|  |
| --- |
| **Final Documentation** |
| **VisCanvas Data Visualization Software** |
| **Provided by:**  **Daniel Ortyn**  **Nicolas Espitia**  **Shane Vance**  **Brennan Barni** |

|  |
| --- |
| Data Visualization Group A  2-28-2018 |

Table of Contents

[Problem Statement 2](#_Toc507594957)

[Requirements 2](#_Toc507594958)

[External Interface Requirements 2](#_Toc507594959)

[User Interfaces 2](#_Toc507594960)

[Hardware Requirements 2](#_Toc507594961)

[Software Interfaces 2](#_Toc507594962)

[System Features 2](#_Toc507594963)

[Back-End Data Structures and Algorithms 2](#_Toc507594964)

[VisCanvas Graphical User Interface 3](#_Toc507594965)

[VisCanvas Visualization View Window 3](#_Toc507594966)

[Additional Nonfunctional Requirements 4](#_Toc507594967)

[Performance Requirements 4](#_Toc507594968)

[Safety/Security Requirements 4](#_Toc507594969)

[Software Quality Attributes 4](#_Toc507594970)

[Design 4](#_Toc507594971)

[High Level 4](#_Toc507594972)

[Low Level 4](#_Toc507594973)

[Programming Documentation 4](#_Toc507594974)

[Test Cases 4](#_Toc507594975)

[Case one – Basic Comma Delimited Text file Read-in Function 4](#_Toc507594976)

[Scenario two – Saving and Loading Custom File Functions 5](#_Toc507594977)

[Scenario three – Client Software Testing 5](#_Toc507594978)

# Problem Statement

The requested software from out client was to create a proprietary software to visualize and analyze multidimensional data from a comma, delimited text file. The software should be capable of multiple instances of hypercubes in respect to a specified n-dimensional coordinate; panning and zooming to navigate the display window; save progress to an output comma delimited text file to resume progress from a previous session with the software. The client heavily emphasizes a user-friendly interface, as the previous iteration of this project was lacking reasonable accessibility. The team is creating the software under callsign VisCanvas.

# Requirements

## External Interface Requirements

### User Interfaces

Client specified that the UI must be easy to understand with as little explanation as possible in order to use the functions of the software. The main window will connect the user to many of the specified functions requested by the client. These functions will be represented by buttons on the left side panel on the main window. Requirements concerning properties of the data (i.e. color; class/set names; data values) will be accessible through the right panel of the window. There will also be a secondary window that can be opened to access more data properties and preferences.

Users who are familiar with popular word processors will be immediately familiar with the options presented to them by the VisCanvas user interface. In addition, they will be able to access comprehensive help documentation through the “help” option in the menu bar at the top of the screen.

### Hardware Requirements

The software should be capable of being run offline on systems running Windows operating systems as old as Windows 7. Input taken from any I/O devices will be from a keyboard and mouse. All functions will be able to be accessed through just the mouse if necessary. The memory requirements to achieve the necessary objectives are minor when compared to most contemporary word processors or image editing software. The software should theoretically be able to be accessed by any hardware that has access to C++ and OpenGL.

### Software Interfaces

Aside from the OpenGL Graphical Framework, VisCanvas will not rely on any other plugins, software, or libraries. Everything used in the development of the product will be available via the standard C++ library and/or OpenGL. This was explicitly done to ensure portability, as well as prevent the software from being muddled down by unnecessary libraries or interfaces.

## System Features

### Back-End Data Structures and Algorithms

The underlying data structures, algorithms, and functions will parse, process, and prepare the data to be sent to the VisCanvas viewing window.

The only user input necessary in the operation of the back-end of this software will be the data files themselves. Other than that, the software will automate the process of producing a visualization from user input data. However, if the data contains characters, numbers, symbols, etc. that aren't intended for the parsing algorithm, the user will be informed that their input is invalid and instructed on how to prepare a data file for the purposes of visualization via VisCanvas.

The functional requirements are as follows:

* Data parsing algorithm
* User input function
* Functions to pass information to other parts of the software
* Extensive testing and try/catch but hunting to ensure a stable product
* File Browser

### VisCanvas Graphical User Interface

Provides the user with an immediately familiar and highly usable interface through which they can interact with and modify the data visualization. A very early prototype of the GUI will need to be implemented to develop the VisCanvas view window, however, so it will take slightly higher precedence in the early stages of development.

Users who are familiar with popular word processors will be immediately familiar with the options presented to them by the VisCanvas user interface. They will be able to load and save files, access all the tools provided by the software nad ideally be able to load data from previous visualizations used by the software. In addition, they will be able to access comprehensive help documentation through the “help” option in the menu bar at the top of the screen.

The requirements are as follows:

* File/Edit/View/Tools/Winodw/Help top-of-screen menu
* File browser window to choose data files to load from and save to
* Toolbar
* Data properties (panel on right side of window and through secondary properties window panel)
* Click & Drag functionality for dropping data into the software to be visualized

### VisCanvas Visualization View Window

The VisCanvas Visualization View Window will be the method by which the software delivers its interactive data visualizations to the user, and also the environment within which the user will modify data points from the visualization. Users will be able to click and drag data points in order to transform the visualization in real time.

The requirements are as follows:

* Functions to transform data points into screen coordinates, then draw lines between coordinates in real time (redrawing them as they are manipulated by the user)
* User input function to add new points and draw lines to and from them
* Functions to pass information to other parts of the software
* Extensive testing and try/catch bug hunting to ensure a stable product

## Additional Nonfunctional Requirements

### Performance Requirements

Despite any user input, the software should run at a smooth rate with no hitches in performance. Due to the relatively simplistic nature of the visualization software (from the perspective of efficient memory access), this requirement will not be difficult to meet. However, it demands that we handle memory access within C++ effectively, use the fastest possible sorting, parsing, and rendering algorithms, and maintain vigilance over potential memory leaks or other possible hits against the software's performance.

### Safety/Security Requirements

Because VisCanvas does not access any information other than what the user chooses to provide, security will not be a likely issue. However, if production of VisCanvas is to continue, Visualization Team A will need to keep potential security risks in the forefront of their design philosophy whenever adding new, potentially more invasive, functions in the future. The only real safety concern is ensuring that the data files are not altered when access to retrieve the data. Otherwise the program has no access to sensitive files and does not generate any sensitive files so security is not a concern and neither is safety.

### Software Quality Attributes

VisCanvas will be continually tested with various potential users in order to identify and categorize bugs, missing features, inefficient UI, or other potential road bumps in the production of the software. The primary quality characteristics that will be championed by Visualization Team A in the development of VisCanvas will be: robustness, usability. The software must be as ordered by Dr. Boris Kovalerchuk, but also must be robust easily testable for future development. Portability and usability will ensure that any of Dr. Kovalerchuk's students can make the best possible use of the software in their research of data visualizations. Because the intended users for VisCanvas are students and faculty of computer science, we will emphasize ease of use over ease of learning.

# Design

## High Level

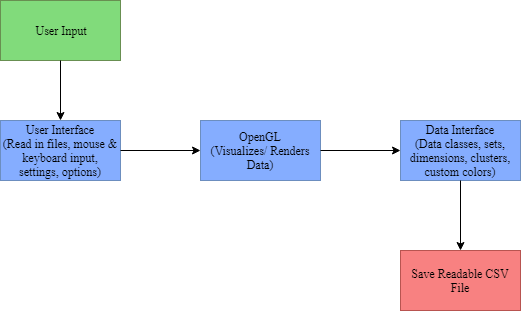


Figure 1 High Level Design of VisCanvas

Many of the functions of the software is dependent on the existence of read in data from a comma delimited file. With that said, a text file loaded by the user is the first most important input needed by the program. Once the program successfully reads the data from the file, the OpenGL class will visualize and render the data, along with saving the values of the plotted points into the Data Interface class. From here the user can continue input through mouse clicks and keyboard presses that will use the functions linked from the user interface to affect the visualization and data values until the user is satisfied with the visualized data. Once satisfied with the analysis and manipulations performed, the user can save the visualization into the form of the readable comma delimited file (which can be re-read into the software during a later session).

## Low Level

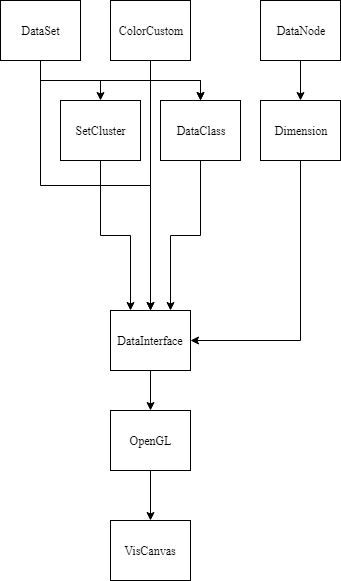


Figure 2 Low-Level Design of the classes created for the software

The final software consists of nine classes for the necessary data and functions appropriately organized for proper information access and modularity. The data read in will be placed into the dimensions, sets, and classes. Dimensions hold individual values of data where n-dimensions consist of one coordinate. Each set is a

# Programming Documentation

# Test Cases

## Case one – Basic Comma Delimited Text file Read-in Function

**Description**: The requirements specified between the client and the team specified that data will only be read in from comma delimited text files. To make sure the software can deal with faulty text files, the program performed through various text files will minor differences in format and errors. This ranged from instances such as spaces along with commas, commas with no data between them, and files with a greater amount of data than specified by the client.

**Outcome**: Performance for this scenario turned out promising. The program can successfully read in and graph data from properly comma delimited text files with no issue. When it came to text files that included spaces along with the commas, the software would not be able to read in the data. However, the necessary provisions were made to cause a “soft” error, allowing the program to continue running afterwards awaiting further instructions. It was requested by the client to be able to have options to toggle the first columns and rows of a file to be read in as dimensional fields and names of the sets respectively. This contrasts with the existing function as considering the first columns and rows to be read in as data dimensions and sets respectively.

## Case two – Saving and Loading Custom File Functions

**Description**: Ability to save load progress of a visualization created from a previous session was one of the first requirements discussed during the first phases of consultation. The plan is to create changes to the same data text file with multiple saves, each consisting of different changes performed to the data. This can include: different color values to the data and backgrounds; different degrees in vertical dimensional shifting; manual sorting of dimensions; sorting lines along different sets of data; etc. Each instance of a visualization with the same original data was created in their own software session. After loading a visualization save file to check for one to one accuracy, the software will be closed and reopened to load for the other save file cases.

**Outcome**: There were a total of five test cases, with all different degrees of changes performed with no visualizations appearing the same. All results were the same for call cases: all visualizations were maintained except for the background color. This was an intentional decision as the team deemed the background color to be of very low priority. The consensus was that the user can change the field on the fly upon the reloading of save files. During client testing however, it was requested to have the background color to be a saved feature in the data file format. The necessary changes can be easily implemented and are expected to be complete within the next few days.

## Case three – Client Software Testing

**Description**: While the team and client share a mutual satisfaction with the layout of the main window and secondary window, the software now must be tested to see if the average end user can navigate all the necessary functions with as little outside assistance as possible. The team tested with students from the computer science department. Tester selection emphasized selection from students familiar with CS 445 as they are familiar with principles regarding hypercubes. This criterion however did not make final decisions among students however, as students who still did not take CS 445 were still chosen. It should be noted that the primary user of this program is our client. Outside users are not specified

**Outcome**: On the 27th of February, we had shown our software to our client/supervisor the project. While we were able to fulfill many of the requirements requested by our client, we had our share of issues and bugs become visible once our client got their hands on the program.

The data files that were read in were reading the first columns to be the names of each set. In the client’s data, the first data was the first dimension of data for every coordinate. Ability to toggle the first column and row to be read in as names of fields or data is requested by our client.

Testing showed that navigation of the user interface was more of a feat than we wanted it to be. To start, while we did have the zoom feature in the build, the function was too discrete to effectively find for our client’s first run at the software. There is also an evident issue with highlighting a set of data. In order, to highlight a set of data, the only way to do so is to navigate with the ‘up’ and ‘down’ or ‘w’ and ‘s’ keys. Consequently, if the user wants to reach data in the middle of the text file, they would have to continuously press the keys until the desired set of data has been reached. Client kept attempting to access the data properties by looking for the properties at the top left. Unfortunately, to reach the properties, the link was located at the bottom right. This can either be remedied through either placing the button at one of the menus at the top left of the screen or making the button at the bottom left much more visible. Labeling of some fields and buttons are also necessary as some functions and fields were not very clear or it was misleading during testing.

Regarding functionality, many of the requirements requested by the user was fulfilled with little friction. While accessing functions may have been less clear than desired, they performed just as the client desired.

## Case Four – Data Properties Manipulation

**Description:** The software provides a secondary window to view and manipulate properties of multiple data groups: class, set, dimension, and cluster. The fields could be manipulated

**Outcome:**