System Specification

David Zwerdling
zwerdlds@gmail.com

Version 1
September 23, 2023

Contents

1 Version History

Version	Date	Author	Comments
1	September 23, 2023	Zwerdling	Initial version

2 Introduction

GlyphMosaic is a graphic design program. The application facilitates a specialized graphic design workflow, in which a user supplies a source image, text, and other parameters. The application then produces a reproduction of the source image using a mosaic of textual elements from a user-supplied source.

3 System Overview

3.1 Scope

GlyphMosaic is a specialized graphic design application. It comprises the following functionality:

- Enable the user to finely specify the parameters with which the output image is generated.
- Produce the output image.

3.2 Context

The application is installed on a host system. Users interact with the system's GUI to create, load, and modify graphical mosaics.

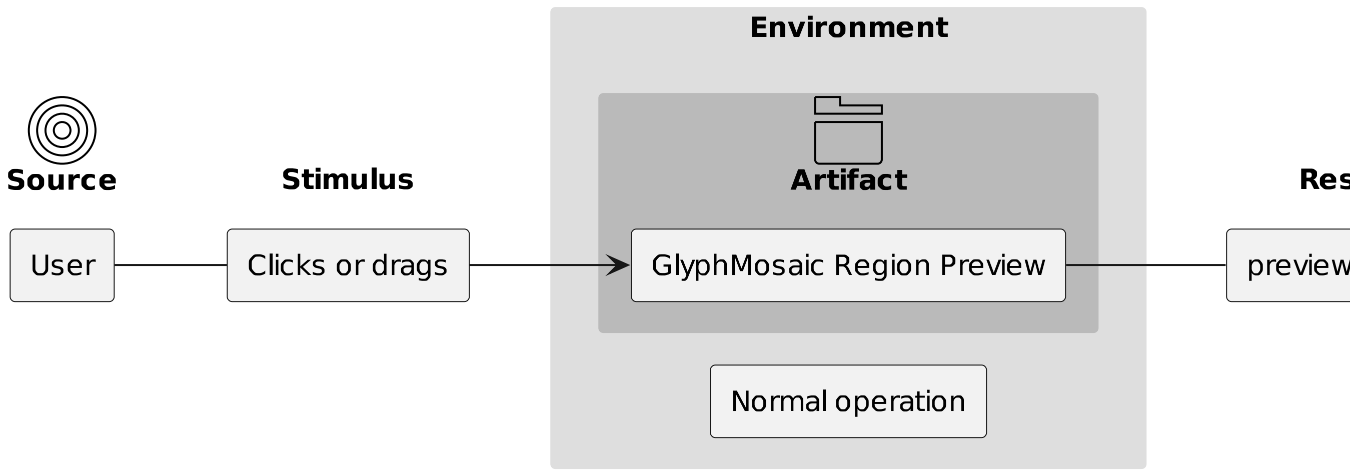

Figure 1: A hypothetical local install of GlyphMosaic and related components.

3.3 Audience

This document is intended for contributors and consumers involved in the development, maintenance, and use of the GlyphMosaic application. It aims to provide audiences with a complete description of the underlying organization of the application, facilitating insight into how all system components interact to deliver system functions. By reading this document developers can make informed decisions about how to design and implement functionalities that will deliver an optimal user experience. This document's purpose is to be a comprehensive overview of the GlyphMosaic software architecture to support all audience members involved.

3.4 Statement of Purpose

GlyphMosaic is described in this architectural document. Diagrams are also supplied to facilitate comprehension of the system internals. All aspects of the GlyphMosaic architecture, including context, modules, and code structure are described. End-user-facing functionality is also described.

This document also highlights architectural compromises and alternatives which were considered during engineering phases of the system. This document should act as a reference for any person interested in better understanding the GlyphMosaic architecture.

This report attempts to accomplish the following goals:

- Describe the complete architecture of the system.
- Describe available process alternative methods and their trade-offs.
- Provide a broad statement of use for the system itself.

4 Glossary

- **Bitmap**
  A 2-dimensional matrix of pixels, representing an image. Each pixel may contain an arbitrary number of channels. In practice, bitmaps typically contain 3 channels (in the case of source images and output images) or a single channel (in the case of region masks and path dilation masks).
- **Document**
  A file which contains all the necessary elements to recreate the output image. A user will create and modify documents for each output file they wish to create. Documents are stored and managed outside this system, by the host.
- **Glyph**
  A single textual element. Generally corresponds to a single letter, but that definition is incomplete due to complexities introduced by Unicode, which the system utilizes in the process of rendering.
- **Output Image**
  A system-rendered bitmap. Users may export output images
- **Path Dilation Masks**
  The system generates a series of dilations around the given region line kernel. This involves taking the prior iteration, or kernel in the case of the first iteration, and accreting (dilating) pixels within a given radius of existing pixels, generating the next iteration. In this method, a series of concentric lines can be generated, enabling a text path to be calculated.
- **Glyph Path Kernel**
- **Render**
- **Region**
- **Region Mask**
- **Source Image**
  A bitmap which serves as an oracle for individual pixels. Source image pixels are sampled to determine the size and color of each glyph before stamping.
- **Source Text**
  A corpus of text is specified as a source for glyphs, which are then applied to the calculated glyph data elements. These glyphs are scaled, translated, and rotated accordingly, and then stamped on a bitmap, forming the final image render.

5 Stakeholders

Individuals who have an investment in the system include:

- **Output Image Consumers**
  Individuals who view or request graphic design produced by the system.
- **Developers**
  Individuals responsible for producing the system as described in this document.
- **Graphic Designers (Users)**
  Individuals who produce graphic designs using the system.

6 Requirements

Requirements for GlyphMosaic aim to create a responsive and productive environment for its users. The GlyphMosaic System provides a specialized method to build unique graphic mosaic designs.

6.1 Functionality

GlyphMosaic's main built-in functionality is focused entirely on the workflow of creating mosaics. In the most common workflow, a user will load source image and source text into the program, adjust various settings and bitmaps to their needs, render a high-resolution version of the output mosaic, and then save that file for use outside the application.

6.2 Differentiation

Existing systems perform a similar, but limited subset, of functionality of the system. Textaizer[**?**], for example, has the ability to specify various line patterns, such as spirals. Many other examples exist that focus solely on LTR horizontal mosaics.

GlyphMosaic does not have these limitations. The creator is able to specify the glyph line pattern by specifying a region line kernel for each region. The system then determines a text path which encompasses the entire region, and develops a

6.3 Use Cases

- **System Installation**
  - ✦ Users should be able to download, install, and run the system on their own devices.
- **Content Creation**
  - ✦ Users should be able to create and modify GM documents.

This workflow is further detailed in Section 8.1.

7 System Qualities

7.1 Performance

Users should be able to rapidly receive feedback representing the current state of the project. This includes a preview of the document they wish to create.

- **Tactics**

- **Cacheing**
  Throughout the image creation pipeline, certain modules may not necessarily produce different output, given a user's input. Recomputing these results is wasteful, and avoided by implementing a cacheing layer available to other modules in the system. This method involves trading future computational load for memory space.
- **Profiling**
  Architects of the system utilize system profiling on typified workloads to determine bottlenecks in performance. These bottlenecks are then examined and the system is revised to eliminate it.

- **Scenarios**

  - **Region Drawing**
    When a user draws a region mask on the preview pane, the system should display a preview of the mask within one tenth of a second.

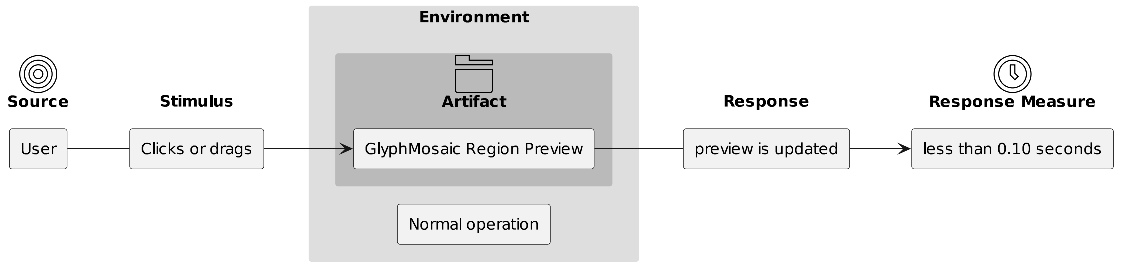

Figure 2: GlyphMosaic gives near-instantaneous feedback for region specification.

  - **Mosaic Preview "Cold" Response**
    When a user opens the preview pane, it should display a preview of the mosaic within one second.
  - **Mosaic Preview "Warm" Response**
    When a user changes a parameter in the document while the preview pane is already open and viewing a result of that change, the user should see the result of that change within one quarter of one second.
  - **Mosaic Full Render Response**
    When the user triggers a full render of the document, the system should write the result to the host filesystem within 20 seconds.

8 Architecture

8.1 Drivers

- **Design Purposes**

  - **Modeling a Specialized Workflow**
    The system is designed to facilitate a specific graphic design workflow. This workflow follows an iterative approach, similar to most graphic design workflows, in which a document is created and modified in successive phases until the designer is content with the result. Eventually the designer indicates to the system that the image should be rendered at high-resolution for output and consumption by other systems.

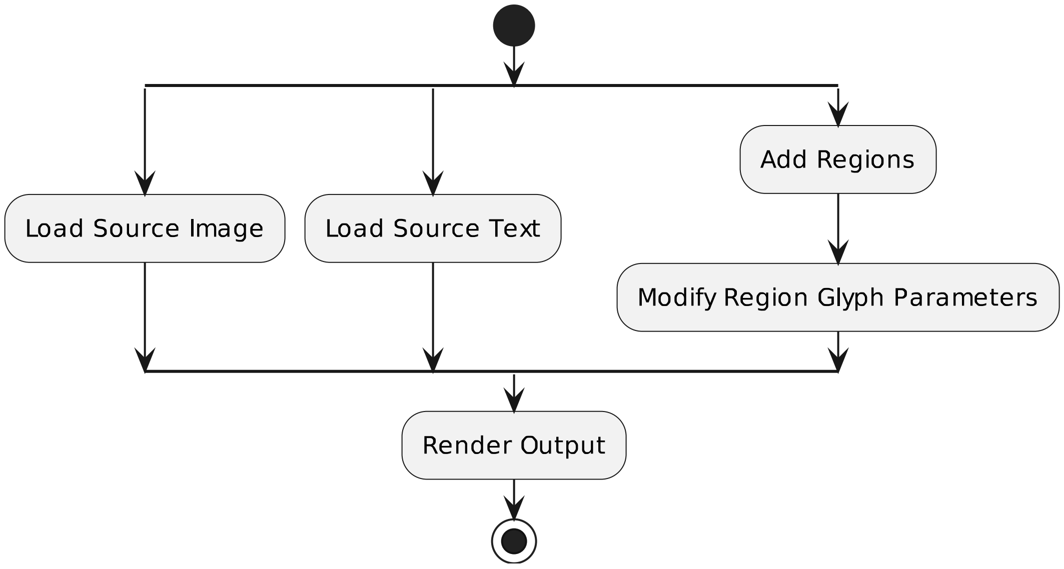

Figure 3: The system workflow models a specialized graphic design process.

- **Constraints**
  - ✦ **Resource Use Limitations**
    Devices serving GlyphMosaic may be constrained by their ability to perform the large number of computations necessary for the production of the end results. In particular, resource use may be constrained in the following aspects of the system's functions:
    - ✶ **Path Calculation**
      The system must calculate the path on which glyphs are then vectorized. This involves using the region kernel specified by the user and applying a dilation function onto it. Over successive operations, this eventually will cover the entire region. The accretion operation can be implemented using various methods. However, the highest-quality version involves creating an image kernel sized corresponding to the user-defined line distance and applying the transform over a large area.
      The system implements this calculation as detailed in Section **??**.
    - ✶ **Glyph Data**
      The position and direction of each individual glyph must then be calculated based on the calculated path. This involves walking the path and generating locations at user-specified intervals. Additionally, each glyph must calculate its scale and color by sampling the source image at the calculated location. This operation is computationally intensive due to the large number of glyphs, and must be optimized to this document's performance standards.
      The system implements this calculation as detailed in Section **??**.
    - ✶ **Glyph Rendering**
      Each glyph is stamped on a bitmap as part of the rendering process. This is constrained by the host system's ability to process potentially thousands of glyphs. Glyphs can be stamped on arbitrary bitmaps which can then be merged to form the final rendered bitmap. This pattern allows the system to trade memory (additional bitmaps) and processing power (additional glyph-data-consuming threads) for wall-clock time.
      The system implements this calculation as detailed in Section **??**.

8.2 Patterns

The application is implemented at the top level as a unified message bus. This enables decoupling between other components of the system. Typed messages are freely submitted to this bus by any component or sub-component of the system. A shared handler is used to reference the bus by other subsystems. Subsystems do not send messages directly to one another.

- **Monolithic**
  GlyphMosaic communicates with the host operating system using standard methods. Within the host operating system, the process exists within a singular executable. This simplifies potential complexity of interacting with the host operating system by reducing the abstract footprint of the application. In summary: the system need only manage a single executable element, eliminating the need to implement IPC-, or networking-related requirements. This decision does impact potential performance, as discussed in Section 8.4.
- **Layered**
  Within the monolithic system, subsystems compose into macroscopic functionality as layers. This method of design is an attempt to mitigate complexity of the system by reducing the possible interaction between sub-systems. This approach is utilized in the following locations:
  - ✦ **Test**
    Desctiption.
- **Pipe/Filter**
  In many cases, systems are composed in series to build larger functionality. This approach is utilized in the following locations:
  - ✦ **Test**
    Desctiption.
- **Model-View-Controller**
  This approach is utilized in the following locations:
  - ✦ **Test**
    Desctiption.
- **Event Bus**
  This approach is utilized in the following locations:
  - ✦ **Test**
    Desctiption.

8.3 Rationales

A few of the most enabled qualities include:

- **Performance**
- **Maintainability**

8.4 Alternative Architectures

- **Distributed Computing**
  Breaking up components of GlyphMosaic into services could enable more computational resources to be added to the system. In the future, it may be beneficial to implement a networked architecture to enable the system to scale horizontally and improve responsiveness. A distributed approach will need to consider the following aspects:

- ✦ **File Transfers**
  Preview and rendered images may be large. Operating on bitmaps in memory is substantially faster than over a network. Performance profile comparisons between implementations will need to show a clear benefit to this approach before full integration.
- ✦ **Complexity**
  Interacting with remote compute nodes introduces complexity to the system which is not related to the core functionality of the system. Adding, removing, and sending data will all need distinct components to provide a hygenic solution, which will require additional engineering effort.

8.5 Challenges and Limitations

8.5.1 Performance

The system requires computational power to produce result images. Specifically, the large number of individual glyphs represent a potential scaling issue. In addition, the large bitmaps the system may interact with may also contribute to performance degradation.

9 Architectural Views

9.1 Top-Level System

The system interacts with the following aspects of the host system:

- **Filesystem**
  The system reads source images and text from the host filesystem. Output images are written to the host filesystem. Application documents are read and written to the host filesystem.

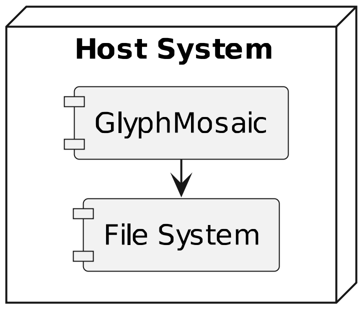

Figure 4: The GlyphMosaic system has limited responsibility from the host system perspective.

9.2 Modules

The system is composed of the following top-level modules:

- **UI**
  The user interface manages the frontend of the system. In particular, it manages the state of the UI.

The UI module publishes the following data:

- ✦ **Region Parameters**
  When a region parameter is updated by a user, the UI module is responsible for forwarding that change to the remainder of the system.
- ✦ **Bitmap Draw Events**
  The user may draw on the canvas to modify a region mask or a path line kernel.
- ✦ **File Save**
  The user may specify a location from which a file must be saved.
- ✦ **File Load**
  The user may specify a location from which a file must be loaded.

The UI must also reflect when a relevant change is made to the internal system, to allow the user to view that change. These changes are also forwarded to the document management system to ensure the change is reflected when the document is exported.

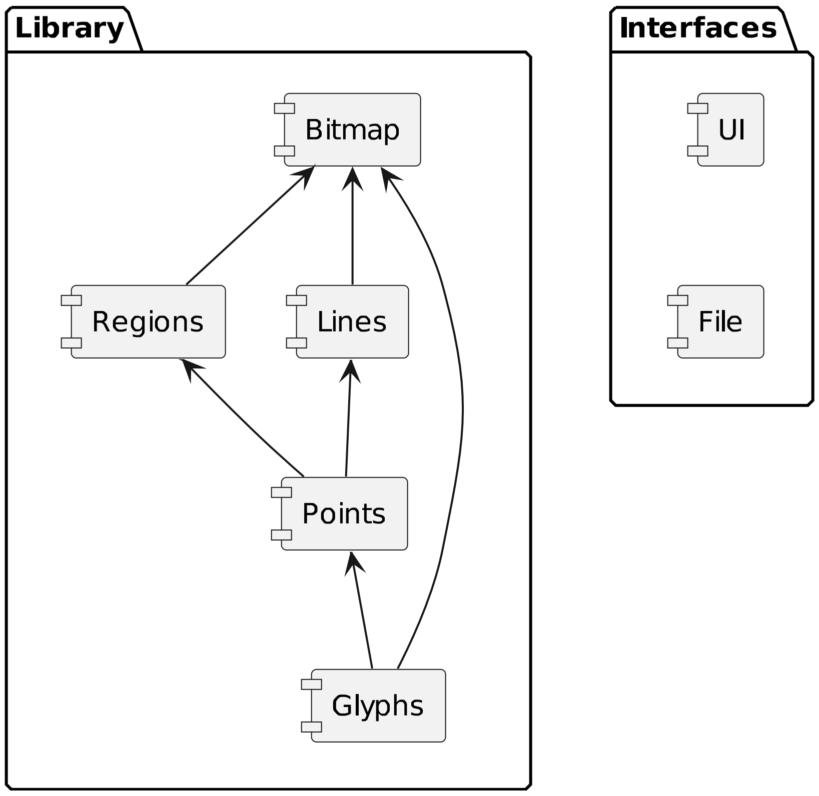

Figure 5: The top-level organization of the system.

9.3 Data Flows

Data flows between each module in such a way as to accomplish the workflow set forth in Section 8.1. This process somewhat mirrors the components described in Section 9.2, but also describes the relationships necessary to update the user interface and document.

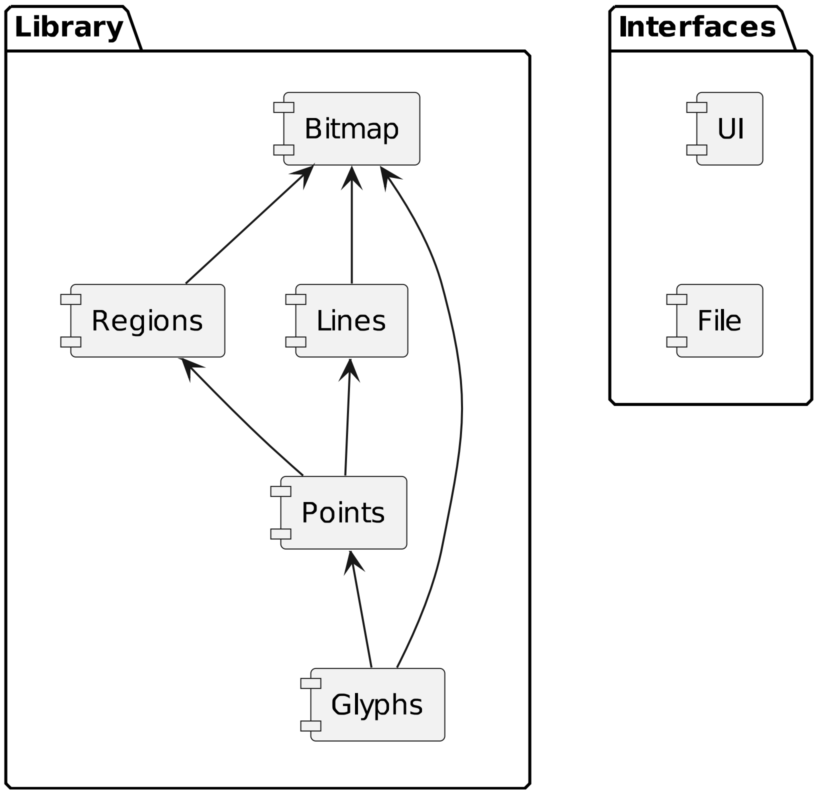

Figure 6: Succinct description.

10 Subsystems

The top-level modules described in Section **??** enumerate subsystems

Appendix A Mosaic Parameters

The process of developing mosaics necessitates user manipulation of the following parameters to ensure correct output:

- **Font**
  The specific glyph set used when stamping glyphs in the rendered image.
- **Source Image**
  Used to calculate pixel sizing.
- **Source Text**
  Used to determine which glyph is placed at the successive locations.
- **Regions**
  When a region is added, it is ‘empty’ in that it will not allow any glyph points to be added to the output image. Removing a region is a cascading delete - all parameters associated with the region are also removed.

  Each region maintains a set of additional parameters which determines the rendered glyphs:

  - ✦ **Mask**
    A 1-channel bitmap representing the area for which glyph points will be considered valid. This can be thought of as the effective area of the region.
  - ✦ **Glyph Path Kernel**
    A 1-channel bitmap representing the seed of the dilated layers which are then used to generate the region text path.
  - ✦ **Line Height**
    The distance between the lines of text on which glyphs are placed.
  - ✦ **Glyph Kerning**
    The fixed distance between each glyph.
  - ✦ **Glyph Minimum Size**
    The smallest size glyphs are scaled to. This corresponds to the size a glyph would be generated from by encountering a white pixel at its point.
  - ✦ **Glyph Maximum Size**
    The largest size glyphs are scaled to. This corresponds to the size a glyph would be generated from by encountering a black pixel at its point.

Appendix B UI Mockups

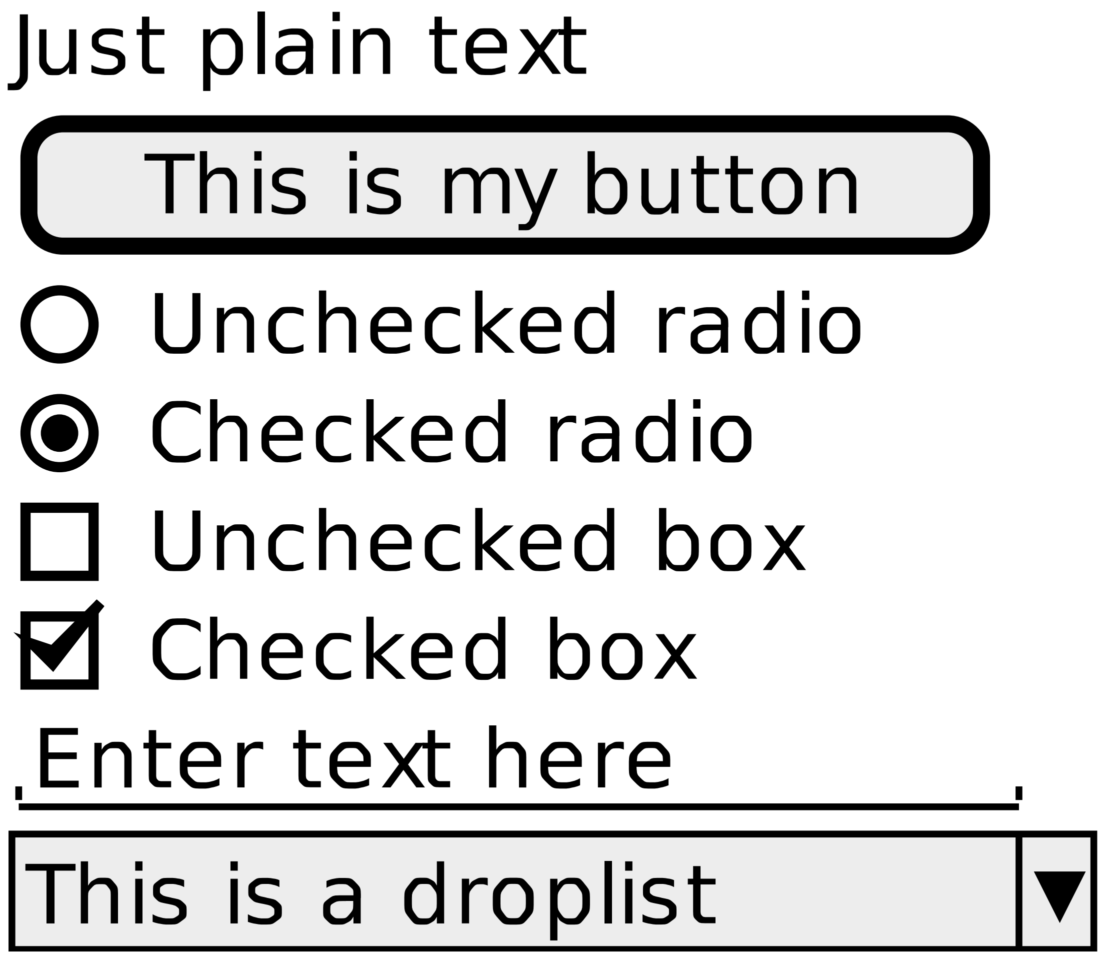

Figure 7: The main graphical user interface of the application includes a large main preview area and a tools side bar.