Software Requirements Specification

for

VisCanvas Visualization Software

Prepared by: Brennan Barni,
Daniel Ortyn,
Shane Vance
Nico Espitia

The A Team (Visualization Team A)

11/20/2017

Table of Contents

Table of Contents ..... **ii**
1. Introduction **1**
1.1 Purpose 1
1.2 Document Conventions 1
1.3 Intended Audience and Reading Suggestions 1
1.4 Product Scope 1
2. Overall Description **2**
2.1 Product Perspective 2
2.2 Product Functions 2
2.3 User Classes and Characteristics 2
2.4 Operating Environment 2
2.5 Design and Implementation Constraints 2
2.6 User Documentation 2
2.7 Assumptions and Dependencies 3
3. External Interface Requirements **3**
3.1 User Interfaces 3
3.2 Hardware Interfaces 3
3.3 Software Interfaces 3
4. System Features **4**
4.1 Back-End Data Structures, Functions, Algorithms 4
4.2 VisCanvas Graphical User Interface 4
4.3 VisCanvas Visualization View Window 4
5. Other Nonfunctional Requirements **4**
5.1 Performance Requirements 4
5.2 Safety/Security Requirements 5
5.3 Software Quality Attributes 5
6. Analysis Modules **7**

1. Introduction

1.1 Purpose

This document will outline the requirements, purpose, and other vital details that are relevant to the understanding and further development of the VisCanvas data visualization software. This document will cover the entire functionality of the VisCanvas system.

1.2 Document Conventions

Unless otherwise explicitly stated, each requirement is of equal importance to the function of this software. Aside from additional quality of life features that will be added after the initial deliverable release, every requirement and feature of the software must be in at least a basic prototype state before the first release of the software.

VisCanvas: Refers to the software being developed
Text File: Text files will refer to .txt, .text and .csv

1.3 Intended Audience and Reading Suggestions

The intended audience for this document will largely be the developers of the VisCanvas data visualization software, and the interested parties including client and project supervisors. All language and terminology used will be understandable to all intended readers.

1.4 Product Scope

The VisCanvas data visualization software will take text files with comma-delimited numerical data (the same format accepted by Microsoft Excel), and output an interactive environment within which users can manipulate data points in real time. In future releases, there will also be a function allowing users to export the current configuration of their data visualization to a text file.

2. Overall Description

2.1 Product Perspective

This project will implement the first major features of the VisCanvas software while allowing for potential further development in the future. Specifically, this project will involve the development of three major features that will comprise the first functional release of the VisCanvas software: the data visualization back-end algorithms, functions, and data structures; the graphical display of the data and the graphical user interface.

2.2 Product Functions

The VisCanvas software will perform its intended function of visualizing data sets input by users by allowing users to:

- Input any text file with comma-delimited datasets.
- Experience a readable, easily understood n-dimensional data visualization constructed from their input data. In the form of line graphs.
- Reorder, and transform data points.
- Change the graph colors, and (in a subsequent release) pin notes to the visualization canvas.
- Export their data visualization in the form of a .PDF document.

For colorblind users, there will also be a colorblind option for both the data visualization canvas itself, as well as any exported .PDF visualizations.

2.3 User Classes and Characteristics

The of intended users for this project will be either our client, Professor Boris Kovalerchuk, or his students. The software and documentation will, however, be designed in such a way that any user with a basic understanding of data sets or data visualizations will be able to make use of the software.

2.4 Operating Environment

The software will be developed in C++, utilizing the OpenGL graphical framework and tested only on Windows operating systems. This will ensure the software will work on the most commonly used computers, Central Washington University computers in particular. The software will require no installer, but will instead launch from a portable .exe file. As it will take text files with comma-delimited data sets, it shares an input format with Microsoft Excel. Because of this shared input format, any dataset used in Excel will be usable in VisCanvas.

2.5 Design and Implementation Constraints

The primary constraint affecting development of VisCanvas is the relative lack of development time. As Visualization Team A are students at Central Washington University, each

member will have other obligations and deadlines to meet, so production will be constrained based on how much time can be allocated by each member for the development of VisCanvas. A secondary constraint will the addition of new requirements by the client. Which the client has stated will happen.

2.6 User Documentation

Along with the first major release of VisCanvas, there will be documentation in the form of an extensive user manual, as well as documentation in the form of a help menu. All functions, shortcuts, key bindings, etc. will be explained in detail. Potentially, if user testing suggests it would be of use, there will also be a short tutorial included in the initial release.

2.7 Assumptions and Dependencies

The entirety of the VisCanvas data visualization software operates off of the OpenGL graphical framework. Because this dependency is so fundamental, there might be unexpected or unintentional limitations in development that will have to be worked around or otherwise addressed during the development of VisCanvas. Furthermore, Visualization Team A will be performing user testing once we have established representative prototypes, and as such might add new requirements or features if user testing proves them to be necessary.

3. External Interface Requirements

3.1 User Interfaces

The user interface will be divided into five major segments. The first segment will be a panel on the left side of the window that will contain a toolbar, the second segment will be a collapsible “console”, on the bottom of the window, that will allow the display of text to the user. The third segment will be a panel similar to the first segment, but instead of tools, will be a graph key, which will allow the display of data sets to be toggled. The center of the screen will be occupied by our VisCanvas view window, and will be where the data will be displayed, as a line graph. The last segment will be a standard top-of-window bar with the typical File/Edit/View/Format/Tools/Window/Help menus.

3.2 Hardware Interfaces

Because our software will be relatively simple in design, VisCanvas should be usable on even older hardware architecture. The only hardware the software will need to access will be the keyboard and mouse, but will have options for using *only* a mouse if desired. The memory requirements of this software should be minor when compared to most contemporary word processors or image editing software. Because the software will be written in C++ and based on the OpenGL graphical framework, the software will be able to be accessed by any hardware that has access to C++ and OpenGL.

3.3 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 is to ensure portability, as well as to prevent the software from becoming muddled down by unnecessary libraries or interfaces.

4. System Features

4.1 Back-End Data Structures and Algorithms

4.1.1 Description

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

4.1.2 Stimulus/Response Sequences

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.

4.1.3 Functional Requirements

- Data parsing algorithm.
- User input function.
- Functions to pass information to other parts of the software.
- Extensive testing and try/catch bug hunting to ensure a stable product.
- File browser.

4.2 VisCanvas Graphical User Interface

4.2.1 Description

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.

4.2.2 Stimulus/Response Sequences

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.

4.2.3 Requirements

- File/Edit/View/Tools/Window/Help top-of-screen menu
- File browser window to choose data files to load from and save to
- Toolbar (panel on the left hand side of window)
- Collapsible console (panel on the bottom of window)
- Graph Key (panel on the right side of the window)
- Click & Drag functionality for dropping data into the software to be visualized.

4.3 VisCanvas Visualization View Window

4.3.1 Description

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.

4.3.2 Stimulus/Response Sequences

Users will be able to click and drag data points in order to transform the visualization in real time. In addition, users will be able to zoom in and out of the visualization and pan, if necessary, to get a better perspective on their data.

4.3.3 Functional Requirements

- 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.
- Zoom and pan bars, as well as keybindings and mouse controls to zoom and pan more intuitively.
- Function to produce a .PDF of the view window's contents.

5. Other Nonfunctional Requirements

5.1 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.

5.2 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.

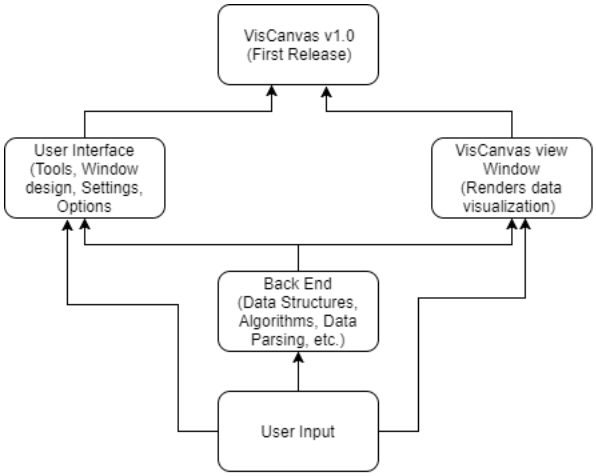
The only real safety concern is ensuring that the data files are not altered when access to retrieve the data. Otherwise the program has no access to sensitive files and does not generate any sensitive files so security is not a concern and neither is safety.

5.3 Software Quality Attributes

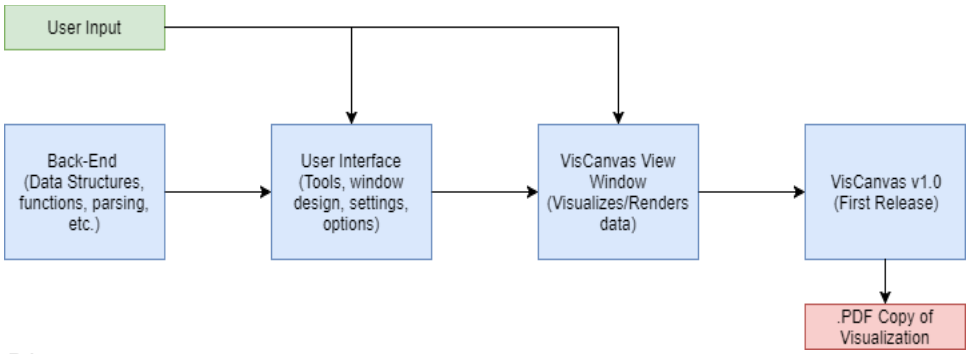
VisCanvas will be continually tested with various potential usersr in order to identify and categorize bugs, missing features, inefficient UI, or other potential road bumps in the production of the software. The primary quality characteristics that will be championed by Visualization Team A in the development 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.

6. Analysis Models

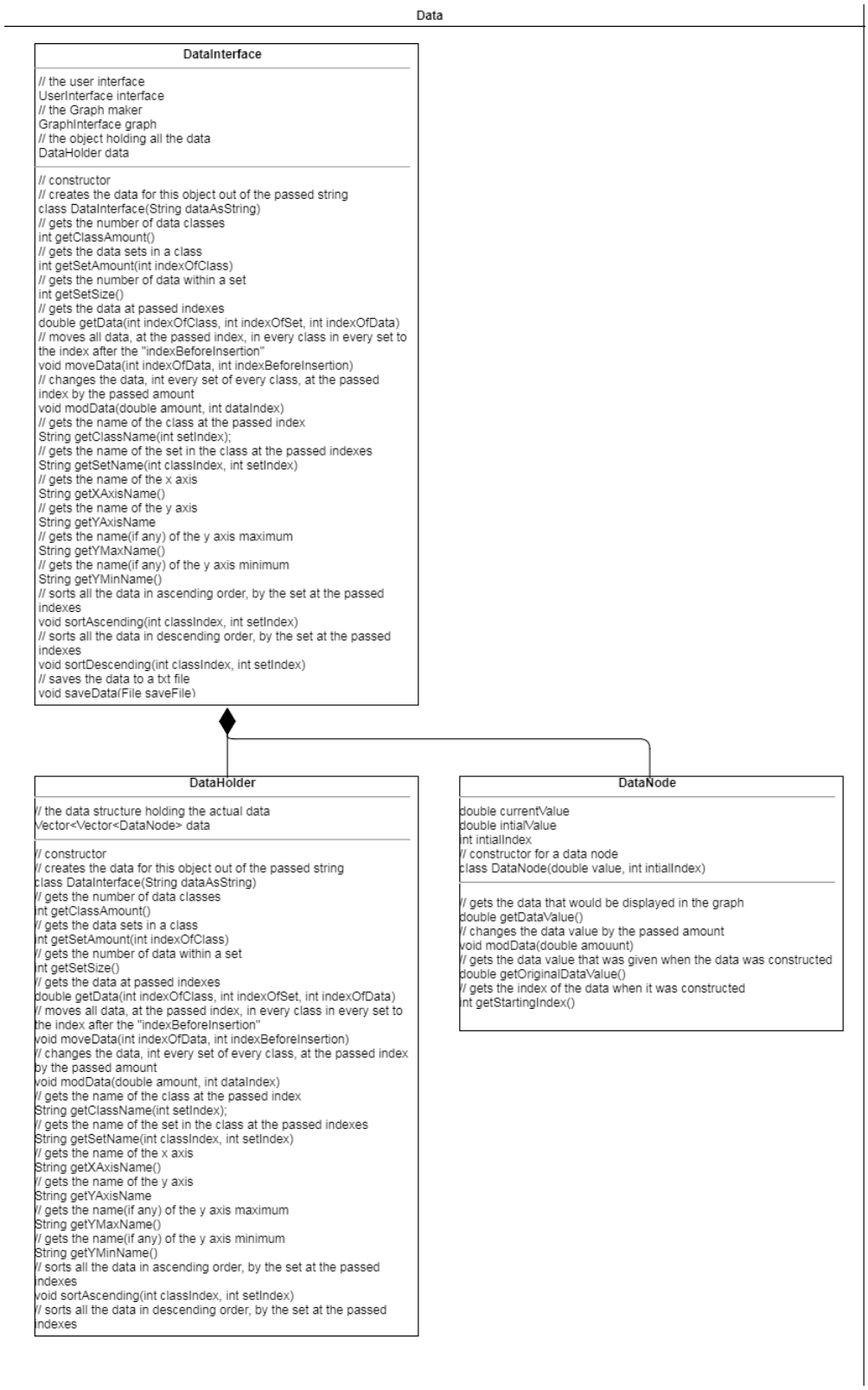
Structural Diagram:

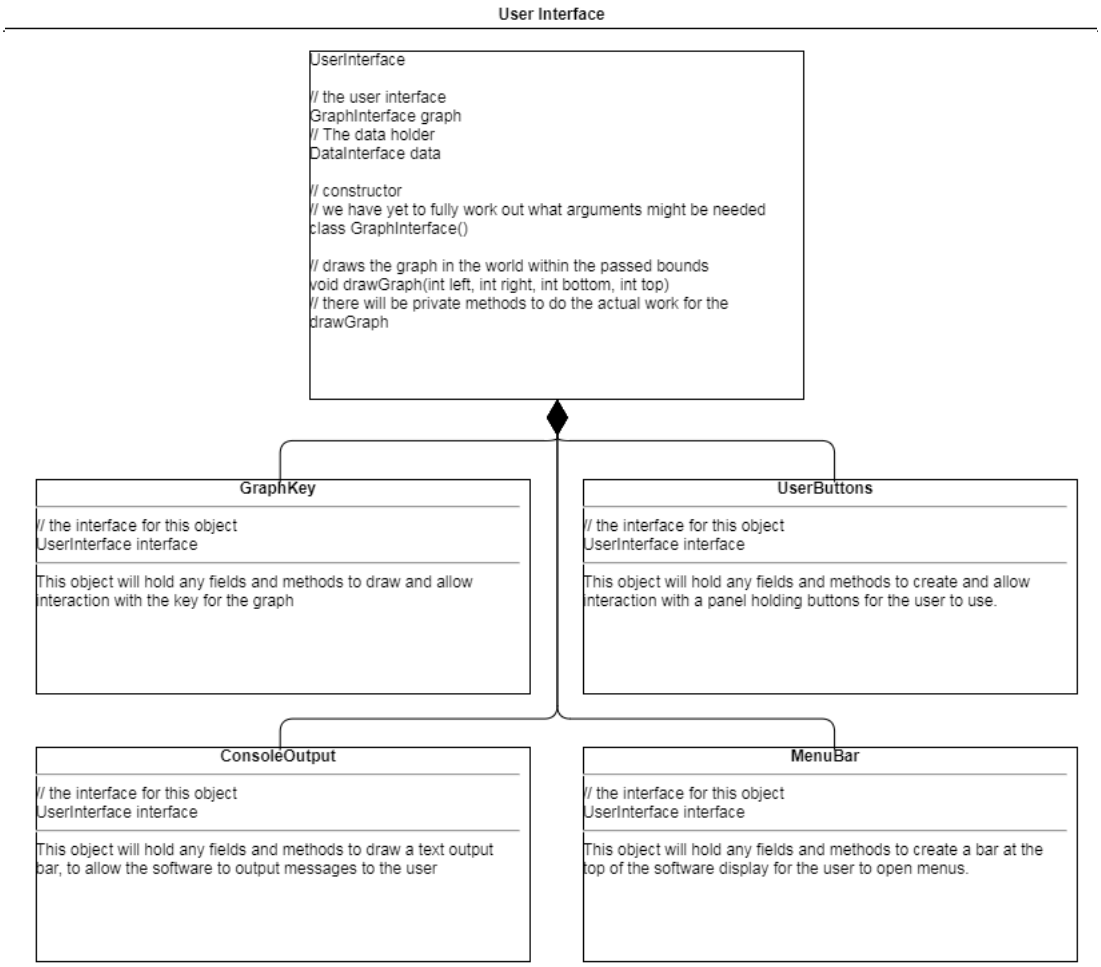


Structural Block Diagram:



UML Diagram:


Data
DataInterface
// the user interface
UserInterface interface
// the Graph maker
GraphInterface graph
// the object holding all the data
DataHolder data
// constructor
// creates the data for this object out of the passed string
class DataInterface(String dataAsString)
// gets the number of data classes
int getClassAmount()
// gets the data sets in a class
int getSetAmount(int indexOfClass)
// gets the number of data within a set
int getSetSize()
// gets the data at passed indexes
double getData(int indexOfClass, int indexOfSet, int indexOfData)
// moves all data, at the passed index, in every class in every set to the index after the "indexBeforeInsertion"
void moveData(int indexOfData, int indexBeforeInsertion)
// changes the data, int every set of every class, at the passed index by the passed amount
void modData(double amount, int dataIndex)
// gets the name of the class at the passed index
String getClassName(int setIndex);
// gets the name of the set in the class at the passed indexes
String getSetName(int classIndex, int setIndex)
// gets the name of the x axis
String getXAxisName()
// gets the name of the y axis
String getYAxisName
// gets the name(if any) of the y axis maximum
String getYMaxName()
// gets the name(if any) of the y axis minimum
String getYMinName()
// sorts all the data in ascending order, by the set at the passed indexes
void sortAscending(int classIndex, int setIndex)
// sorts all the data in descending order, by the set at the passed indexes
void sortDescending(int classIndex, int setIndex)
// saves the data to a txt file
void saveData(File saveFile)
DataHolder
// the data structure holding the actual data
Vector<Vector<DataNode> data
// constructor
// creates the data for this object out of the passed string
class DataInterface(String dataAsString)
// gets the number of data classes
int getClassAmount()
// gets the data sets in a class
int getSetAmount(int indexOfClass)
// gets the number of data within a set
int getSetSize()
// gets the data at passed indexes
double getData(int indexOfClass, int indexOfSet, int indexOfData)
// moves all data, at the passed index, in every class in every set to the index after the "indexBeforeInsertion"
void moveData(int indexOfData, int indexBeforeInsertion)
// changes the data, int every set of every class, at the passed index by the passed amount
void modData(double amount, int dataIndex)
// gets the name of the class at the passed index
String getClassName(int setIndex);
// gets the name of the set in the class at the passed indexes
String getSetName(int classIndex, int setIndex)
// gets the name of the x axis
String getXAxisName()
// gets the name of the y axis
String getYAxisName
// gets the name(if any) of the y axis maximum
String getYMaxName()
// gets the name(if any) of the y axis minimum
String getYMinName()
// sorts all the data in ascending order, by the set at the passed indexes
void sortAscending(int classIndex, int setIndex)
// sorts all the data in descending order, by the set at the passed indexes
DataNode
double currentValue
double intialValue
int intialIndex
// constructor for a data node
class DataNode(double value, int intialIndex)
// gets the data that would be displayed in the graph
double getDataValue()
// changes the data value by the passed amount
void modData(double amouunt)
// gets the data value that was given when the data was constructed
double getOriginalDataValue()
// gets the index of the data when it was constructed
int getStartingIndex()


User Interface
UserInterface
// the user interface
GraphInterface graph
// The data holder
DataInterface data
// constructor
// we have yet to fully work out what arguments might be needed
class GraphInterface()
// draws the graph in the world within the passed bounds
void drawGraph(int left, int right, int bottom, int top)
// there will be private methods to do the actual work for the drawGraph
GraphKey
// the interface for this object
UserInterface interface
This object will hold any fields and methods to draw and allow interaction with the key for the graph
UserButtons
// the interface for this object
UserInterface interface
This object will hold any fields and methods to create and allow interaction with a panel holding buttons for the user to use.
ConsoleOutput
// the interface for this object
UserInterface interface
This object will hold any fields and methods to draw a text output bar, to allow the software to output messages to the user
MenuBar
// the interface for this object
UserInterface interface
This object will hold any fields and methods to create a bar at the top of the software display for the user to open menus.