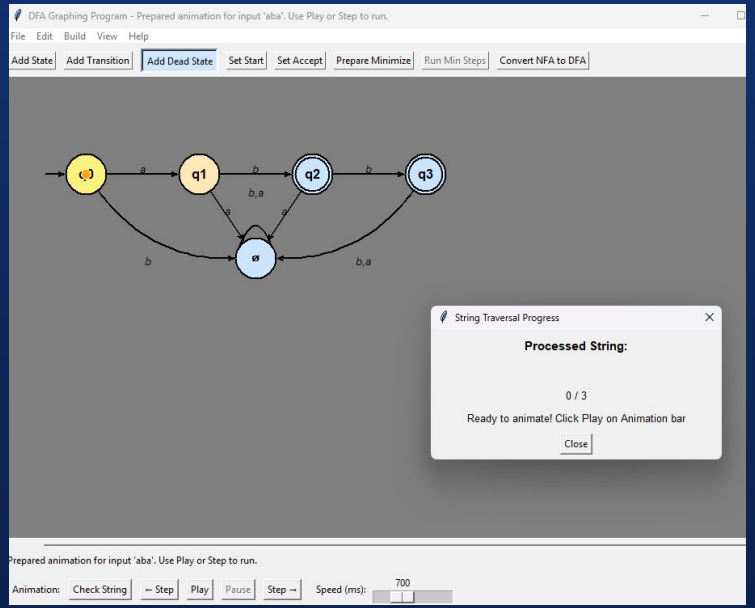
We set out to finalize our project for the senior design showcase through a combination of adding new highly requested features, and refining and testing old ones as to ensure a robust and polished end product.

Milestone Six Deliverables

- Finalized Bug Testing and User Experience
- Automatic Dead State Button
- Revamped Tutorial Mode
- DFA Save and Reload
- Build from Regular Expression
- Finalized User/Dev Manual
- Zoom In/Zoom Out Function
- Finished Teaching Mode
- Release Ready .exe release with custom icon
- Finished Poster
- Reorganized GUI
- Completed Demo Video

Finalized Touches on User Experience

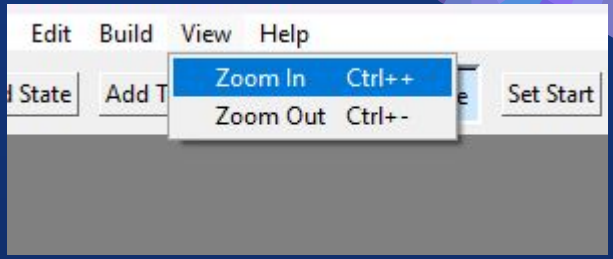
Revamped features such as the automatic DFA builder and program created transitions to not overlap/maintain visual appearance consistently. Bug tested a wide swath of program features and cleaned up visual issues and frustrating elements which could impact user experience.

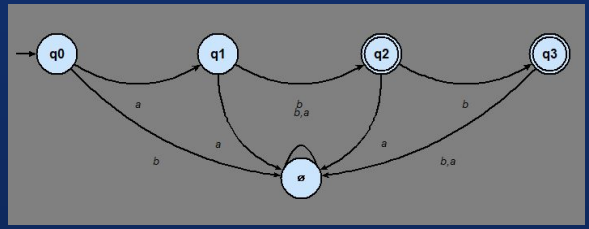


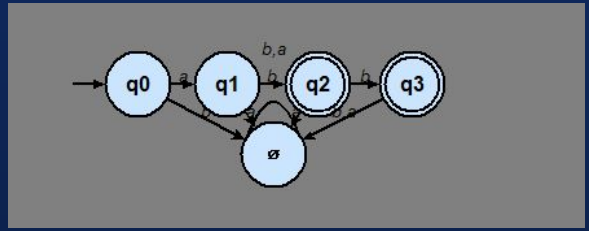
Zoom In And Out (Canvas Scaling)

Implemented a new feature where in which the use can (with the use of the view buttons on the top of the GUI) the user can zoom in and out from the canvas, thus scaling canvas objects accordingly, allowing the user to reposition their work and expand their DFAs dynamically.

The user can zoom out using the mouse in a similar fashion to a web page, where in which the zoom occurs localized around the mouse cursor, giving the user an unparalleled level of canvas repositioning.

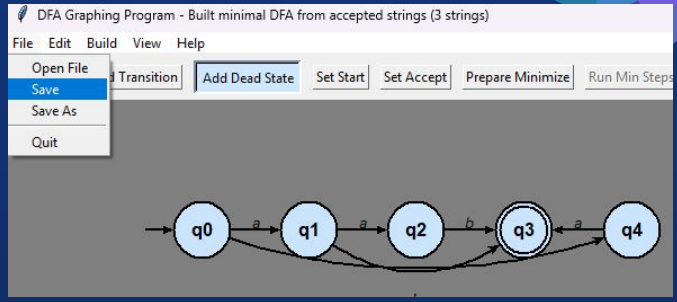


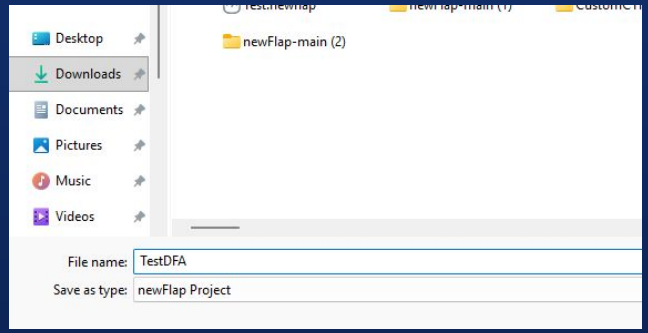


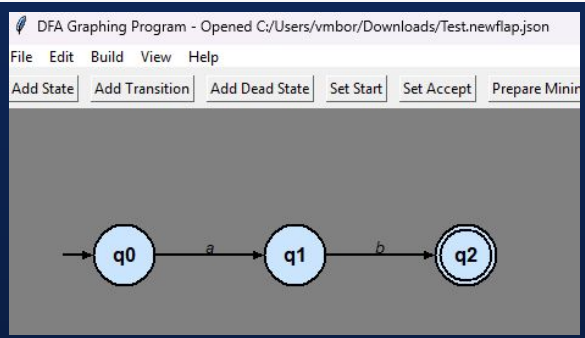


DFA Save and Load

Users can save a constructed DFA as a custom newflap object which can be loaded into the canvas space and edit and traverse at will. This bespoke feature was something we wished to nail down early on, as it helps support the program's role as a teaching tool, as shared newflap files can be used by anyone with the program, letting teachers and students and collaborators exchange graphs freely and easily.

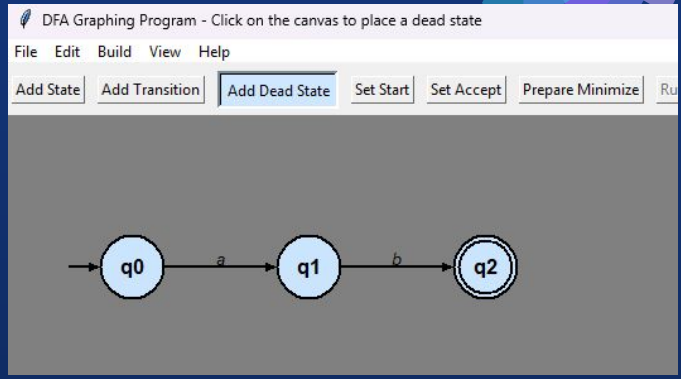


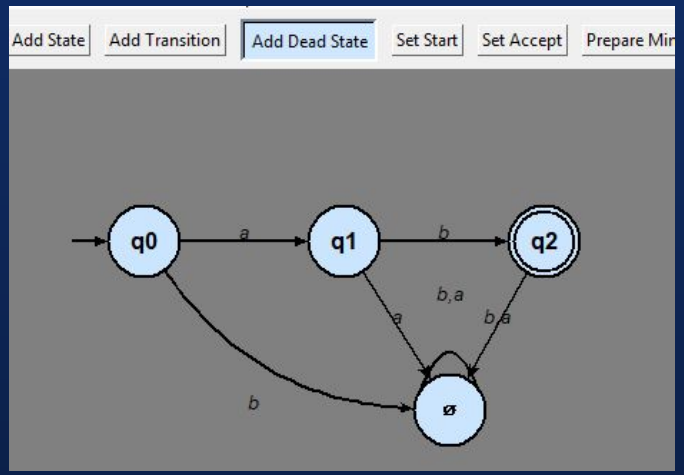




Automatic Dead State

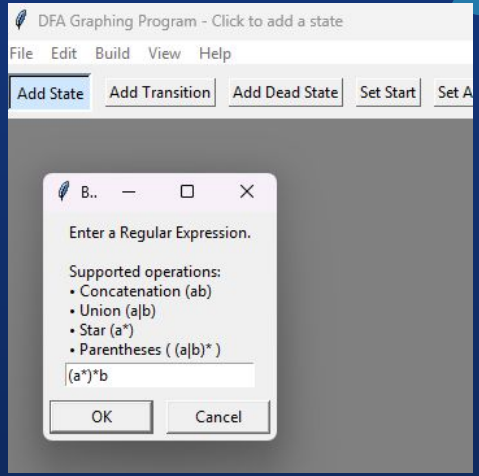
One of the most important quality of life features, and one tied to an algorithmically complex process, we built out a button within the gui which allows the user to construct a dead state for their DFA automatically as to save time within the DFA construction process. All DFAs require a dead state for completion, so allowing the user to add one intuitively to complete their work will assist them greatly.

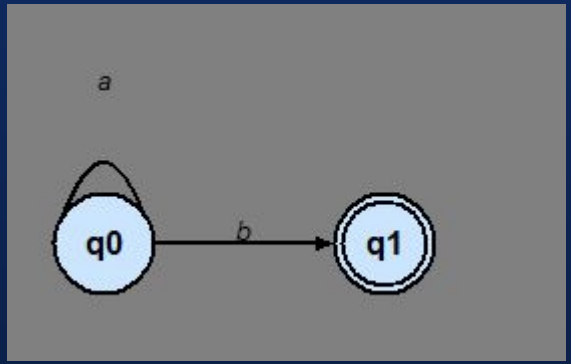




Build From Regular Expressions

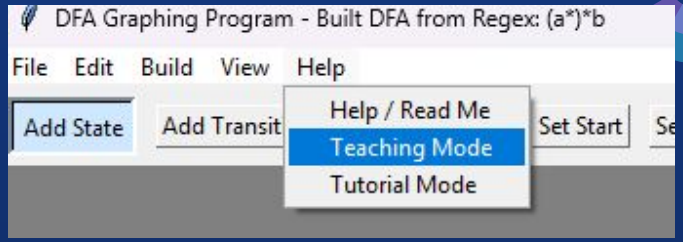
Expanding on our string based construction introduced last milestone, this feature lets the user write out a DFA compliant regular expression and have the program build it out. This is something that often happens in real life teaching of DFAs and thus is a vital feature for our program to demonstrate. This aspect of our program employs a vast algorithmic complexity to accomplish fitting graphs.

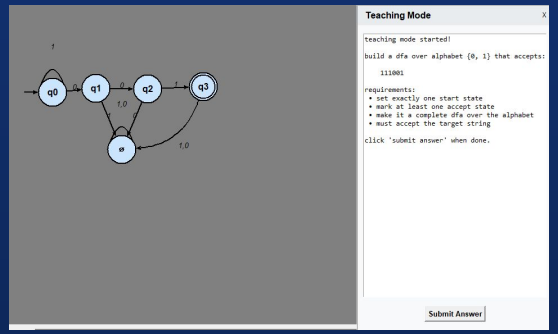


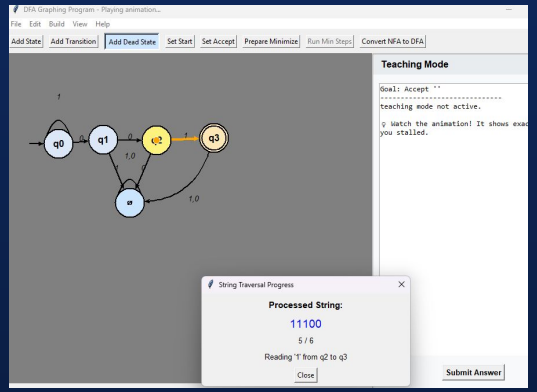


Finished Teaching Mode

Finalized the previously implemented Teaching mode from the previous milestone, with it now incorporating animations into the teaching process via our traversal system. User is now prompted to enter the provided random string and the program shows if it works, helping to communicate the connection between construction and traversal. The dead state button makes this far easier.

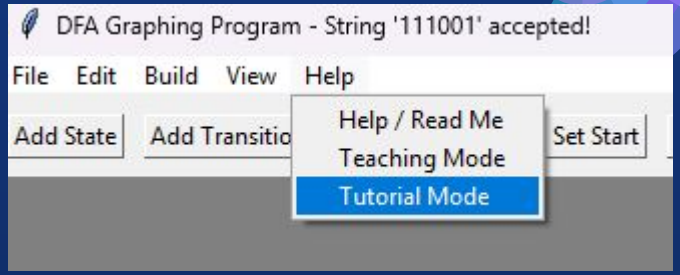


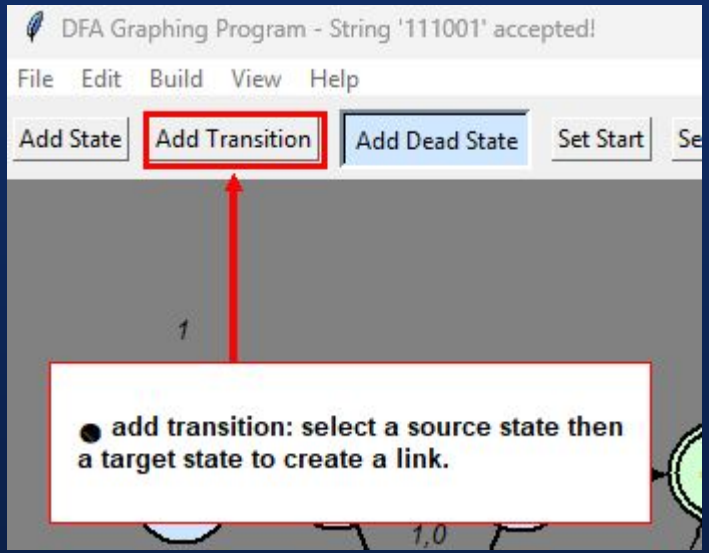




Revamped Tutorial Mode

A departure from the tutorial mode seen in the previous milestone, this version lets the user explore the GUI space, now letting them click on different buttons and being told what they do.

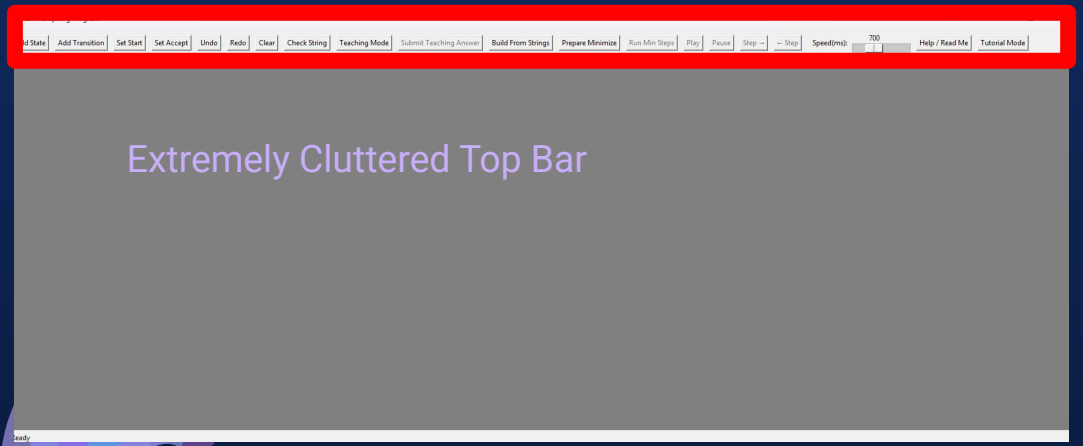




ReOrganized GUI

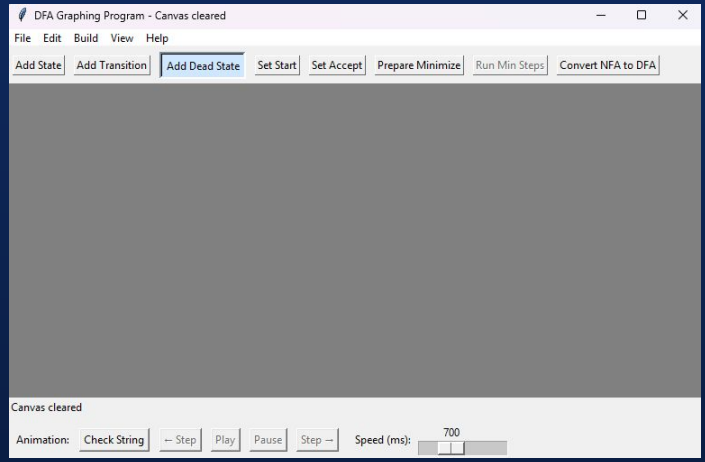
Fixed the cluttered GUI by rearranging the animation controls to the bottom of the screen, and folding program features into new header segments, such as “build” which lets the user access the build from strings and regular expressions aspects of the program.

Old:



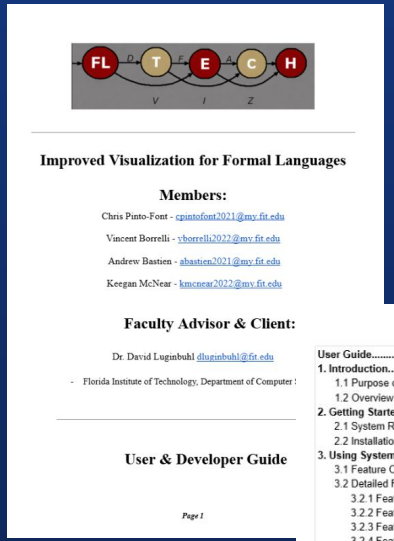
Extremely Cluttered Top Bar

New:



Finalized User/Dev Manual

Wrote and finalized a 25 page user and developer manual to let people view and explore the full scope of the development process and how people should use the program. Comprised of several sections including, but not limited to, System Architecture, System Features, Development Guidelines, and user cases.



Improved Visualization for Formal Languages

Members:

Chris Pinto-Font - cpintofont2021@my.fit.edu

Vincent Borrelli - vborrelli2022@my.fit.edu

Andrew Bastien - abastien2021@my.fit.edu

Keegan McNear - kmcnear2022@my.fit.edu

Faculty Advisor & Client:

Dr. David Luginbuhl dluginbuhl@fit.edu

- Florida Institute of Technology, Department of Computer

User & Developer Guide

Page 1

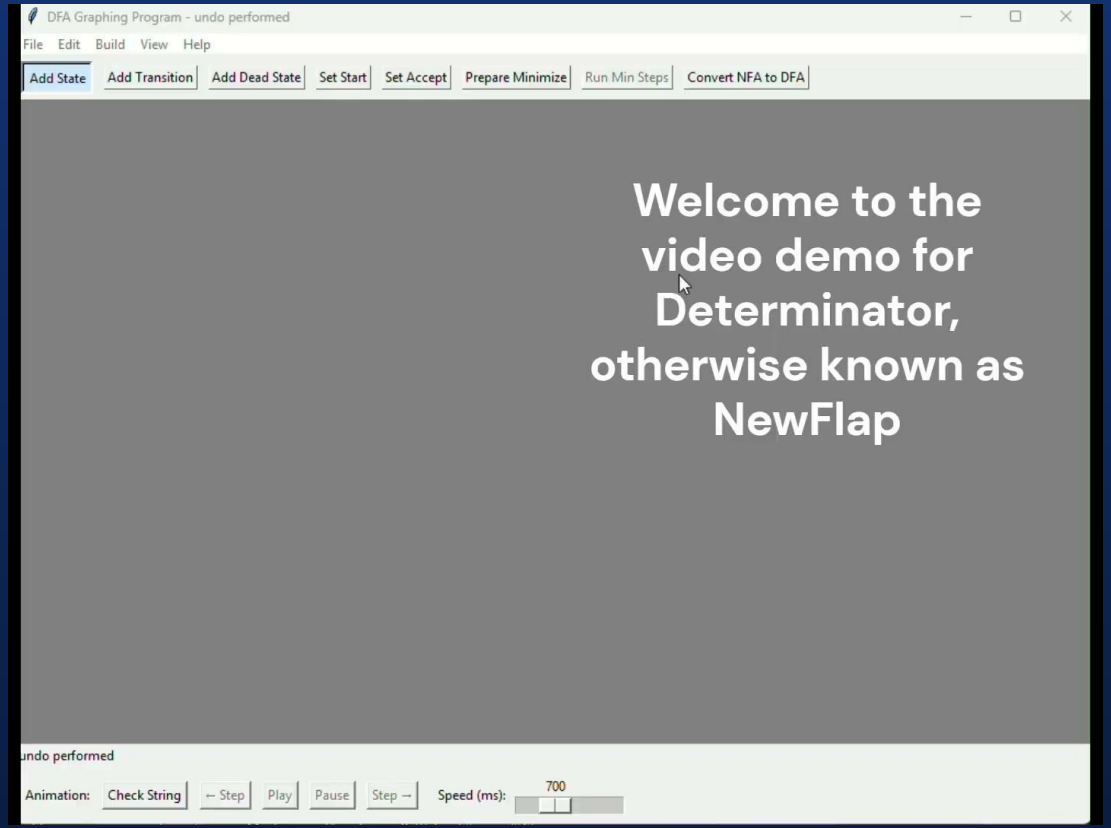
Table of Contents

Page 2

Finalized Ready Version of Program

Compiled and exported a release ready version of our finalized program, including a custom icon for desktop use.

Video Demo with Voice Over



Finalized Poster

NORTHROP GRUMMAN
ENGINEERING & SCIENCE
STUDENT DESIGN SHOWCASE
FLORIDA TECH

VISUALIZATION FOR FORMAL LANGUAGES

Chris Pinto-Font, Andrew Bastien, Vincent Borrelli, Keegan McNear
Faculty Advisor(s): Dr.David Luginbuhl, Dept. of Computer Science, Florida Institute of Technology

Motivation/Goal

- The primary goal is student comprehension of complex DFA concepts.
- This project is important for students and professors because being able to visualize Deterministic Finite Automota (DFA) and how it is created and expressed can lead to better understanding of how parsing and computation works.

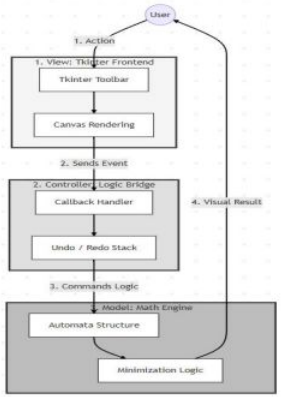
Approach/Features

- Modify states in an interactive dynamic workspace.
- Draw using point and click state and transitions.
- Traverse states given a string input and accept or reject based on DFA with real-time animation.
- Reverse engineer a DFA from string input using teaching mode. (Figure 1)
- Minimize a DFA. (Figure 2)
- Input a regular expression to create a matching DFA. (Figure 3)
- Convert from NFA to DFA.
- Input a string to create a matching DFA.

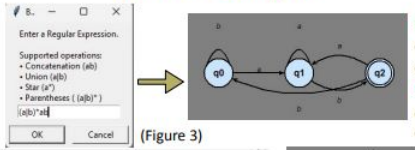
Implementation/Tools

Python, Tkinter, GitHub

System Architecture

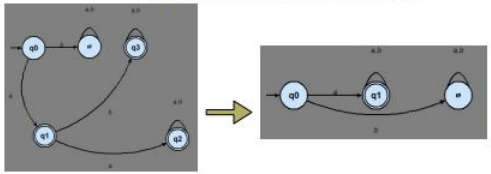


Visualizer Examples



(Figure 3)

(Figure 1)



(Figure 2)

Evaluation Results

- Accuracy:
  - Convert NFAs to DFAs using construction logic.
  - Minimize deterministic machines to their most efficient state.
  - Manage string and regular expression conversion correctly.
- Performance:
  - Ensure GUI responsiveness by handling canvas updates.
  - Optimize memory usage through efficient data structures.
  - Utilize UUID generation with O(1) time complexity.
- Stability:
  - Prevent application crashes through error handling and input validation.
  - Prevent GUI errors using system architecture.
  - Maintain machine integrity by preventing memory collisions.

Future Works

Limitations:

- Dense graphs become visually cluttered
- Limited to finite automata

Possible Improvements:

- Implement a layout to have better visuals
- Upgrade to support PushDown Automata

Milestone 6 Progress - Pt 1

Task	Completion %	Chris	Vincent	Andrew	Keegan	To Do
Finalized Touches on user experience/bug testing	**100%**	**25%**	**25%**	**25%**	**25%**	**N/A**
Zoom out/Zoom in (Canvas Rescale)	**100%**	**35%**	**15%**	**35%**	**15%**	**N/A**
DFA save and load	**100%**	**20%**	**20%**	**40%**	**20%**	**N/A**
Automatic Dead State Button	**100%**	**40%**	**20%**	**20%**	**20%**	**N/A**
Build from regular expression	**100%**	**25%**	**15%**	**35%**	**25%**	**N/A**

Milestone 6 Progress - Pt 2

Finished Teaching mode	**100%**	**15%**	**20%**	**15%**	**40%**	**N/A**
Revamped Tutorial Mode	**100%**	**15%**	**20%**	**40%**	**15%**	**N/A**
Reorganize GUI for user experience	**100%**	**20%**	**20%**	**30%**	**30%**	**N/A**
Updated Poster	**100%**	**50%**	**20%**	**15%**	**15%**	**N/A**
Finalized“read me” file	**100%**	**20%**	**40%**	**20%**	**20%**	**N/A**
Finalized Release Ready .EXE of program	**100%**	**20%**	**40%**	**20%**	**20%**	**N/A**
User/Developer Manual	**100%**	**10%**	**75%**	**5%**	**5%**	**N/A**
Video Demo w/voice over	**100%**	**10%**	**70%**	**10%**	**10%**	**N/A**

Questions?

Visit Our Site

https://kmcnear2022.github.io/