Team SMART

SMART@cwuwildcat.onmicrosoft.com

GENERAL LINE COORDINATES FOR GRAPHS AND   
TIME SERIES

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| 1. **The Team** | ………………………………………………................................................. | 2 |
| 1. **Objectives** | ………………………………………………................................................. | 3 |
| 1. **Graphs** |  |  |
| * 1. **CPC vs SPC** | ………………………………………………................................................. | 3 |
| * 1. **Time Series** | ………………………………………………................................................. | 7 |
| 1. **Benchmarks** | ………………………………………………................................................. | 9 |
| 1. **Iterative Model** | ………………………………………………................................................. | 9 |
| 1. **Python** | ………………………………………………................................................. | 10 |
| 1. **C++ Conversion** |  |  |
| * 1. **Solution** | ………………………………………………................................................. | 11 |
| * 1. **Tools** | ………………………………………………................................................. | 12 |
| 1. **Challenges** |  |  |
| * 1. **General** | ………………………………………………................................................. | 12 |
| * 1. **C++ Specific** | ………………………………………………................................................. | 13 |
| 1. **Schedule** | ………………………………………………................................................. | 13, 15 |
| 1. **Conclusion** | ………………………………………………................................................ | 13 |
| 1. **Links** | ………………………………………………................................................ | 14 |
| 1. **Weekly Reports** | ………………………………………………................................................ | 16 |

1. **The Team**

|  |  |  |
| --- | --- | --- |
| **Supervisor/Client:**  Dr. Boris Kovalerchuk  CS Department, CWU | | |
| **Julia Carter**  Front-end  Python/ C++ Implementation  *(never used)*  SRS | **Allyn Vo**  Front-end Lead  Back-end  Program Lead  *(final weeks)* | **Phillipe Shin**  Back-end Lead  Graph Lead |
| **Paul Collet**  Manager  Documentation Lead  Presenting Lead | **Mahlet Saketa**  Back-end  Time Series Lead |  |

1. **Objective**

Our objective is to provide a desktop application that takes data from a text or CSV file and create multiple graphs from that data. The application is for general use and will include a GUI for the user to view and modify the graphs. This type of software application is referred to as *data visualization software*. Only structured data, in a comma delimited format, will be considered.

Data visualization, if done correctly, helps people make sense, or realize patterns, from data. The more data a data-set includes, the greater the need for tools that communicate that information clearly and efficiently. Examples of the type of information this software could be used on are:

|  |  |
| --- | --- |
| * Surveys/Census * Weather data | * Networking * Traffic congestion |

1. **Graphs**

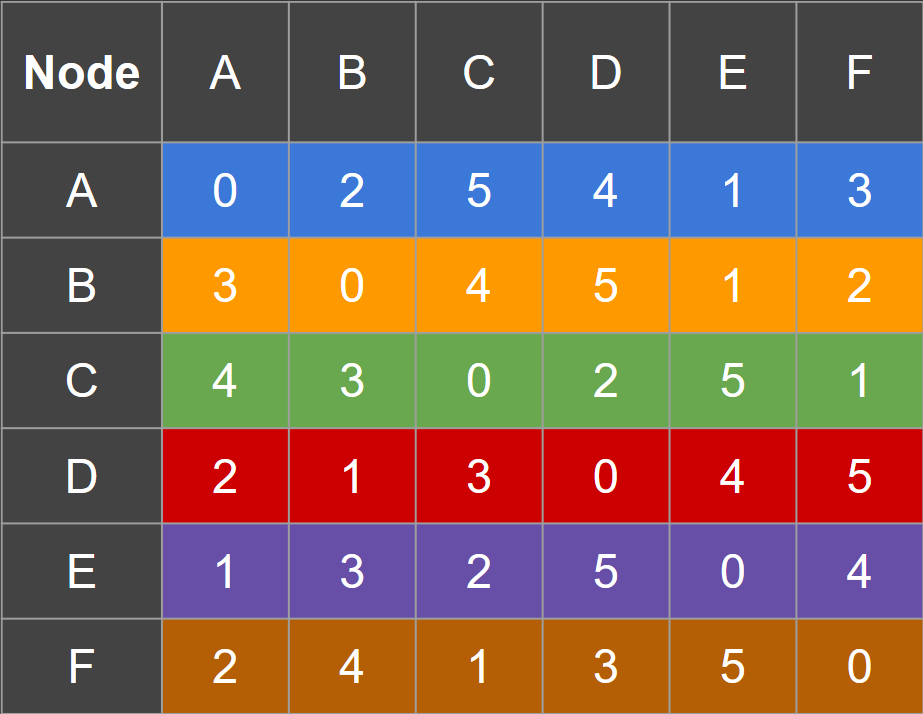
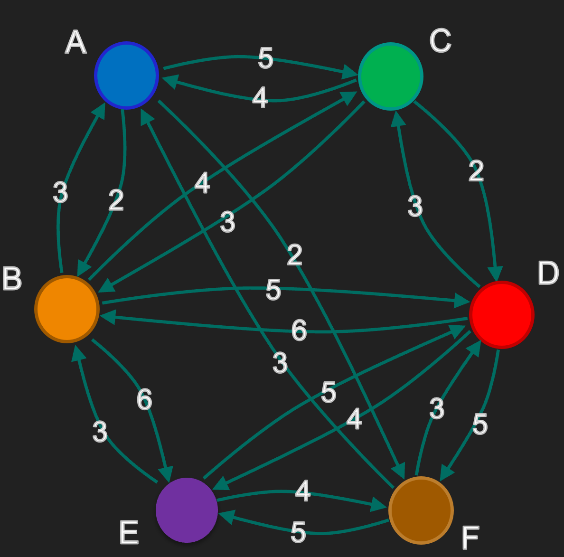
This application creates the following types of graphs:

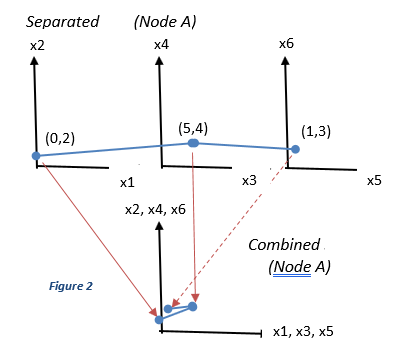
* Separated Collocated Paired Coordinates (S-CPC)
* Combined Collocated Paired Coordinates (C-CPC)
* Separated Shifted Paired Coordinates (S-SPC)
* Combined Shifted Paired Coordinates (C-SPC)
* Standard Time Series (Std-TS)
* Distance Time Series (Dst-TS)
* Time SPC-CPC (T-SPC-CPC)
* Continuous Parallel Coordinates (C||C)
* Distance Parallel Coordinates (Dst-C||C)
  1. **CPC vs SPC**

To understand how these graphs can help people visualize data, we need an example. CPC and SPC works best when dealing with groups, so let’s assume *thirty* students took a survey on what majors, other than their own, they would be interested in. There are *five* different academic majors represented, with *five* students representing each major. The following table (**table 1**) represents the majors (or graphical Node) and how many students from each major said he or she said was interested in that major. The figure (**figure 1**) to the right of the table, is it’s corresponding bidirectional graph (see next page):

**Figure 1**

**Table 1**



******Before we can explain the graphs, though, let’s examine what we mean by separated and combined:

Using only row-A, Node-A outgoing in ***table 1***, we create points from every two columns of each row from the table: (0,2), (5,4), and (1,3). If there were an odd number of columns, then the last point can be created by pairing the column with itself. As we can see in ***figure 2***, three points can be represented on separate graphs or combined on the same graph (which is what we normally see).

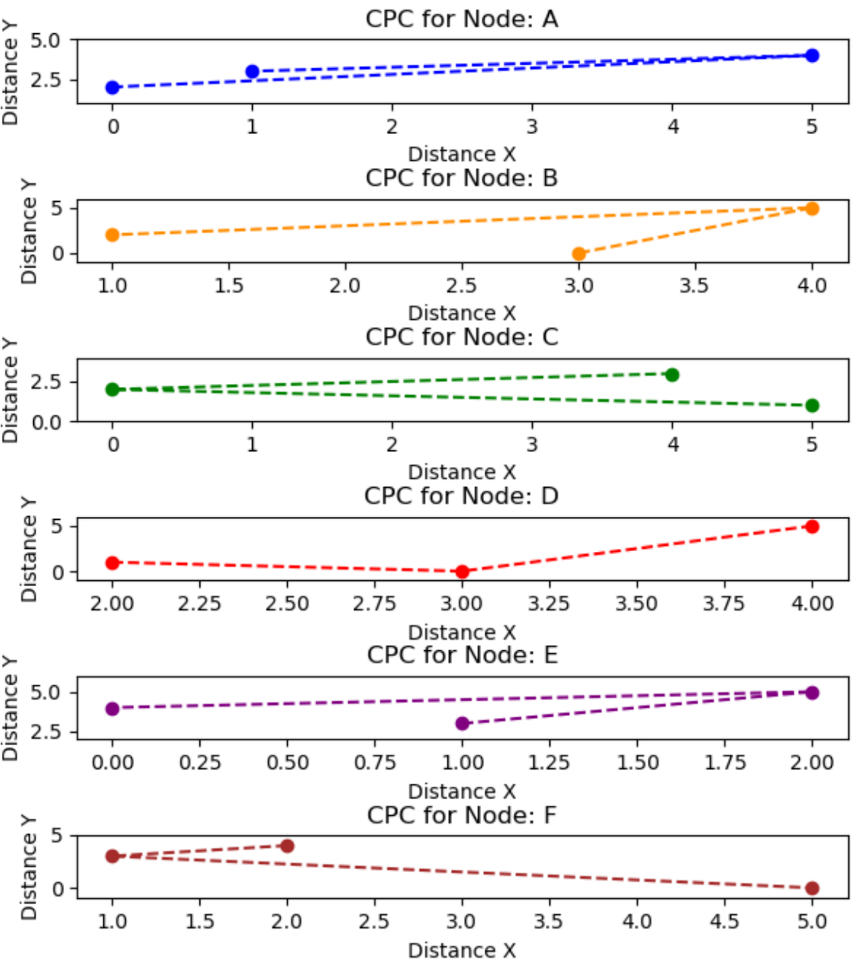
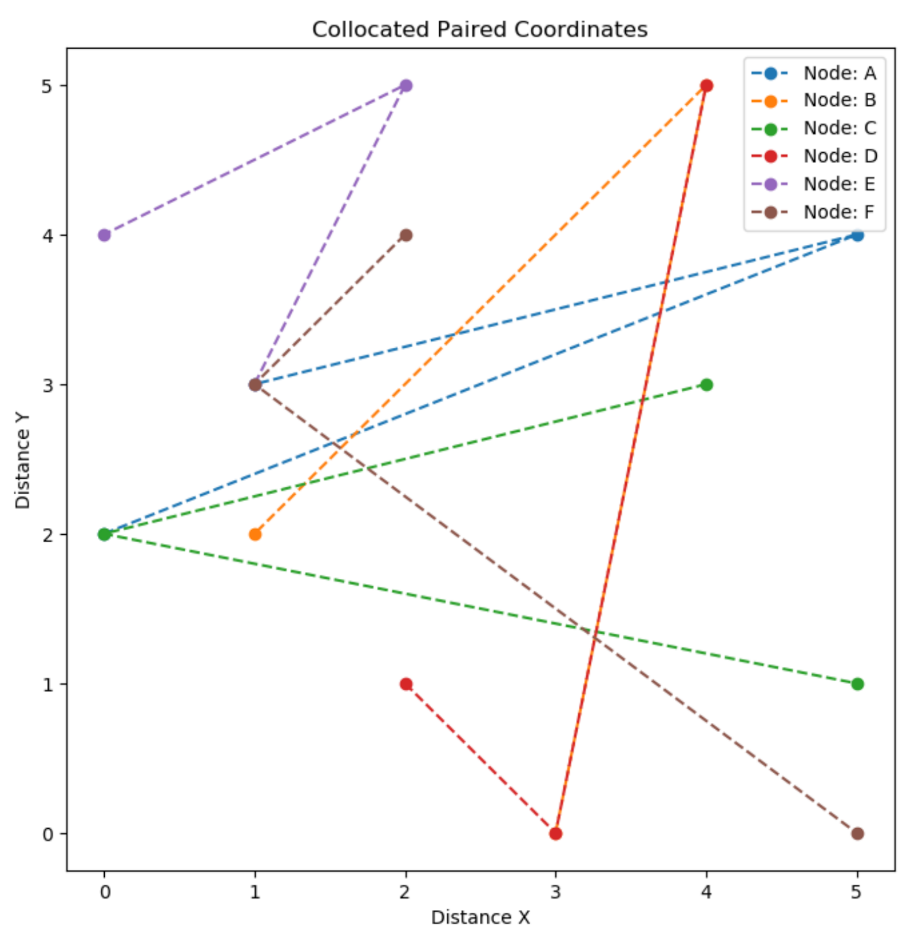
Now, let’s look at our outputted graphs:

Figure 3

This is a *Separated CPC graph*, with each Node (or major) represented separately. Note that in this example, the units of measurement are adjusted, so we can see the general pattern of changing interests.

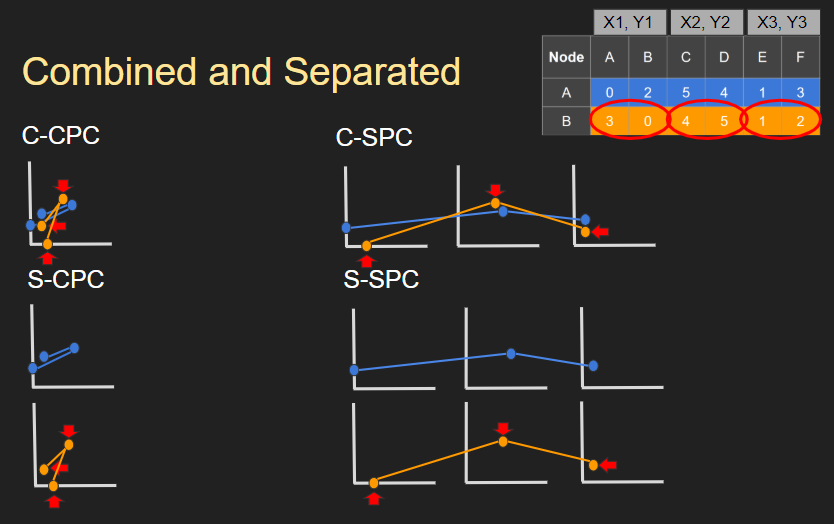
Although not the same, Nodes *B and E* are like each other.

Let’s see a combined version of a CPC, with the same information and unit of measurement, to see how alike they really are:



**Figure 4**

Notice that Nodes B and E share the same shape, but node E represents a greater change of interest.

A Shifted Paired Coordinate (SPC) graph follows the same methodology as CPC, but each point is shifted. There are several ways of shifting which we won’t focus on here. Please check out the links section of this document for examples and explanations. The focus here is how our program displays an SPC versus a CPC:

**Figure 5**

This was taken from our slide show presentation (see links). As you can see, a CPC combines each “row” of data into a single graph while an SPC keeps each row separate. Typically, this is used to separate classes of data.

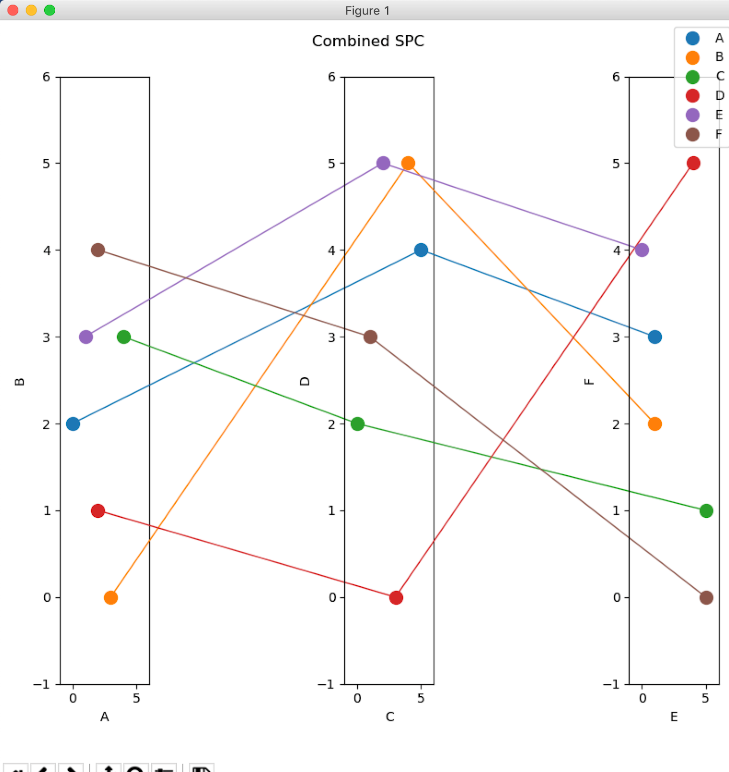


Figure 6

This what a *Combined SPC* graph looks like using our table of interests in academic majors (***table 1***).

* 1. **Time Series**

This software implements five time-related graph types:

* Standard Time Series (Std-TS)
* Distance Time Series (Dst-TS)
* Time SPC-CPC (T-SPC-CPC)
* Continuous Parallel Coordinates (C||C)
* Distance Parallel Coordinates (Dst-C||C)

The standard time series is what is used repeatedly in a physics class: a graph with a *x* and *y* axis that displays positions (or some other data) as *y* while time is represented as *x* (usually consistently incremented). Here is an example of a standard time series that’s based on the occurrence of Tuberculosis in the U.S. (1953-2009):

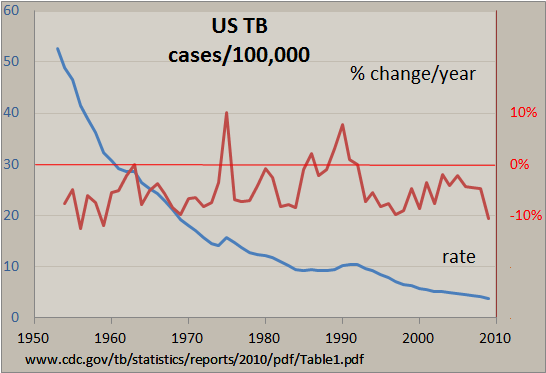


Figure 7

A Distance Time Series takes the same data that was used in a standard version, but the graph is based on the distance (or amount of change) between each point. Referring to our physics class, again, we can view this graph type as a *Displacement-*of-data graph.

The Time SPC-CPC is a hybrid of the previous graphs. Each row represents a *class, object, or container* of data while each column represents incremented time:

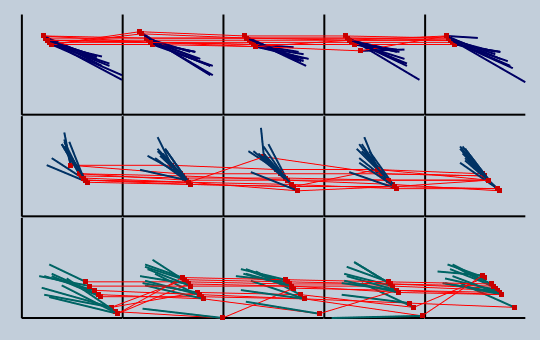


Figure 8

This graph is taken directly from our application. The data is based of the “iris.csv” file that comes with the program. Each red dot represents the current location of that strand of data while the line indicates the direction and distance that data moves.

Before we can talk about Continuous Parallel Coordinates, we need make sure we understand what a standard Parallel Coordinate graph works and looks like. In general, a Parallel Coordinate is a multi-parameterized graph that represents change for all parameters for a given time.

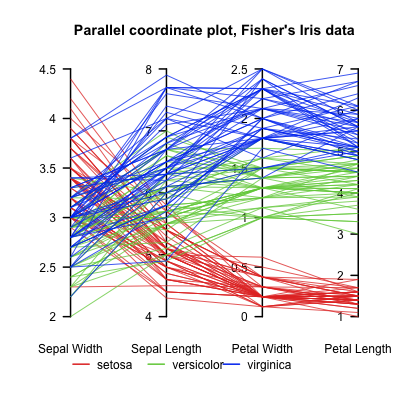


Figure 9

This is from a separate set of iris data that was used as an example on Wikipedia about Parallel Coordinates.

If this graph represented a “class” of data, however, then the continuous version would look something like this:

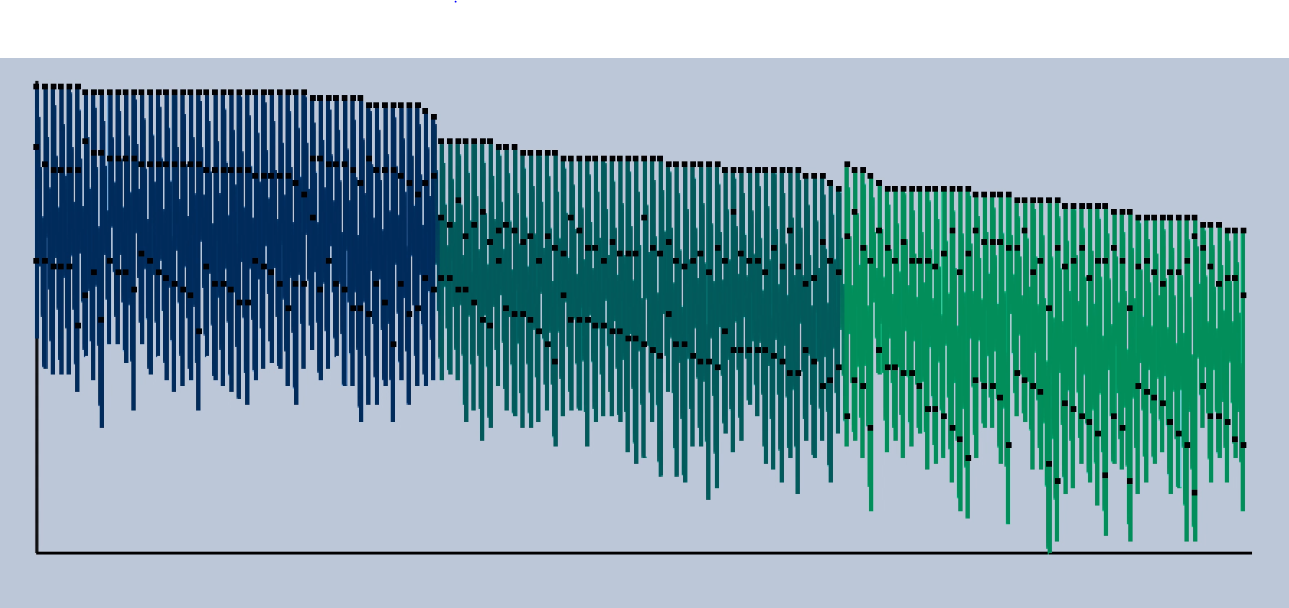


Figure 10

This is the T-SPC-CPC output from our iris data (“iris.csv”). It shows the Parallel Coordinates for each class. You may have to zoom in it see everything, however.

Like with the Standard Time Series, the distance version of the T-SPC-CPC is graphing the distance (or displacement) between points.

1. **Benchmarks**

To help us, Dr. Kovalerchuk, our client, gave us our objectives as benchmarks to complete. They are, roughly, as follows:

|  |  |
| --- | --- |
| * Read weighted graphs from file * Develop a dynamic GUI that will allow the user to alter graph properties * CPC and SPC * Separated and combined series for CPC and SPC | * Use real data with larger data-set(s) * Time series graphs * Documentation * Tests |

We completed the benchmarks, but we’re missing some GUI requirements and testing has been limited due to the changes. Thus, the reason why this is a *continuing project*.

1. **Iterative Model**

Since we’re dealing with benchmarks, it makes sense to use a development cycle that works well with them. This is our modified iterative model:

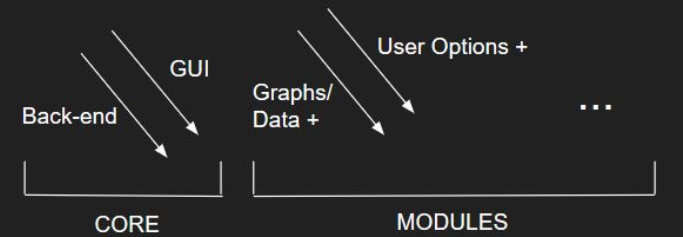
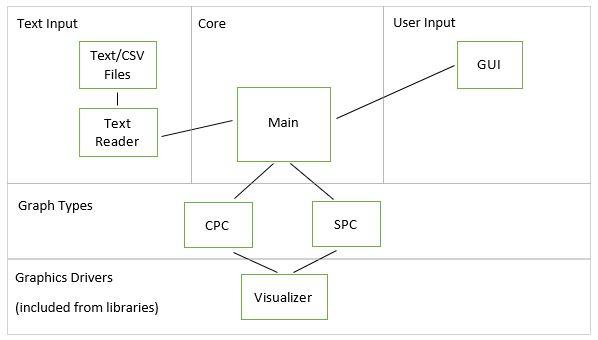


Figure 11

1. **Python**

We won’t spend long discussing something we didn’t use in the end, but our use of Python is important to our development story. Python is one of the most highly recommended languages for data-visualization. Hence, the reason we originally went with it. While it was a new language for us, it was relatively easy to learn compared to C++. There were five essential functions to our Python software: our main, the file parser, the GUI, the graphs, and the visualizer:

Figure 12



As we’ll see later, the C++ version will look similar. The GUI and visualizer were parts to the Python library: matplotlib. This meant less coding for us to do. Matplotlib (v3+) is an excellent library that provides some *dynamic options* for *static graphs* but is very limited to any dynamic modification outside of what it’s designed for. They looked great, though!

Unfortunately, one of the drawbacks of benchmark-based planning is missing out on the “big-picture” or final detail. In our case, there were some Graphical User Interface (GUI) requirements that was given to us after we had already started programming. As far we knew, these new/clarified requirements couldn’t be met with Python. Our client, who is also our supervisor, didn’t know those requirements couldn’t be met with Python, either.

Theoretically, interfacing Python with C++ should be straight-forward, since they’re both based off the original C language. The libraries we were using with Python was another matter. Perhaps there could be a library that will allow us to use matplotlib? Perhaps it could have been possible by using one of the many libraries based off matplotlib could have helped us? We researched these possibilities, but we were never able to confirm a solution. Even if there was a possibility of keeping our Python code, there were no guarantees we would still be able to meet the new requirements (from our perspective, at least). Also, we were rapidly running out of time to meet our deadline. So, we went with what we knew will work: a C++ application. We learned how SPC and CPC works with Python, which helped, but we were back to square-one with code.

1. **C++ Conversion**
   1. **Solution**

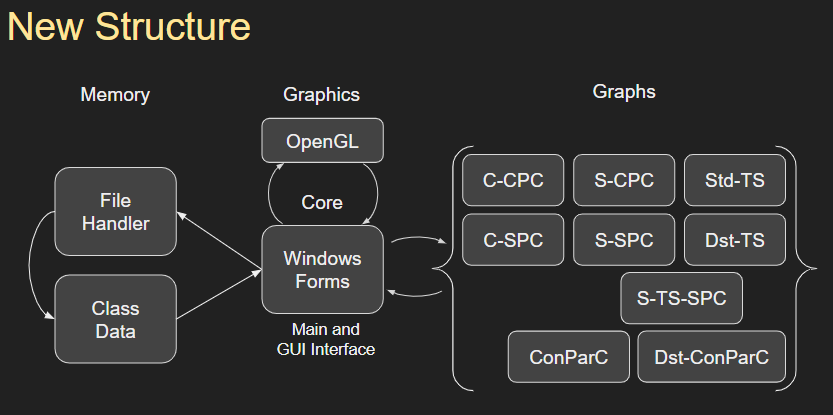


Figure 13

As mentioned, the final solution/framework is similar to the Python version. This time around, however, the GUI was not handled for us, which gave us the control we needed to implement the GUI requirements.

We used Windows Forms namespace, from the .NET Framework library, for our GUI needs and implemented OpenGL for the rest of our graphical requirements.

* 1. **Tools**

A different language means different tools:

Figure 14

|  |  |  |
| --- | --- | --- |
| **Tool** | Purpose | Description |
| **C++** | Language | Given (C++ 14, mostly) |
| **GitHub** | control | given (we were already using it) |
| **Discord** | communication | given (we were already using it) |
| **Windows Forms** | library | client provided example |
| **OpenGL** | library | we knew it (mostly) |
| **Doxygen** | documentation | we knew it (mostly) |

1. **Challenges**
   1. **General**

There were four general challenges:

|  |  |
| --- | --- |
| * Time management * Miscommunication | * The material * Tools |

Time management is something everyone deals with, and we expect no sympathy in that regard. It should be noted, though, that we never had full control of our time. This is due to our second general challenge: miscommunication. Our original project, a mobile application, was scraped due to miscommunication between the client and a third-party company that was running the database in which we were supposed to interface with. The client (we’re avoiding names here) assumed that we would be able to interface with the database this “company” was making for him. *He was wrong.* Thus, our team had to find a new project while the project deadline remained unchanged. The second instance of miscommunication was with the GUI requirements for our current client (please read Python section).

Our third general challenge was understanding the material. It’s never a good idea to program something you don’t understand, right? There were many graphing concepts in this project that were unfamiliar to us. This has led to (wait for it) some misunderstandings between us and client and cost us time. Thankfully, these misunderstanding got resolved, and we continued forward with our development.

Lastly, we had to be familiar with our tools... and we changed tools when the project changed and when the coding language changed. There is always a learning curve when using unfamiliar tools, and we were unfamiliar with Python, and, for some, GitHub. After Python, the learning curve was with the Windows Forms library, and, for some, OpenGL. Time, dedication, and teamwork resolved these challenges, every time.

* 1. **C++ Specific**

Always keep your platforms the same. We did, initially. We did not maintain that during the changes, however. We knew that if we all used the same platform, that would limit discrepancies, but we discovered an example of why this is the case: arrays are handled differently with different compilers. We still don’t know what Apple’s XCode uses to compile C++, but it lets people run a Vector Class, which is usually used as dynamic array in C++, past the number of elements that existed. Who knew?

1. **Schedule**

Up until the language change, the development schedule we setup for ourselves during the previous project remained applicable. The only changes were the meeting times and a few construction dates. That is why we’re including it in this report: It was relevant for most of our development cycle. *Please see our schedule, on* ***Page 15****, after the links page of this report.*

We don’t have a schedule for the C++ conversion. It occurred at the end of January 2019, and this document was written during the first week of March 2019. Our final product was coded in a month, and I (Paul Collet) am proud of what we did in that time. *Our weekly reports, containing what we did individually during this quarter starts immediately after the schedule (****page 16****).*

1. **Conclusion**

Using our software, we hope that our user(s) can use data visualization to make sense of the overwhelming data that they’re faced with and share what they’ve found with others. We kept our software as flexible and accessible as possible with the time that we had. That said, our client, Dr. Kovalerchuk, takes precedence in usability. We developed a plan to complete the software on time with the minimum requirements with the possibility of exceeding those requirements if time permitted. Unforeseen

1. **Links**

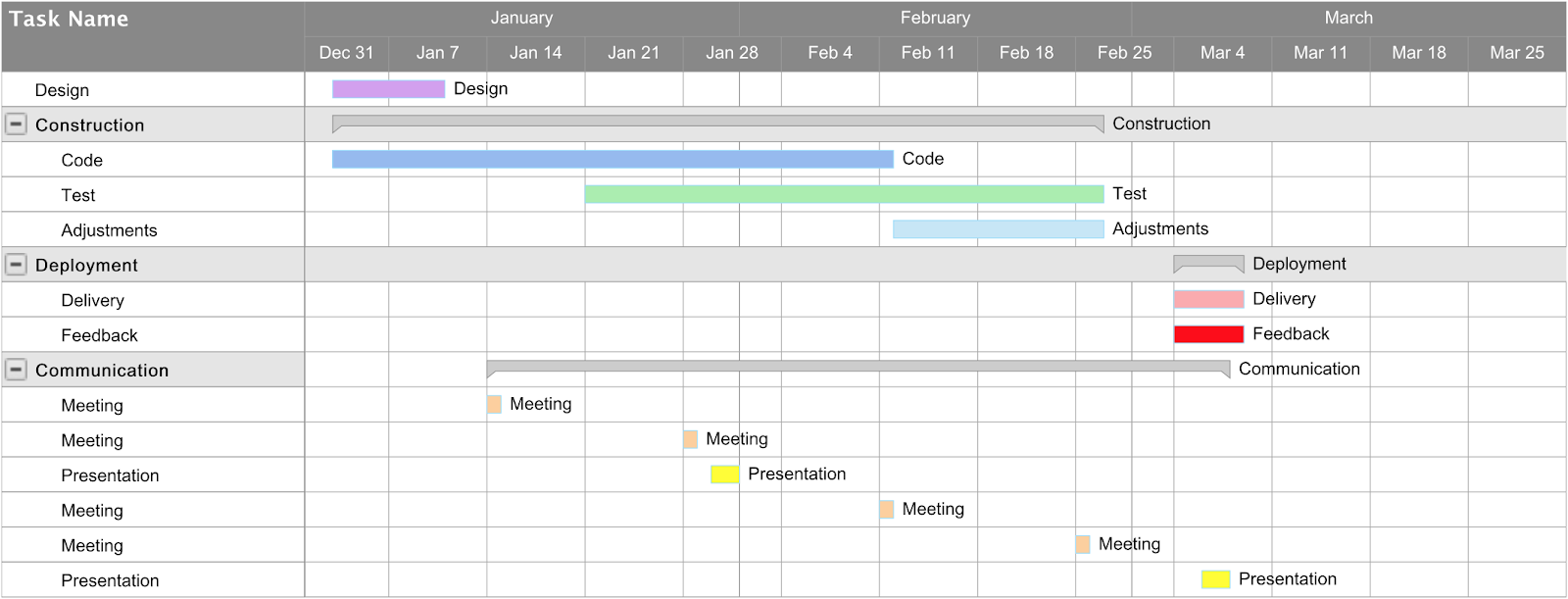
Project on GitHub: <https://github.com/StudentMobileApp/WorkingGUI/tree/NewBranch>

Dr. Kovalerchuk’s conference paper (Feb, 2014): <https://www.researchgate.net/publication/269323695_Visualization_of_multidimensional_data_with_collocated_paired_coordinates_and_general_line_coordinates>

Our final project presentation:

<https://docs.google.com/presentation/d/1b88W-IbQih2d_sK5gtbkupnN5isRTCj11gmlv9QLFkk/edit?usp=sharing>

Presentation (CPC and SPC in depth): <https://docs.google.com/presentation/d/1OTxa_wxaL_QvXyC1QyyicwzkXUTD0Kfn4KudXQ9tKhQ/edit?usp=sharing>

****

**Initial Schedule**

**SMART Weekly Report 1 (Jan 7 – 14, 2019)**

|  |  |  |
| --- | --- | --- |
| Team Member | Past Week | Next Week |
| Allyn: | tKinter GUI library for Python:  functionality, application, etc. | Coordinate with Julia to implement paper prototype |
| Julia: | Created paper prototype and general coding | Redesigning GUI based off paper prototype results and started on functional prototype |
| Mahlet: | C++ / Python testing and comparison | Help Phillipe and Owen on back-end coding |
| Owen: | Application of tKinter library and python/C++ interface methodology | Determine/create program core and  help Phillipe on back-end graph implementation |
| Phillipe: | Added csv file parsing and finished some of SPC across two graphs. | Implement code for task 3 |

**SMART Weekly Report 2 (Jan 15 – 21, 2019)**

|  |  |  |
| --- | --- | --- |
| Team Member | Past Week | Next Week |
| Allyn: | * Laid out ground work for GUI * Tested graphs on new structure |  Continue testing and adding functions to interface |
| Julia: | Finished design and begun working on prototype | Work with Allyn on prototype interface |
| Mahlet: | Testing alternative coding and methods to Phillipe’s code | * Continue to test and optimize code * Assist with given tasks from Phillipe and Paul |
| Owen: | Refamiliarize myself with dOxygen  Test code provided from other members | * Start presentation * Add documentation * More research into data set limits for Python vs C++ libraries |
| Phillipe: | Task 5a: Completed CPC and currently implementing SPC | Successfully implement a SPC graph for a given number of columns, > 4, from a table of data |

**SMART Weekly Report 3 (Jan 12 – 28, 2019)**

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Last Week’s | | This Week’s |
| Plan | Achievements | Plan |
| Allyn: | * Continue testing and adding functions to interface | * Cleaned GUI * Updated graph implementation | * Continue testing and adding functions to interface |
| Julia: | * Work with Allyn on prototype interface | * Finished paper prototype, and currently working computer prototype | * Continue work on GUI computer prototype |
| Mahlet: | * Continue to test and optimize code * Assist with given tasks from Phillipe and Paul | * Worked on mode and median for graphs | * Continue adding features |
| Owen: | * Start presentation * Add documentation * More research into data set limits for Python vs C++ libraries | * Almost finished presentation * C++ better for dynamic libraries, but unsure with changes to Python | * Get team member’s code to work for me * After presentation, I will find best solution for C++ implementation. |
| Phillipe: | * Successfully implement a SPC graph for a given number of columns, > 4, from a table of data | * Finished SPC for task each class and CPC for all classes in one CPC. | * Finish task 5. |

**SMART Weekly Report 4** (Jan 29 – Feb 3, 2019)

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Last Week’s | | This Week’s |
| Plan | Achievements | Plan |
| Allyn: | · Continue testing and adding functions to interface | · Worked on demo and schedule for presentation  · Looked through provided GitHub example | · Continue working on GUI |
| Julia: | Continue work on GUI computer prototype | Continued work on GUI computer prototype | · Finish GUI computer prototype  · Work with Allyn on integrating GUI |
| Mahlet: | · Continue adding features | · Worked on implementing switching axis | · Enable remainder (odd) data to pair with itself  · Min or max options  · Network graphs? |
| Owen: | · Get team member’s code to work for me  · After presentation, I will find best solution for C++ implementation. | · Created and managed presentation and midterm report | · Network graphs?  · Determine if we’re using C++ or not; if so:  · C++ implementation, else:  · Help Phillipe with time series |
| Phillipe: | Finish task 5. | Implemented Combined SPC (task 5) | · Reformatting code  · Fixing scales  · Start time series |

**SMART Weekly Report 5** (Feb 4 – Feb 11, 2019)

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Last Week’s **(We switched to C++)** | | This Week’s |
| Plan | Achievements | Plan |
| Allyn: | · Continue working on GUI | · Studied provided code  · Learn OpenGL  · Separated SPC | · Finish separated SPS  · Help rest of group |
| Julia: | · Finish GUI computer prototype  · Work with Allyn on integrating GUI | · Started GUI using Window Forms  · Learn OpenGL | · Finish a Window Forms GUI and post it to group  · Start on combined CPC |
| Mahlet: | · Enable remainder (odd) data to pair with itself  · Min or max options  · Network graphs? | · Learn OpenGL  · Started on separated CPC | · Finish separated CPC  · Help rest of group |
| Owen: | · Network graphs?  · Determine if we’re using C++ or not; if so:  · C++ implementation, else:  · Help Phillipe with time series | · Studied Windows Forms and researched some alternatives  · Provided thoroughly commented OpenGL graphing examples to help learning curve | · Create a Window Forms app (unless Julia beats me to it)  · Create a header file as an interface tool for consistency  · Help rest of group |
| Phillipe: | · Reformatting code  · Fixing scales  · Start time series | · File reading  · Data sorting | · Learn OpenGL  · Graphing combined SPC and in general |

Our plan is to:

· have something for Dr. Kovalerchuk to “play with” by Thursday

· start on time series by next week

· polish everything on the following week

**SMART Weekly Report 6** (Feb 12 – Feb 21, 2019)

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Last Week’s | | This Week’s |
| Plan | Achievements | Plan |
| Allyn: | · Finish separated SPS  · Help rest of group | · Helped Phillipe and Mahlet with graphs  · Helped interface OpenGL to Windows Forms | · Figure out mouse functionality with opengl and window forms.  · Clean up code |
| Julia: | · Finish a Window Forms GUI and post it to group  · Start on combined CPC | · Generic OpenGL integration  · Worked on integrating graphs with GUI (Allyn beat me to it) | · Integrate proper window scaling for OpenGL  · Integrate more buttons with OpenGL |
| Mahlet: | · Finish separated CPC  · Help rest of group | · Finalizing graphs  · Flipping axis | · Features  · time series |
| Owen: | · Create a Window Forms app (unless Julia beats me to it)  · Create a header file as an interface tool for consistency  · Help rest of group | · Got OpenGL to interface with Windows Forms with Allyn’s help  · Cleaned up code | · Improving code and GUI appearance  · Separate classes so GUI options become available. |
| Phillipe: | · Learn OpenGL  · Graphing combined SPC and in general | · Implemented Combined SPC and combined CPC | · Fix file handling so that same classes are printed on the same graph.  · Implement Time Series |

**SMART Weekly Report 7** (Feb 22 – Feb 25, 2019)

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Last Week’s | | This Week’s |
| Plan | Achievements | Plan |
| Allyn: | · Figure out mouse functionality with OpenGL and window forms.  · Clean up code | · Figured out method for mouse functionality  · Zooming and panning for some graphs | · Finish zooming and panning  · Implement graph moving  · Improve GUI design  · Work on presentation |
| Julia: | · Integrate proper window scaling for OpenGL  · Integrate more buttons with OpenGL | · Window resizing/scaling | · Finish resizing/scaling  · New SRS  · Work on presentation |
| Mahlet: | · Features  · time series | · Times series:  - Parallel Coordination (PC)  - PC w/ differences | · Finish time series implementation  · Clean up code and add features  · Work on presentation |
| Paul: | · Improving code and GUI appearance  · Separate classes so GUI options become available. | · Adapted Phillipe and Mahlet’s code for GUI  · Started on report and presentation | (Gave Allyn code overcite)  · Adapt time series for GUI (if Allyn doesn’t do it first)  · Finish most/all documentation  · Polish presentation |
| Phillipe: | · Fix file handling so that same classes are printed on the same graph.  · Implement Time Series | · Classes are now read and separated in graphs  · Started times series implementation | · Finish time series implementation  · Clean up code and add features  · Work on presentation |

**SMART Weekly Report 8** (Feb 26 – March 4, 2019)

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Last Week’s | | This Week’s |
| Plan | Achievements | Plan |
| Allyn: | · Finish zooming and panning  · Implement graph moving  · Improve GUI design  · Work on presentation | · Interfacing GUI with new graphs  · Adding panning and graphics functions | · Polish program |
| Julia: | · Finish resizing/scaling  · New SRS  · Work on presentation | · Worked on SRS | · Finish SRS  · Polish program |
| Mahlet: | · Finish time series implementation  · Clean up code and add features  · Work on presentation | · Finished Continuous Parallel Coordinates | · Finalize Continuous Parallel Coordinate requirements |
| Paul: | (Gave Allyn code overcite)  · Adapt time series for GUI (if Allyn doesn’t do it first)  · Finish most/all documentation  · Polish presentation | · Helped Allyn implement code for win-forms  · Finished most of presentation  · Started report  · Added Doxygen comments to code | · Finish documentation |
| Phillipe: | · Finish time series implementation  · Clean up code and add features  · Work on presentation | · Made fixes to the time series graphs | · Assist Mahlet  · Polish program |

Paul: I plan on giving Dr. K a couple of days to demo/proof-read our work before we present it on Thursday.