

Processing Programming Project

Maria Ceanuri

July 2025

1 Introduction

The purpose of this project is creating a GUI library in Processing designed to support multiple screens and reusable components (widgets). This library will allow developers to:

- Create multiple "screens" and being able to switch between them
- Add and manage both static and interactive widgets
- Access widgets by reference to retrieve or modify data

2 Features

1. Static Widgets:

- Label: Displays static text.
- Image: Displays images (logos, icons, etc.).

2. Dynamic Widgets:

- Button: Executes an action on click.
- Checkbox: Stores a checked or unchecked state.
- Checkbox List: A group of labeled checkboxes.
- Text Input: Stores user entered text, reacts to keyboard input.

3. Screens:

- Independent sets of widgets per screen.
- Ability to change: screen title, background color, visibility and state of widgets.

4. Widget referencing:

- Each widget is assigned a unique name.
- Widgets can be retrieved using a string key.
- Developers can modify the characteristics of the widget (color, text) or read its state (see if checkbox is checked).

3 Architecture

Classes	Purpose
GUIManager	Manages the current screen and input forwarding
Screen	Holds a map of widgets and their attributes
Widget (abstract)	Base class for all elements
Label, Image	Static widgets
Button, Checkbox, CheckboxList, TextInput	Dynamic widgets

4 Component Details

1. GUIManager:

- setCurrentScreen(Screen s).
- draw() — calls current screen's draw.
- handleMouse() and handleKey()

2. Screen:

- addWidget(String name, Widget w)
- getWidget(String name)
- setBackgroundColor()
- setTitle()
- draw()
- handleInput()

3. Widget (base):

- x, y, w, h
- visible
- enabled
- text, textColor, bgColor
- display(), handleInput()

Subclasses:

Class	Notes
Label	Static text
Image	Draws image from file
Button	Calls onClick()
CheckBox	toogles checked/unchecked
CheckboxList	Array of checkboxes with labels
TextInput	Stores user-typed string

5 Widget Standard Features

All widgets will support:

- setText(String text)
- setTextColor(color c)
- setBackgroundColor(color c)
- setPosition(float x, float y)
- setSize(float w, float h)
- setVisible(boolean v)
- setEnabled(boolean e)

6 Event Handling Model

Central event system will be managed by GUIManager.

Processing's mousePressed(), mouseReleased(), and keyPressed() will be forwarded to GUIManager, which will then pass input to the current screen, which passes it to widgets.

For example:

```
void mousePressed() {  
    guiManager.handleMousePressed();  
}
```

7 Data Access via References

Screen will maintain a `HashMap<String, Widget>`.

This way widgets will be added in the following manner:

```
screen.addWidget("submitButton", new Button(...));
```

And their attributes will be retrieved as follows:

```
Button b = (Button) screen.getWidget("submitButton");  
b.setText("Submit");
```

8 Screen Switching

Screen switching will be controlled by `GUIManager` in the way described below:

```
guiManager.setCurrentScreen("Settings");
```

The Title and background color will be automatically updated when the new screen becomes active and the old screen's widgets will be hidden / not drawn.

9 Folder Structure

```
Processing-GUI/  
├── README.md  
├── src/  
│   ├── GUIManager.pde  
│   ├── Screen.pde  
│   ├── Widget.pde  
│   ├── Button.pde  
│   ├── Checkbox.pde  
│   ├── CheckboxList.pde  
│   ├── Label.pde  
│   ├── TextInput.pde  
│   └── ImageWidget.pde  
├── examples/  
│   └── demo_app/  
│       ├── demo_app.pde  
│       └── assets/  
└── docs/  
    └── DESIGN_DOCUMENT.md
```

10 Project Timeline

This section outlines the structured 9-day timeline for the development of this library.

- **Day 1 – Planning & Setup**
Create the initial design document, set up a public repository, and outline the project architecture and schedule.
- **Day 2 – UML & Project Initialization**
Draft UML class diagrams, define interface structures, and set up the Processing environment with a basic sketch.
- **Day 3 – Core System Implementation**
Implement foundational classes: `GUIManager`, `Screen`, and `Widget`. Validate architecture with a simple test screen.
- **Day 4 – Interactive Widgets**
Develop dynamic widgets such as buttons, checkboxes, and text inputs. Implement event handling for mouse and keyboard input.
- **Day 5 – Midpoint Reflection**
Reassess design choices and update the UML diagram. Document deviations or improvements in a reflective write-up.
- **Day 6 – Full Feature Development**
Implement remaining features like screen switching, widget referencing, and stylization. Focus on output over polish.
- **Day 7 – Finalization & Demo Planning**
Complete all implementation features and brainstorm a comprehensive demo to showcase the GUI system.
- **Day 8 – Demo Creation**
Assemble widgets and screens into a realistic example application. Document limitations or unexpected issues.
- **Day 9 – Presentation & Retrospective**
Present the demo, write a retrospective reflection, and summarize what worked well, what didn't, and potential future extensions.

Optional days (Days 10–12) may be used for final polishing, feature extension, or deeper reflection, keeping the full timeline within a 10–12 day scope.