INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

VINÍCIUS SILVA DE SAMPAIO

EDITOR DE DESENHO GRÁFICO

CAMPOS DO JORDÃO

2025

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

VINÍCIUS SILVA DE SAMPAIO

Projeto de desenvolvimento de um editor gráfico em C++ apresentado como parte dos requisitos para aprovação na disciplina de Programação Orientada a Objetos, do curso de Análise e Desenvolvimento de Sistemas, no Instituto Federal de São Paulo no Campus de Campos do Jordão, sob orientação do professor Paulo Giovani de Faria Zeferino.

CAMPOS DO JORDÃO

2025

RESUMO

O presente trabalho apresenta o desenvolvimento de uma aplicação de desenho 2D, construída na linguagem C++ com a framework gráfica Qt. O ambiente de desenvolvimento foi a IDE Qt Creator, que provê um conjunto de ferramentas integradas. O projeto teve como objetivo principal a aplicação prática do paradigma de Programação Orientada a Objetos (POO) para criar uma aplicação gráfica de desktop completa e funcional. Foram explorados conceitos como a manipulação de eventos de interface, o gerenciamento de estado de ferramentas, a renderização gráfica 2D e o mecanismo de sinais e slots para a comunicação entre objetos. A metodologia adotada foi iterativa e incremental, com a arquitetura do software inteiramente modelada em classes, o que garantiu a organização e a modularidade do código. O processo abrangeu desde o planejamento das funcionalidades, passando pela implementação progressiva das ferramentas de desenho, até a realização de testes funcionais e de usabilidade. Os resultados demonstraram que a aplicação final é estável e cumpre os requisitos propostos, evidenciando a eficácia da abordagem orientada a objetos no desenvolvimento de software com interface gráfica.

Palavras-Chave: Programação Orientada a Objetos; C++; Qt; Design de Interface; GUI; Sinais e Slots.

ABSTRACT

This paper presents the development of a 2D drawing application, built using the C++ language and the Qt graphics framework. The development environment was the Qt Creator IDE, which provides an integrated toolset. The project's main objective was the practical application of the Object-Oriented Programming (OOP) paradigm to create a complete and functional desktop graphics application. Concepts such as user interface event handling, tool state management, 2D graphics rendering via the QPainter class, and the signals and slots mechanism for inter-object communication were explored. An iterative and incremental methodology was adopted, with the software architecture modeled entirely in classes, which ensured the code's organization and modularity. The process ranged from planning the application's features and the progressive implementation of the drawing tools, to conducting functional and usability tests. The results demonstrated that the final application is stable and meets the proposed requirements, highlighting the effectiveness of the object-oriented approach in developing software with a graphical interface.

Keywords: Object-Oriented Programming; C++; Qt; Interface Design; GUI.

LISTA DE ILUSTRAÇÕES

FIGURA 1 – *Qt Creator (2009*) 12

FIGURA 2 – Tela principal (2025) 13

FIGURA 3 – Seleção de cor (2025) 14

FIGURA 4 – Tamanho do pincel (2025) 21

FIGURA 5 – Exemplo de uso (2025) 22

LISTA DE SIGLAS

IDE	*Integrated Development Environment* (Ambiente de Desenvolvimento Integrado)
POO	Programação Orientada a Objetos
GUI	Graphical User Interface (Interface Gráfica do Usuário)

SUMÁRIO

1 INTRODUÇÃO ____ 8

1.1 Objetivos ____ 8

1.2 Justificativa ____ 9

1.3 Aspectos Metodológicos ____ 10

1.4 Aporte Teórico ____ 11

2 FUNDAMENTAÇÃO TEÓRICA ____ 12

2.1 Ferramentas e Bibliotecas Utilizadas ____ 12

2.2 Abordagem de Programação Orientada a Objetos ____ 13

3 PROJETO PROPOSTO (METODOLOGIA) ____ 15

3.1 Considerações Iniciais ____ 15

3.2 Ferramentas Utilizadas ____ 16

4 AVALIAÇÃO ____ 17

4.1 Condução ____ 17

4.2 Resultados ____ 19

4.3 Discussão ____ 23

5 CONCLUSÃO ____ 24

REFERÊNCIAS ____ 25

GLOSSÁRIO ____ 26

1 INTRODUÇÃO

O desenvolvimento de aplicações de software é uma área que integra a programação com campos criativos como o design de interação e a experiência do usuário. Dentro desse universo, projetos que oferecem ferramentas intuitivas para a criação visual costumam gerar experiências produtivas e engajadoras, o que serviu como um dos pilares para a construção deste trabalho.

Este relatório apresenta o processo de desenvolvimento de um "Editor de Desenho Básico", uma aplicação gráfica 2D. O projeto foi integralmente construído na linguagem C++, com o suporte da framework Qt para a renderização e o gerenciamento da interface. O ambiente utilizado foi a IDE Qt Creator, que provê um conjunto de ferramentas integradas para o desenvolvimento com a framework Qt.

O "Editor de Desenho Básico" foi projetado para oferecer uma experiência de criação visual simples e funcional. Sua mecânica central envolve a manipulação direta do mouse para desenhar livremente sobre uma tela digital. Para expandir as possibilidades criativas, o usuário pode selecionar diferentes ferramentas, como pincel, borracha e formas geométricas, além de poder alterar atributos como a cor e a espessura do traço.

Essa estrutura de aplicação proporcionou um cenário ideal para a aplicação prática de conceitos fundamentais da programação de interfaces gráficas. Entre eles, destacam-se a manipulação de eventos de mouse (clique, arrasto e soltar), o gerenciamento de estado das ferramentas ativas, a renderização de elementos gráficos com a classe QPainter e a conexão da interface com a lógica através do mecanismo de sinais e slots.

1.1 Objetivos

Este trabalho apresenta o desenvolvimento da aplicação gráfica 2D “Editor de Desenho Básico", um projeto focado na aplicação prática da Programação Orientada a Objetos em C++. Toda a arquitetura da aplicação foi estruturada com essa abordagem, com o suporte do *framework* Qt para os elementos gráficos e de interação.

Para a sua realização, o projeto foi orientado pelos seguintes objetivos:

- Projetar uma interface gráfica de usuário intuitiva que permita o acesso rápido e claro a todas as ferramentas de desenho.
- Modelar e implementar as funcionalidades centrais da aplicação utilizando classes e objetos com a framework Qt e a linguagem C++, distribuindo as seguintes responsabilidades:
    - A criação de uma área de desenho interativa que responda aos comandos do mouse (clicar, arrastar e soltar).
    - A lógica para o desenho à mão livre, para o uso da borracha e para a alteração de cor e tamanho do pincel.
    - A funcionalidade para desenhar formas geométricas básicas, como linhas, retângulos e círculos.
    - O gerenciamento do estado da aplicação, controlando qual ferramenta está ativa no momento.
- Integrar recursos visuais do sistema operacional, como caixas de diálogo para seleção de cor e entrada de números, para criar uma experiência de usuário coesa.
- Validar o funcionamento da aplicação através de testes contínuos, garantindo a estabilidade e a usabilidade do produto final.

1.2 Justificativa

A escolha de desenvolver o "Editor de Desenho Básico" foi motivada pelo seu alto potencial de aprendizado. Uma aplicação de desenho gráfico é um excelente cenário para aplicar na prática conceitos essenciais do desenvolvimento de software de desktop, como a manipulação de eventos de interface do usuário em tempo real, a renderização gráfica 2D e o design de uma experiência interativa. Este projeto se tornou, portanto, uma oportunidade ideal para consolidar os conhecimentos em C++ através de um desafio prático e bem definido.

A decisão de estruturar todo o código com base na Programação Orientada a Objetos foi um pilar do projeto. Essa abordagem permitiu que a lógica de desenho e o gerenciamento das ferramentas fossem desenvolvidos como uma classe independente e modular (DrawWidget). Isso facilitou o desenvolvimento de forma

incremental, permitindo focar na qualidade de cada funcionalidade antes de integrá-las, o que foi fundamental para gerenciar a complexidade de uma aplicação com múltiplas ferramentas de interação.

As ferramentas foram escolhidas para criar um ambiente de desenvolvimento produtivo e focado. A linguagem C++ foi selecionada por sua performance e controle, características importantes para aplicações gráficas. A framework Qt foi usada para toda a parte gráfica e de interação por ser uma solução robusta e completa para a criação de interfaces de usuário multiplataforma, permitindo que o foco fosse dedicado à lógica da aplicação. Por fim, a IDE Qt Creator se mostrou muito eficiente, pois oferece um ambiente totalmente integrado com um designer visual de interfaces, editor de código e depurador, o que simplificou drasticamente todo o ciclo de desenvolvimento.

1.3 Aspectos Metodológicos

O desenvolvimento desta aplicação seguiu uma abordagem iterativa e incremental. O processo começou com uma etapa de planejamento, na qual foram definidos os requisitos funcionais do software: as ferramentas de desenho necessárias (pincel, borracha, formas), os controles de atributos (cor, tamanho) e o layout geral da interface do usuário. O ambiente de desenvolvimento foi então configurado com C++, a framework Qt e a IDE Qt Creator.

Com o planejamento definido, o ciclo de desenvolvimento foi guiado pelos princípios de POO. Cada sistema principal foi modelado como uma classe específica: a lógica de desenho e o gerenciamento das ferramentas foram encapsulados na classe DrawWidget, enquanto a classe MainWindow ficou responsável por gerenciar a janela e conectar os sinais da interface aos seus respectivos comportamentos. A implementação ocorreu de forma incremental, com cada funcionalidade sendo codificada e testada de forma isolada antes de ser integrada ao projeto principal.

Ao longo de todo o processo, foram realizados ciclos de testes contínuos. Esses testes serviram para validar a estabilidade da aplicação, a precisão da resposta aos comandos do mouse e o funcionamento correto de cada ferramenta. Foram realizados ajustes na interação e na usabilidade para refinar a experiência do usuário. Essa

abordagem iterativa de construir, testar e refinar foi fundamental para chegar ao produto final de forma organizada e coesa.

1.4 Aporte Teórico

A seção 2 aborda a Fundamentação Teórica. Nela, serão detalhadas as ferramentas e tecnologias utilizadas, como a linguagem C++, a framework Qt e a IDE Qt Creator. Também será apresentada a arquitetura da aplicação, com foco na abordagem de Programação Orientada a Objetos adotada e em suas principais funcionalidades.

A seção 3 descreve o Processo de Desenvolvimento. Esta seção detalha as etapas do trabalho, desde o planejamento inicial até a implementação incremental de todas as ferramentas e a construção da interface gráfica, tudo seguindo a metodologia orientada a objetos.

A seção 4 apresenta e analisa os Resultados Obtidos. Aqui, o funcionamento de cada ferramenta, a estabilidade da aplicação e a usabilidade do produto final serão avaliados e comparados com os objetivos estabelecidos para o projeto.

Por fim, a seção 5 traz a Conclusão do trabalho. Serão discutidos os aprendizados obtidos, os pontos fortes do projeto e as dificuldades encontradas durante o desenvolvimento.

2 FUNDAMENTAÇÃO TEÓRICA

O design da aplicação de desenho foi focado em criar uma experiência de usuário direta e funcional. O projeto se concentrou em traduzir ações físicas de desenho para um ambiente digital, oferecendo um conjunto de ferramentas essenciais, como pincel, borracha e formas geométricas, além de permitir a manipulação de seus atributos, como cor e tamanho. A implementação dessas funcionalidades exigiu uma arquitetura de software bem definida, que foi construída sobre os princípios da Programação Orientada a Objetos e utilizando ferramentas específicas para o desenvolvimento de interfaces gráficas.

2.1 Ferramentas e Bibliotecas Utilizadas

O projeto foi desenvolvido na linguagem C++, escolhida por sua performance e controle sobre os recursos do sistema, que são fatores importantes para aplicações gráficas. Toda a parte de interface, janelas e interação foi gerenciada pela framework Qt, selecionada por ser uma solução robusta e multiplataforma para a criação de Interfaces Gráficas de Usuário. Seus componentes, como o sistema de Sinais e Slots e o módulo QPainter, foram essenciais para a construção da aplicação.

Para o ambiente de trabalho, foi utilizado a IDE Qt Creator. O Qt Creator serviu como a principal ferramenta para a codificação, design visual da interface e depuração do código. Sendo a IDE oficial da framework Qt, ela oferece um ambiente totalmente integrado que agiliza o ciclo de desenvolvimento, desde a criação do projeto até a compilação final.

Figura 1 – *Qt Creator* (2009)

2.2 Abordagem de Programação Orientada a Objetos

A arquitetura da aplicação foi construída com base na Programação Orientada a Objetos (POO), em C++. Essa abordagem foi escolhida para organizar o código de forma modular, o que facilita sua manutenção e futuras expansões. As principais responsabilidades do software foram divididas em classes distintas, principalmente a MainWindow e a DrawWidget.

Cada classe foi projetada para encapsular dados e comportamentos específicos. A classe DrawWidget tornou-se a peça central da lógica, sendo responsável por armazenar todas as informações sobre o estado do desenho (a imagem do canvas, a cor e o tamanho do pincel, a ferramenta ativa) e por conter os métodos que processam os eventos do mouse (mousePressEvent, mouseMoveEvent, mouseReleaseEvent) e que desenham na tela (paintEvent). A classe MainWindow ficou com a responsabilidade de gerenciar a janela principal e os componentes da interface (botões, slider), conectando as ações do usuário aos métodos públicos da DrawWidget.

A dinâmica da aplicação acontece através do sistema de eventos do Qt. Diferente de um loop de jogo contínuo, a aplicação permanece em um loop de eventos,

aguardando por interações do usuário. Quando um evento ocorre — como um clique de botão ou um movimento do mouse — o Qt emite um sinal que ativa a função correspondente (um slot), que por sua vez atualiza o estado da aplicação e solicita o redesenho da tela.

3 PROJETO PROPOSTO

Seguindo a abordagem iterativa e incremental, o projeto iniciou-se com uma fase de concepção e planejamento. Nesta etapa, foram detalhadas as funcionalidades essenciais da aplicação e a experiência de usuário desejada. A mecânica de interação foi definida como o uso direto do mouse sobre uma tela digital, e o sistema de ferramentas foi estabelecido com um objetivo claro: fornecer funcionalidades para desenho à mão livre, borracha, seleção de cor, ajuste de tamanho do pincel e a criação de formas geométricas básicas.

O ponto central desta fase foi o desenho da arquitetura do software, planejada em torno dos princípios da Programação Orientada a Objetos. Foi feita uma especificação das duas principais classes do sistema e suas responsabilidades: a classe MainWindow, responsável por gerenciar a janela principal e os componentes da interface, e a classe DrawWidget, projetada para encapsular toda a lógica de desenho, o estado das ferramentas e a manipulação de eventos do mouse. Ter esse planejamento detalhado da arquitetura e das funcionalidades antes do início da codificação foi fundamental, criando um roteiro de desenvolvimento claro e servindo como um alicerce sólido para o processo.

3.1 Ferramentas e Configuração do Ambiente

Para o desenvolvimento do projeto, foram selecionadas e configuradas ferramentas específicas para garantir um processo de trabalho eficiente e estável:

- **Linguagem C++:** Foi a linguagem de programação principal, escolhida por sua performance, robustez e forte suporte ao paradigma orientado a objetos.
- **Framework Qt:** Utilizada para toda a parte de renderização gráfica, gerenciamento de janelas e de interações do usuário. Sua arquitetura baseada em sinais e slots e seus módulos para criação de interfaces foram decisivos para a escolha.
- **Qt Creator:** Serviu como o Ambiente de Desenvolvimento Integrado (IDE) para a edição, compilação e depuração de todo o código. Sua integração nativa com

a framework Qt e suas ferramentas visuais, como o Qt Designer, foram essenciais para a produtividade.

A configuração do ambiente consistiu na instalação do Qt e do Qt Creator, seguida pela criação de um novo projeto do tipo "Qt Widgets Application", que já estabelece toda a estrutura inicial necessária.

3.2 Etapas do Desenvolvimento

O processo de desenvolvimento foi dividido em três etapas principais, seguindo uma abordagem organizada e incremental.

1. Implementação da Estrutura Orientada a Objetos: Com o ambiente pronto, a primeira fase da codificação foi a criação da arquitetura do software. Seguindo o paradigma de POO, as classes MainWindow e DrawWidget foram criadas. O princípio do encapsulamento foi aplicado desde o início: a classe DrawWidget tornou-se responsável por gerenciar seus próprios dados (a imagem do canvas, a cor e o tamanho do pincel) e seus comportamentos essenciais, que foram definidos em métodos para responder aos eventos do mouse e para se redesenhar na tela.

2. Desenvolvimento Incremental das Funcionalidades: A construção da aplicação ocorreu de forma incremental, adicionando camadas de funcionalidades sobre a estrutura de classes já estabelecida. A primeira mecânica implementada foi o sistema de desenho à mão livre, incluindo a lógica para responder aos eventos de clique e arrasto do mouse. Logo após, foram adicionados os elementos de controle, como a seleção de cor e o ajuste de tamanho do pincel. Com a base da interatividade pronta, as demais ferramentas foram construídas: a borracha e as formas geométricas. Os toques finais incluíram o design da interface no Qt Designer e a conexão de todos os botões e controles aos seus respectivos slots na MainWindow.

3. Testes e Depuração: Os testes foram uma atividade constante ao longo de todo o ciclo de desenvolvimento. Cada nova funcionalidade era testada de forma isolada e integrada assim que implementada, o que permitiu a identificação e correção de erros de maneira ágil. Além dos testes funcionais, foram realizadas diversas iterações de testes de usabilidade. Nesses ciclos, foram feitos ajustes no layout da interface, na resposta visual das ferramentas e na intuitividade dos controles, refinando a experiência para que o produto final fosse coeso e polido.

4 AVALIAÇÃO

A mecânica central de desenho com o mouse foi o primeiro ponto avaliado. Os testes confirmaram que os comandos são precisos e que a renderização dos traços na tela é consistente, permitindo que o usuário desenhe de forma fluida. O sistema de ferramentas, que permite ao usuário alternar entre pincel, borracha e formas geométricas, provou ser uma regra de interação eficaz, pois incentiva a exploração criativa e adiciona uma camada de funcionalidade à aplicação.

Os elementos de personalização do desenho foram avaliados individualmente. As funcionalidades de alteração de cor, através da caixa de diálogo do sistema, e de ajuste de tamanho do pincel, através de um controle deslizante, funcionaram bem como ferramentas que oferecem controle e flexibilidade ao usuário. A ferramenta de borracha também foi um destaque: sua capacidade de apagar o conteúdo da tela com a mesma lógica do pincel criou uma experiência de usuário intuitiva e coesa.

A comunicação de informações e o feedback ao usuário foram considerados claros e funcionais. A interface, com seus botões e controles para cada ferramenta, se mostrou organizada e de fácil acesso. O feedback visual, como a mudança no cursor ou a pré-visualização da cor selecionada, complementou bem a interação, tornando a experiência de uso mais intuitiva.

Do ponto de vista técnico, a versão final da aplicação alcançou um alto nível de estabilidade. Embora diversos bugs e erros tenham sido identificados e corrigidos durante as fases de desenvolvimento e teste, o produto final não apresentou travamentos ou falhas inesperadas durante a avaliação. A performance manteve-se fluida e com uma resposta rápida aos comandos do mouse, mesmo durante o desenho de formas complexas, o que valida a eficiência da arquitetura orientada a objetos e da framework Qt.

4.1 Condução

Antes de iniciar os testes, foram definidos os critérios que guiariam a avaliação. Os principais pontos observados foram a estabilidade da performance da aplicação, a precisão do desenho em relação ao movimento do mouse, a resposta imediata da

interface aos comandos e o funcionamento correto de cada ferramenta (pincel, borracha, formas, cor, tamanho).

Os testes foram conduzidos no ambiente de desenvolvimento do Qt Creator, onde a aplicação foi executada e depurada repetidamente. Diversos cenários foram explorados para garantir a cobertura de todas as funcionalidades:

- A mecânica de desenho livre foi testada com diferentes velocidades de movimento para verificar sua consistência e fluidez.
- A interação com cada ferramenta foi forçada para assegurar que as propriedades (cor, tamanho) fossem aplicadas corretamente.
- O comportamento da aplicação ao alternar rapidamente entre as ferramentas foi observado para confirmar que o gerenciamento de estado estava de acordo com o planejado.
- Finalmente, o ciclo completo de uso foi executado várias vezes, utilizando todas as ferramentas em sequência para garantir que a experiência funcionasse de forma coesa.

Além dos testes técnicos, houve uma atenção especial à experiência do usuário. Foi avaliado se os controles eram intuitivos e se os objetivos de cada botão eram comunicados de forma clara. O feedback visual da interface também foi analisado para confirmar se contribuía positivamente para a usabilidade. Durante todo o processo, foram feitos ajustes iterativos no layout e na resposta das ferramentas, buscando uma experiência de uso equilibrada.

4.2 Resultados

Nesta seção, são apresentados os resultados obtidos durante o desenvolvimento e os testes da aplicação de desenho, com base nas funcionalidades implementadas e na análise da usabilidade. As evidências coletadas e as principais características observadas serão descritas a seguir, ilustradas por capturas de tela representativas da interface principal e da interação com suas ferramentas.

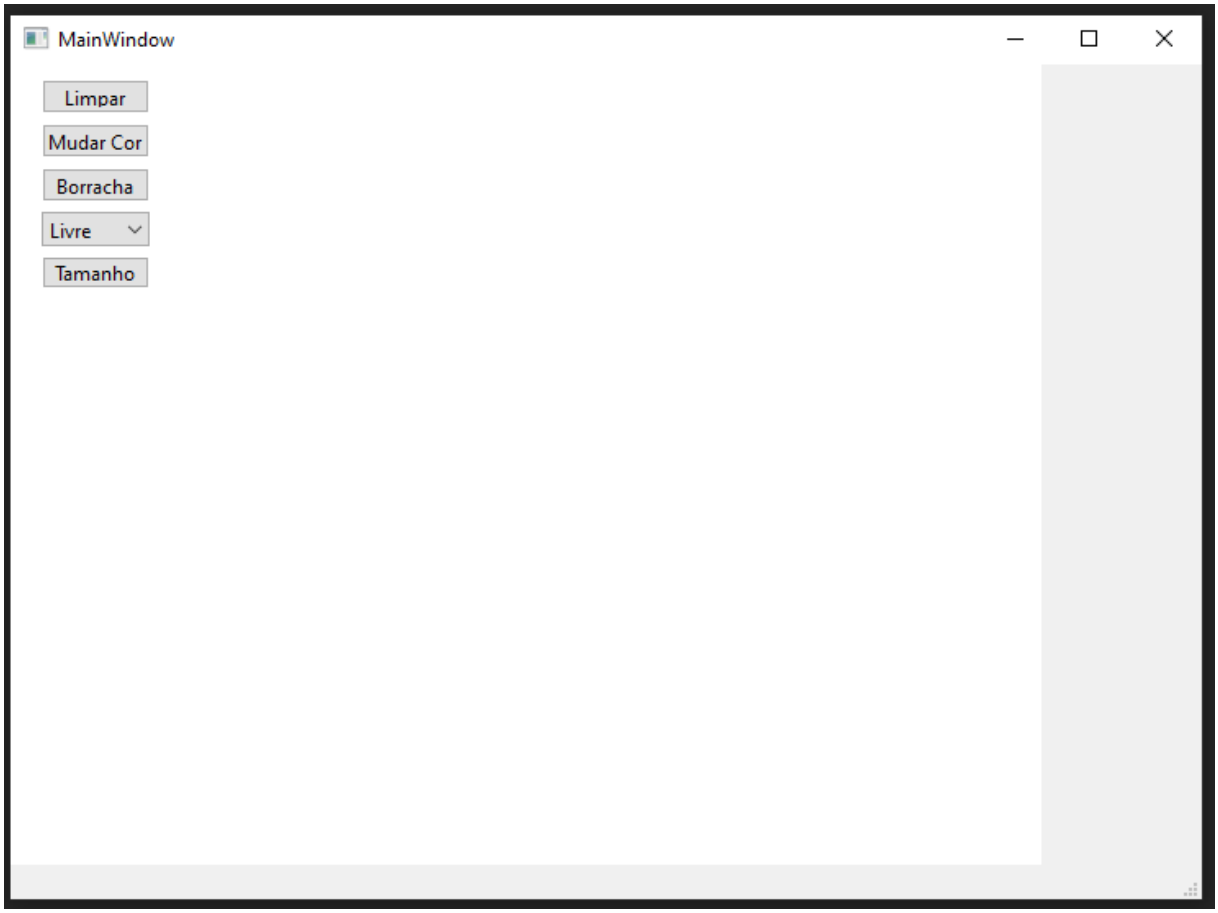

Figura 2 – Tela principal (2025)

A Figura 2 apresenta a tela principal da aplicação de desenho. O layout foi organizado de forma a priorizar a área de desenho, que ocupa a maior parte da janela. Na parte superior, uma barra de ferramentas agrupa todos os controles de forma intuitiva: botões de ação (Limpar, Cor, Tamanho), um botão para a ferramenta (Borracha) e uma caixa de seleção para as formas geométricas. Testes validaram que

a interface é carregada corretamente e que todos os elementos respondem às interações do usuário.

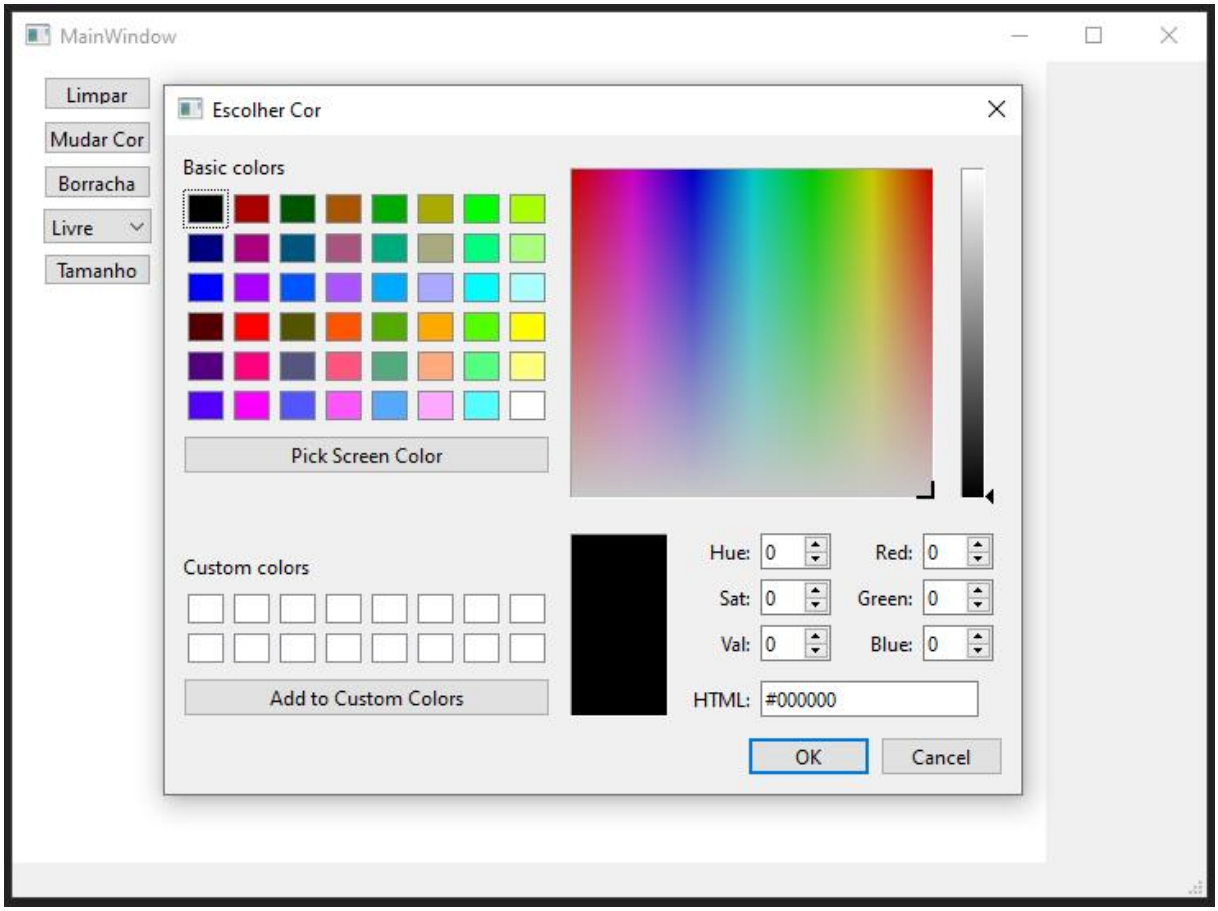

Figura 3 – seleção de cor (2025)

A Figura 3 ilustra a funcionalidade de seleção de cor. Ao clicar no botão "Cor", a aplicação invoca a caixa de diálogo padrão do sistema operacional, QColorDialog, fornecida pela framework Qt. Esta abordagem integra a aplicação de forma coesa com o ambiente do usuário. Testes confirmaram que, após a seleção e confirmação de uma nova cor nesta janela, o pincel e as ferramentas de forma passam a utilizar a cor escolhida para todos os desenhos subsequentes.

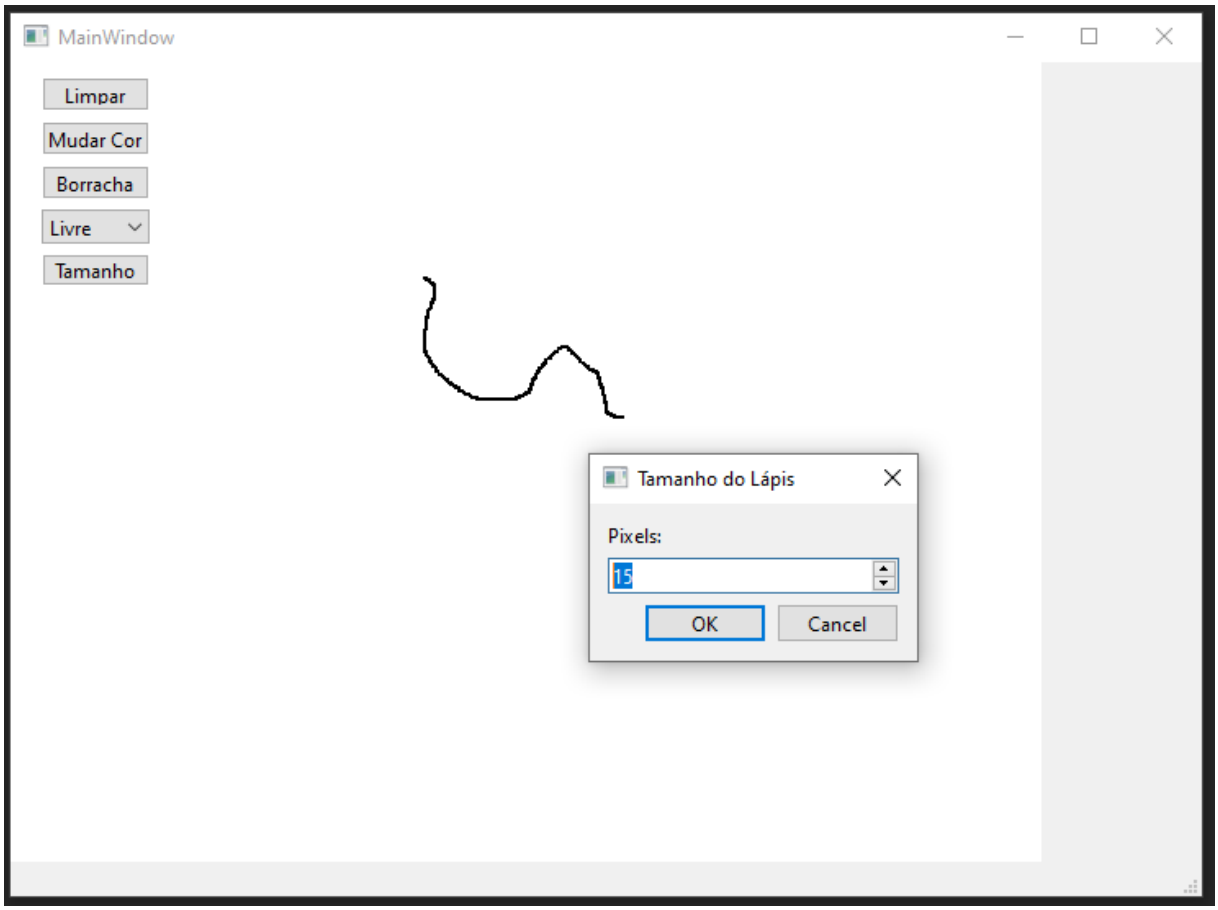

Figura 4 – tamanho do pincel (2025)

A Figura 4 demonstra o sistema de ajuste de tamanho do pincel. A funcionalidade é acionada pelo botão "Tamanho", que abre uma janela de diálogo para a entrada de dados. Nesta janela, o usuário pode definir a espessura do traço em pixels, seja digitando o valor diretamente no campo ou utilizando as setas de controle para ajustá-lo, o que oferece um controle numérico preciso sobre a ferramenta. Os testes validaram que, ao confirmar um valor, a propriedade de tamanho é corretamente atualizada na lógica de desenho, o que se reflete imediatamente na espessura dos novos traços feitos na tela.

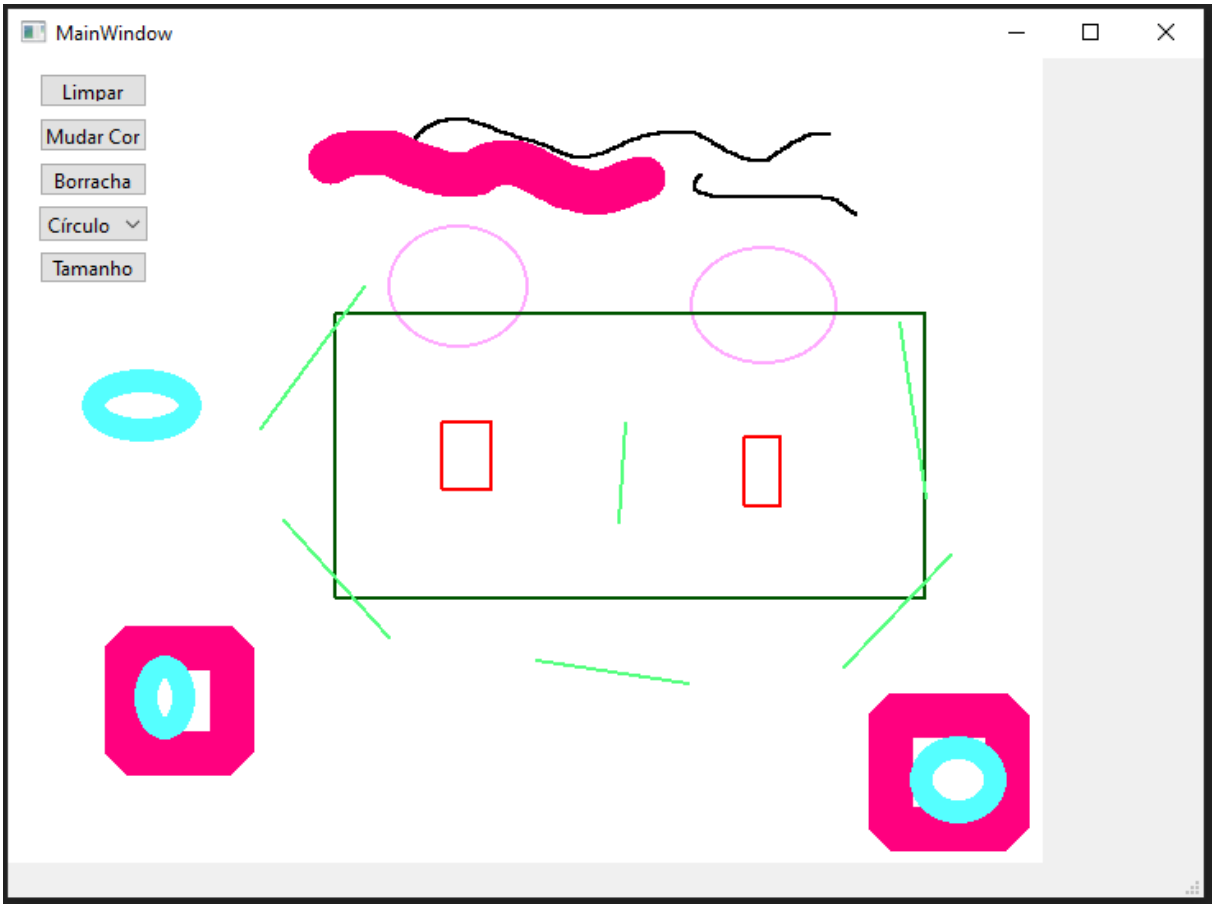

Figura 5 – exemplo de uso (2025)

A Figura 5 apresenta um exemplo de uso combinado das diversas ferramentas disponíveis na aplicação. Na tela, é possível observar traços à mão livre com diferentes cores e espessuras, demonstrando a funcionalidade do pincel, da seleção de cor e do ajuste de tamanho. A imagem também exibe a utilização das ferramentas de formas geométricas, com a presença de retângulos e círculos de atributos distintos. Esta cena valida que o sistema de gerenciamento de ferramentas funciona de maneira coesa, permitindo que o usuário alterne entre os diferentes modos de desenho de forma fluida para criar composições complexas.

Em resumo, os resultados obtidos demonstram que todas as funcionalidades planejadas para a aplicação foram implementadas com sucesso. O fluxo de uso, desde a seleção de ferramentas e atributos até a renderização do desenho na tela, funciona de maneira consistente. A integração de componentes visuais do Qt, enriquece a experiência do usuário, fornecendo feedback claro e aumentando a usabilidade da aplicação.

4.3 Discussão

A análise dos resultados mostra que a interface da aplicação de desenho é integrada e funcional. A mecânica de desenho com o mouse e o sistema de eventos, que são a base da interação, se mostraram precisos e responsivos durante os testes. Sobre essa base sólida, as diversas ferramentas de criação foram bem-sucedidas em oferecer uma experiência de usuário flexível e completa.

As ferramentas de personalização, como a seleção de cores e o ajuste de tamanho do pincel, criaram um ambiente que estimula a criatividade do usuário. A capacidade de alternar entre o desenho à mão livre, a borracha e as formas geométricas se tornou o principal fator de versatilidade da aplicação, permitindo a criação de composições variadas. O engajamento do usuário, nesse caso, vem da combinação de ferramentas que ele pode empregar para alcançar o resultado visual desejado.

A aplicação apresentou a estabilidade e a performance esperadas. A resposta da interface se manteve fluida mesmo ao desenhar rapidamente ou ao preencher a tela com múltiplos objetos, o que valida a eficiência da framework Qt. Esse bom desempenho também é um reflexo positivo da arquitetura orientada a objetos adotada. A organização do código em classes modulares, como a DrawWidget, foi fundamental para gerenciar a complexidade do projeto, permitindo que cada sistema (desenho, estado das ferramentas) fosse desenvolvido e testado de forma independente, o que contribuiu para a estabilidade do produto final.

Em resumo, a discussão dos resultados confirma que a aplicação é um produto completo e funcional, que cumpre o que foi planejado. A metodologia e as ferramentas escolhidas provaram ser adequadas para o escopo do trabalho. Embora os testes tenham validado o software, reconhece-se que avaliações de usabilidade com mais usuários poderiam fornecer dados adicionais para um design de interface ainda mais refinado, um ponto que pode ser explorado em versões futuras.

5 CONCLUSÃO

Com base no desenvolvimento e nos resultados apresentados, conclui-se que o projeto da aplicação de desenho atendeu plenamente aos objetivos estabelecidos. A implementação foi realizada com sucesso, utilizando as ferramentas descritas na metodologia e, principalmente, aplicando de forma prática os conceitos da Programação Orientada a Objetos para estruturar o software.

O desenvolvimento do projeto permitiu um aprofundamento significativo em técnicas de desenvolvimento de interfaces gráficas. A modelagem da lógica de desenho em uma classe encapsulada (DrawWidget), o gerenciamento de eventos do mouse para interação em tempo real e a conexão entre a interface e a lógica de negócio através de sinais e slots foram os principais pontos de aprendizado técnico. Todos os objetivos planejados foram alcançados, e as funcionalidades foram implementadas de forma a garantir a experiência de usuário funcional que foi proposta, como comprovado nas seções de Avaliação e Resultados.

Para trabalhos futuros, o projeto serve como uma base sólida para diversas melhorias. Sugere-se a adição de funcionalidades mais avançadas, como um sistema de "desfazer/refazer" (undo/redo), a capacidade de salvar a imagem em um arquivo e carregar imagens existentes, ou a inclusão de novas ferramentas de desenho, como um balde de preenchimento ou uma ferramenta de texto.

Em resumo, o desenvolvimento desta aplicação consolidou na prática os conceitos de programação e design de interfaces de usuário. O projeto não apenas resultou em um produto de software interativo e funcional, mas também serviu como uma valiosa experiência formativa na área de desenvolvimento de aplicações de desktop com uma arquitetura orientada a objetos.

REFERÊNCIAS

A. ONLINE:

THE QT COMPANY. Qt Framework**.** Disponível em: https://www.qt.io. Acesso em: 1 jul. 2025.

THE QT COMPANY. Qt Creator**.** Disponível em: https://www.qt.io/product/development-tools. Acesso em: 1 jul. 2025.

B. Livro:

DEITEL, H. M.; DEITEL, P. J. C++: como programar. 10. ed. São Paulo: Pearson, 2016.

GLOSSÁRIO

Event Loop: Em aplicações com interface gráfica, é o ciclo principal que mantém o programa rodando. Ele fica constantemente aguardando por eventos do usuário (como cliques de mouse ou teclas pressionadas) e os direciona para as partes corretas do código para serem processados. É o que torna a aplicação interativa.

Framework: Em desenvolvimento de software, é um conjunto de bibliotecas, ferramentas e padrões que fornece uma estrutura base para a construção de aplicações. O Qt é um exemplo de framework para a criação de interfaces gráficas.

GUI (Graphical User Interface): Interface Gráfica do Usuário. Refere-se à forma de interação com um programa que utiliza elementos visuais como janelas, botões e ícones, em oposição a uma interface baseada apenas em texto.

IDE (Integrated Development Environment): Ambiente de Desenvolvimento Integrado. Um software que agrupa diversas ferramentas para programadores, como um editor de código, um compilador e um depurador. O Qt Creator é o IDE que utilizamos.

Widget: Termo usado pelo Qt para se referir a qualquer componente visual da interface, como um botão, uma caixa de texto, um slider ou até mesmo a janela inteira. São os "blocos de construção" de uma GUI.