Michelle Lin
mplin

.: uml designer :.

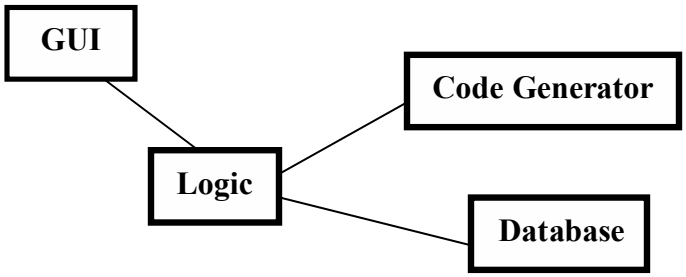
Project Description

The Unified Modeling Language is a standard used for developing models, or blueprints, for software design. Before a project can be coded, its full structure must be laid out which shows how the different parts of the project interact with one another.

uml designer will allow creation of UML class diagrams in a visual development environment. For novices at software development, can be a cumbersome process. Existing programs, such as Rational Rose used by our CS department, can be confusing and overwhelming to use.

uml designer will be different. Its target users are novice programmers and designers, i.e. undergrad CS students. The program will include a tutorial showing how to use the program, as well as a comprehensive manual explaining the different aspects that can be a part of a class diagram. Its ease of use will allow programmers to gain a deeper understanding of the design process.

System Model Diagram



GUI

- UML class diagram design canvas
  - drag and resize of objects on canvas
  - items on diagram "stick" to each other
    - if you move a class and there's an arrow pointing at it, it will extend automatically
  - ability to select a portion of diagram and move it around on canvas
  - when mouse is placed over an object or relationship arrow, bottom of canvas provides short description of item
  - text insertion and editing
  - allows viewing of multiple windows
- color-coded object menu

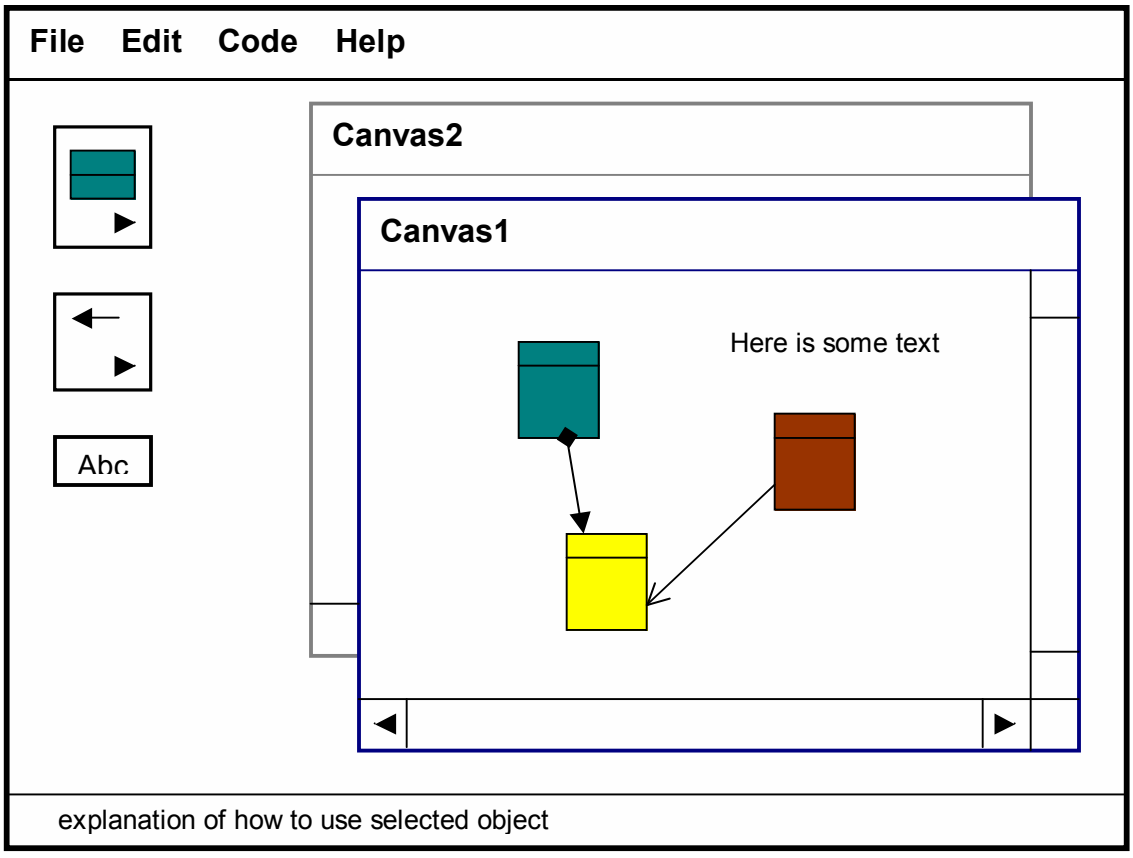
    - subclasses, superclasses, abstract classes, interfaces, etc. each represented by a different color
- relationship arrow menu
    - arrows represented with different line style
- object attribute list
    - popup window appearing on right click with list of attributes that can be assigned to class object

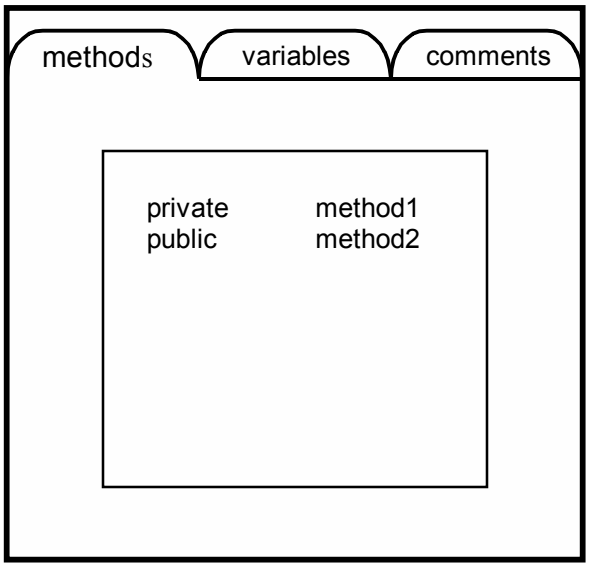
Logic
- directs user input to proper parts of the GUI
- parses user input and relays it to code generator and database
- contains rules to check for invalid relationships, class structures

Code Generator
- creates C++ code from user inputted information provided by logic
- contains rules of C++

Database
- stores UML diagrams and its associated attributes

User Interface Diagrams





The first diagram is the main user interface of uml designer. This is the where creation of uml class diagrams is done. On the left side of the panel are two menus and a text editing option. Interaction with the interface is through mouse point, click, and drag.

The menus allow for selection of object types and relationship types. To see additional object and arrows, you would click on the bottom arrow in each menu to expand the list.

There are two canvases opened in the panel, which means that multiple files may be viewed and worked on at the same time. This is an added feature from the original requirements document.

The second diagram is a view of attributes for a particular object. This is where information is stored about methods and variables within a class. Information stored in these objects are used to generate C++ code for header files.

Other graphical objects include error messages, popup manual and tutorial, and view of C++ codes.

Non-functional Requirements

Performance
- There should not be any slowdown when the canvas contains a large number of objects
- There should not be any slowdown when more than one canvas is opened, or if the tutorial or manual is opened
- Addition or movements of objects on the canvas should be executed without a noticeable lag
- There should be no slowdown when canvas screen is scrolled

Testing
- Best way is to have CS students test this
    - For those who've never used Rose, does this program make sense?
    - Is it easy to catch on and figure out how to use it?
    - If you've used Rose, is this program better?
- Get feedback on graphical interface and its ease of use
- Test code generation
- Do the attributes for objects store enough information about classes?
- Can you break the program?

Reliability
- If the program crashes while creating a diagram, it should recover last saved document. Would be ideal if it could open up with a recovered document.
- Program should never crash

Ease of Use
- Any operation should not take more than two clicks of the mouse
- Functionality should be intuitive and logically organized

Portability
- For this first version, should work on any linux machine

Documentation
- This should be done throughout the entire process from design to the completed final product
- Will enable any group member or non-group member to understand what any portion of the project is doing
- Well written documentation should make sense to any reader. All parts of project should be equally documented

Dependencies
- No outside dependencies except on the computer running the program

Updated Requirements

From class suggestions, auto-arrange is no longer a part of the basic requirements. An additional basic requirements includes multiple canvases opened at once.

Priorities

Basic requirements must be completed first. All parts of basic reqs are vital to the functionality of the program, with the most important being the GUI allowing for creation of class diagrams.

After the basic reqs are done, networking can be considered. This would include setting permissions for groups to access particular files. More advanced features could be added to the GUI, including auto-arrange, templates, and adding pictures. There may also be additional project management functionalities such as scheduling and posting messages.

A professional version may be created with added diagram types such as flow charts, tree views, etc.

Risky Features

The GUI will be the most complicated part. If it's not done in an efficient manner, there could be issues with lag in taking a long time to draw diagrams.

After suggestions in class that auto-arrange is tricky, I've taken it out the basic reqs, which probably reduced the riskiness of the project.