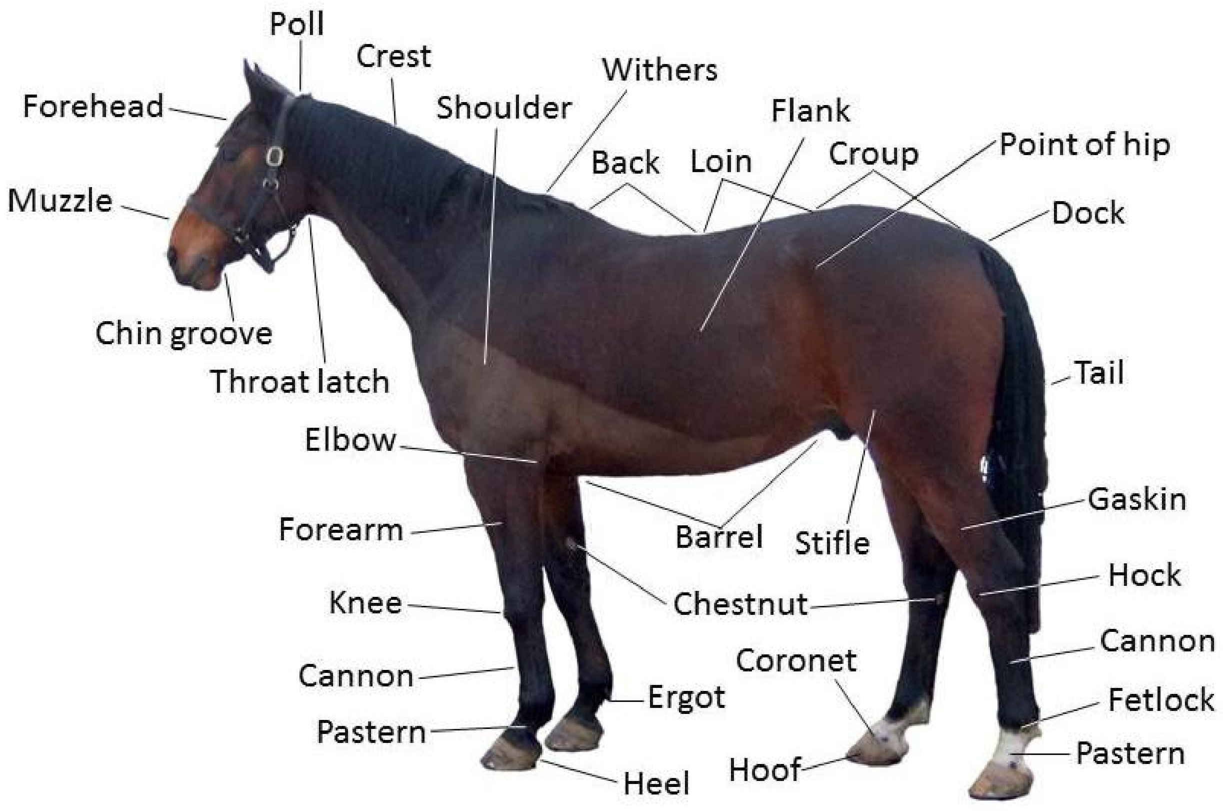
Class Diagram Tool

A tool for displaying C++ class structures



Caleb Eardley, Tim, Lybeck, Austin Lyman, Madison May

CS 4488: Capstone Project

Spring 2022

Table of contents

Abstract

THis manual describes a class diagram Tool tool for displaying C++ class structures in a UML-style diagram. It allows users to generate diagrams of class hierarchies and relationships, and offers customizable options for diagram generation and printing. The tool is implemented in C++ and features a GUI for user interaction.

1. Introduction

This is a class diagramming tool continuing off of a project by a previous group, completed as a semester project for the class CS 4488 CapStone Project. It seeks to fulfill the following objectives in relation to the previous work done.

1. Create an algorithm to organize the class structure based on hierarchy of inheritance, and types of relations, while minimizing edge crossing and improving readability.
2. Implement a feature that allows the user to generate a printable image of said graph diagram.
3. Fix previous errors in the parser, to allow class diagram generation for c++ repositories of a much larger size
4. Give the user the ability to move elements of the class diagram as they please for added customizability.

This is a continuation of a project implemented in C++ using object oriented programming, with a GUI implemented with the Dear ImGUI library and a backend implemented with DirectX11. Parsing of imputed c++ repositories is done using regex.

The current implementation allows users to search through their file system for classes directly from the GUI and specify the desired size of the graph before generation. After the graph is generated users are able to drag and drop nodes as they please, and generate a savable image from the GUI.

2. Usage

The Debug folder contains an executable of the tool for runtime purposes. On open, the executable pulls up a bank canvas and allows the user to either manually input a directory or use the directory chooser. It then allows them to generate a UML style diagram of the classes either within the directory, or within the directory and all subdirectories. It also allows them to save an image of the currently displayed class diagram.

2.1 Front End Interface

The front end interface is composed of 7 parts: the filepath textbox, the node distance dropdown, “Generate” button, “Search Subdirectories” checkbox, “Browse Folders” button, “Print to image” button, and the UML canvas.

The generate button, when clicked, will find the directory specified in the filepath textbox and check the status of the search subdirectories checkbox. If the checkbox is not checked, it will only use the specified directory. Otherwise, it will recursively search through all subdirectories. It will then generate the class diagram from all C++ classes in the directory, and display them on the UML canvas with the spacing from the node distance dropdown.

The browse folders button, when clicked, will open a file browser. It allows the user to navigate and select a directory. When selected and the ok button is clicked, it will fill the filepath textbox with the path and close the file browser. If cancel is selected, it will close the file browser without affecting the contents of the filepath textbox.

The print to image button, when clicked, will open a file browser. It allows the user to choose a directory and either choose a .bmp file or write the name of a new one to be created. When the ok button is clicked, it takes the given filename and path, generates an image that roughly mocks the currently displayed UML diagram, and saves it. If there isn’t a UML diagram currently displayed, it will save a small, blank white image.

The UML diagram shows classes by name in a grid. Clicking the classes opens a popup that shows fields and methods, in the form of +/-(public/private) name : type. Arrows between nodes show relations, with solid arrows for inheritance and open arrows with multiplicity for association. The canvas has scroll bars if the class diagram is large enough to stretch off the existing pane. Individual nodes can be dragged around as needed, arrows update automatically. The location of manually dragged nodes is retained for the print to image systems.

2.2 Input Description

The Diagram Tool uses both header and cpp files to parse UML like design from C++ classes. The file's must have a .h, .hh, .hxx, or .cpp file extension. Class names, methods, fields, scopes, multiplicity, inheritance, and access modifiers are derived using the class definition within files, and the tool handles multiple definitions per file. Parsing is separated into try catch blocks, and will fail the parsing step, but not the program if errors are encountered. Additional information about exact implementations can be found within the classes themselves.

3. Implementation and Development

3.1 Class Design

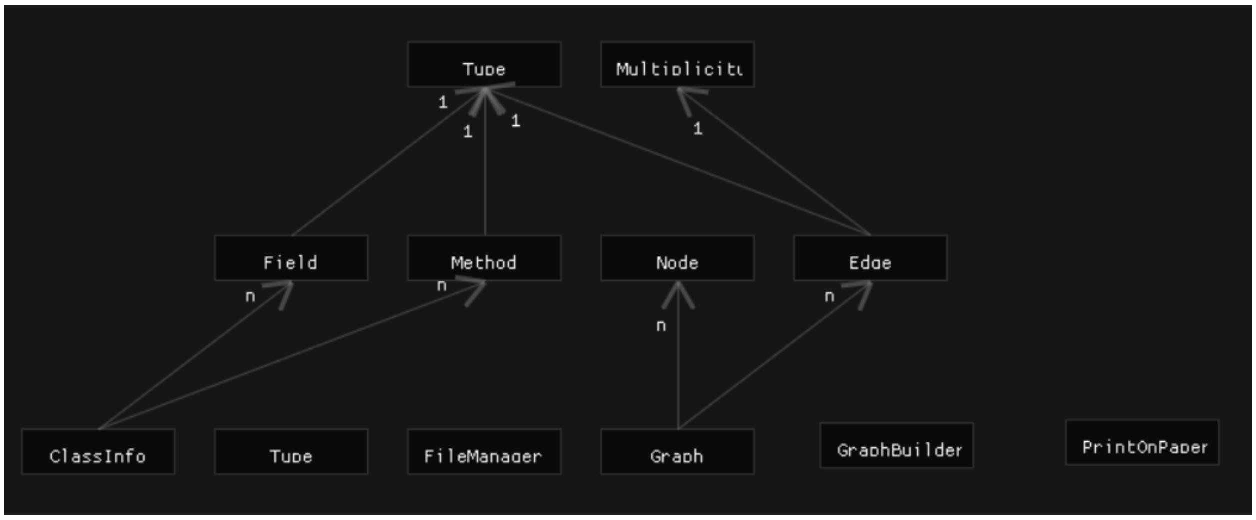


Figure 1: The class diagram tool making a class diagram of itself

The Graph, Node, and Edge classes have been templated to allow for flexibility for the next team's development and flexibility. The type and multiplicity of the edges are defined as an enumeration within the Edge class. Node methods, fields, and types are defined in separate classes.

The functions of the classes are as follows:

1. Graph is a collection of edges and nodes into vectors that can be passed to other classes as needed.

2. GraphBuilder builds the nodes and edges into a graph and passes that graph to the GUI.
3. The FileManager reads in the directory including all .hh and .cpp files in all subdirectories.
4. The ClassParser parses all class fields, such as name, methods, fields, inheritance, collections, scopes. It removes comments before doing anything, as they are considered extraneous information.
5. PrintOnPaper takes the graph information and outputs it into a BMP file with just the ClassNames in the UML objects.

3.2 Tool Usage

The tool utilizes the Dear ImGUI library, a self-contained immediate GUI library. When the second team inherited this project, it already contained the necessary files for ImGUI. They can also be imported from ImGUI's GitHub repository. ImGUI is not an event-driven library, but for the requirements of this program, it is a reasonable GUI library that supports limited user interaction. The greatest weakness of the Dear ImGUI library is the lack of documentation and examples, which may be due to how new the library is. Since the start of this project in January of 2022, the documentation has not improved.

3.3 GUI Design

The GUI design prioritizes simplicity and ease of use. Originally, other class diagram tools such as Doxygen and Visual Studio's Class Designer loosely inspired this tool's design. The color choices reflect recent popularity of "dark modes" in programming tools. After generating a class diagram, the user will see the classes in a graph sorted by inheritance. The directed arrows follow UML styling to make the relationships more intuitive without requiring the user to learn a new toolset. The organization of the classes should follow most inheritance structures, but the user can drag and drop the classes according to their preferences. Once the user is content with the layout of the diagram, they can convert it to a .bmp file as described above. The node boxes only display class names in an effort to keep the display simple, but the user can reveal fields and methods associated with each class by clicking on it. All of these choices in GUI design aim for the program to feel familiar and easy to use for users of all skill levels.

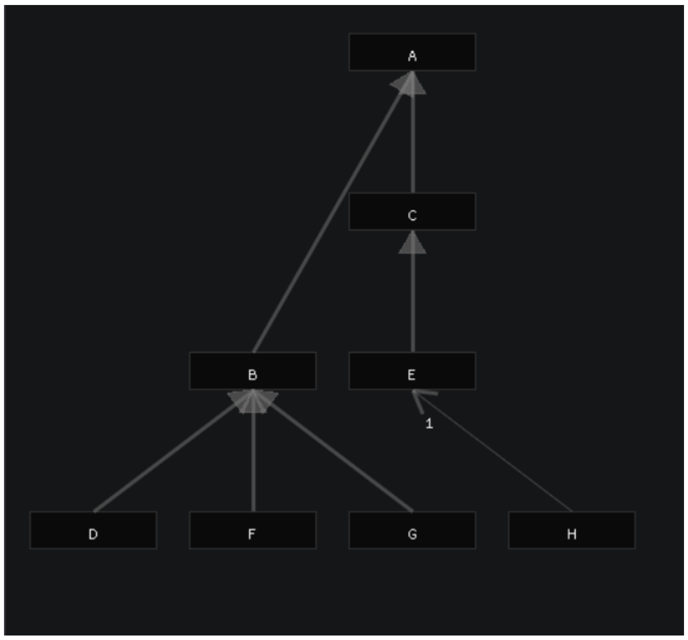


Figure 2: A sorted C++ project structure

3.4 Changes Between Groups

Group 1: Original Group : Patience Lamb, Hunter Harris, Davis Bolt.

Authors of the original project, their PDF will be left in the group_PDFs folder.

Group 2: Second Iteration : Caleb Eardley, Timothy Lybeck, Madison May, Austin Lyman

Our work was focused on:

1. Working on the algorithm to place the UML objects in an appealing manner
2. Adding the functionality of printing the Diagram to a savable file
3. Adding the ability to drag and drop UML classes in the diagram
4. Fixing the parsing to allow a larger set of valid inputs

4. Conclusions

The Class Diagram Tool is a system which will parse c++ code into an organized class diagram, arranged based on hierarchy and relations. The user is then able to move classes around the screen as they please, and generate a printable file based on the class diagram.

4.1 Summary

The Class Diagram Tool is currently able to process C++ and/or header files and create a graphical representation of the classes, fields, and methods they contain. It also identifies relationships such as multiplicity and inheritance between these elements. Since the last iteration the parser has been upgraded to handle larger code bases, although this does result in less organized graph diagrams. It attempts to organize these graphs in a hierarchical manner, and minimize the crossing edges using a local search, but also allows the classes to be moved by the user. It is also now able to be generated as a savable image

4.2 Future Work

1. Zooming abilities
2. A navigation map for more useful zooming
3. Bug in graph organization causes massive slowdown of graph generation for certain code bases
4. Continued improvement of graph algorithm to increase readability and speed
5. Implementation of orthogonal edges, rather than exclusively straight edges.
6. Port GUI to QTEdit to improve interface