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

|  |
| --- |
| Data Visualization Group A  2-20-2018 |

Table of Contents

[Problem Statement 2](#_Toc506909386)

[Requirements 2](#_Toc506909387)

[External Interface Requirements 2](#_Toc506909388)

[User Interfaces 2](#_Toc506909389)

[Hardware Requirements 2](#_Toc506909390)

[Software Interfaces 2](#_Toc506909391)

[System Features 2](#_Toc506909392)

[Back-End Data Structures and Algorithms 2](#_Toc506909393)

[VisCanvas Graphical User Interface 3](#_Toc506909394)

[VisCanvas Visualization View Window 3](#_Toc506909395)

[Additional Nonfunctional Requirements 4](#_Toc506909396)

[Performance Requirements 4](#_Toc506909397)

[Safety/Security Requirements 4](#_Toc506909398)

[Software Quality Attributes 4](#_Toc506909399)

[Design 4](#_Toc506909400)

[High Level 4](#_Toc506909401)

[Low Level 4](#_Toc506909402)

[Programming Documentation 4](#_Toc506909403)

[User Manual 4](#_Toc506909404)

[Test Cases 5](#_Toc506909405)

[Description 5](#_Toc506909406)

[Outcome 5](#_Toc506909407)

# 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 in order 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 and ideally export their current visualization to a PDF document. 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

## Low Level

# Programming Documentation

# Test Cases

## Scenario one – User Interface 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 of 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 criteria however did not make final decisions among students however, as students who still did not take CS 445 were still chosen.

Outcome:

## Scenario two – 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 is able to 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 software was able to run afterwards, waiting for further instructions.

## Scenario three – Saving and Loading Custom File

Description:

Outcome:

## Scenario four – Saving and Loading Custom File

Description:

Outcome: