UNIVERSIDADE ESTADUAL DE FEIRA DE SANTANA
Autorizada pelo Decreto Federal nº 77.496 de 27/04/76
Recredenciamento pelo Decreto n°17.228 de 25/11/2016
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
COORDENAÇÃO DE INICIAÇÃO CIENTÍFICA

XXIII SEMINÁRIO DE INICIAÇÃO CIENTÍFICA DA UEFS
SEMANA NACIONAL DE CIENTÍFICA E TECNOLÓGICA - 2019

INTERFACE GRÁFICA PARA SIMULAÇÃO DE CIRCUITOS

Anésio Sousa dos Santos Neto[1]; Marcos de Araujo Paz[2]; Fernanda Castelo Branco de Santana[3]

1. Bolsista PIBIC/FAPESB, Graduando em Engenharia de Computação, Universidade Estadual de Feira de Santana, e-mail: anesios98@gmail.com
2. Orientador, Departamento de Tecnologia, Universidade Estadual de Feira de Santana, e-mail: marcospaz@ecomp.uefs.br
3. Participante do projeto, Área de informática, Instituto Federal da Bahia, e-mail: fernandacastelo@ifba.edu.br

PALAVRAS-CHAVE: Simulador; Interface gráfica; Circuitos elétricos.

INTRODUÇÃO

Estudos de sistemas elétricos e eletrônicos são realizados com o auxilio de programas de simulação de circuitos. Atualmente existe considerável variedade de simuladores no mercado, porém muitos sofrem de limitações, sejam de usabilidade ou por serem software proprietários - protegidos por direitos autorais, dentre outros. A motivação desse trabalho é o desenvolvimento de um simulador com uma interface robusta e extensível que combinado a modelagem orientada a objetos e a códigos de cálculo já existentes possa ser usado para diferentes tipos de análise, tendo a possibilidade de inclusão de modelos por nós desenvolvidos, visto que é bastante difícil a inclusão de novos modelos em programas pré-existentes.

MATERIAL E MÉTODOS OU METODOLOGIA (ou equivalente)

A linguagem escolhida para o desenvolvimento do simulador foi o C++, utilizando o framework Qt para a criação da interface gráfica de usuário (GUI) e usando o ambiente integrado de desenvolvimento (IDE) Qt Creator para a escrita dos códigos. Com o amadurecimento do projeto, surgiu-se a necessidade do uso de tecnologias robustas voltadas a computação gráfica. Optou-se pelo uso do Graphics View Framework por ser uma rica API já utilizada em sistemas similares e fazer parte do framework Qt, assim simplificando processos.

A linguagem C++

C++ é uma linguagem de programação multiparadigma de nível médio baseada na linguagem de programação C. Foi criada por Bjarne Stroustrup na década de 80 e foi desenvolvida com o objetivo de melhorar uma versão do núcleo Unix (PACIEVITCH). C++ se diferencia de outras linguagens de programação por muitos quesitos. Entre eles estão: portabilidade, multiparadigma, herança múltipla e um suporte incrível a bibliotecas (LIPPMAN; LAJOIE; MOO, 2013).

O framework Qt

Qt foi escolhido por diversos motivos, dentre eles suporte a multiplataformas, ser rica em utilidades prontas, ter extensa variedade de exemplos e ter uma vasta documentação, amplamente necessária no desenvolvimento de projetos assim (SUMMERFIELD; BLANCHETTE, 2006).

Graphics View Framework

O Graphics View Framework (GVF) é um framework gráfico que faz parte do framework do Qt. Foi desenvolvido pela Qt Company Ltd e foi lançado na atualização 4.2 do Qt em 4 de outubro de 2006. Foi criado para substituir seu predecessor QCanvas e provê widgets (objetos gráficos) para manipulação de uma grande quantidade de itens 2D customizados e para a visualização desses itens disponibilizando utilidades como rotação e zoom tanto nos objetos quanto na visualização dos mesmos (QT COMPANY).

Diagrama de classes

Inicialmente, com base nos requisitos que o simulador deve cumprir foram gerados casos de uso que identificam situações de utilização do usuário em relação ao simulador. Estes casos de uso foram aprimorados de uma forma sistêmica, indicando situações necessárias e situações posteriores a cada um desses eventos. Estes aprimoramentos elevaram o conhecimento a cerca dos requisitos da interface e levaram a tona os objetos constituintes da mesma. Com esses objetos foi dado inicio a criação do *diagrama de classes* (Figura 1), utilizando notação em Linguagem de Modelagem Unificada (UML).

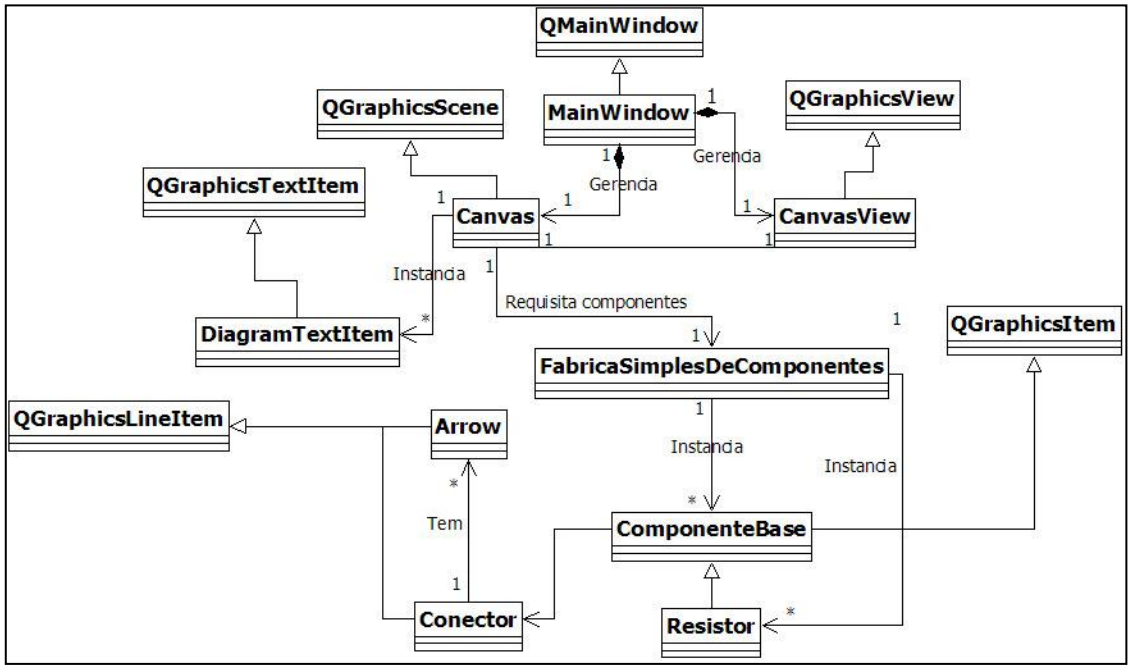

Figura1: Diagrama de classes modelado para o desenvolvimento do simulador.

Classes do sistema

Os códigos do simulador são formados por classes que se misturam entre classes nativas do framework Qt e classes desenvolvidas no projeto que são subclasses dessas classes nativas. A seguir são brevemente apresentadas as classes que constituem o simulador.
MainWindow - É uma subclasse de *QMainWindow* e é responsável por gerenciar todo o fluxo do simulador. Age como um contêiner de botões, barras de tarefas, menus e da área de criação (Canvas) e visualização (CanvasView) dos circuitos. **Canvas** - É uma subclasse de QGraphicsScene e é responsável por gerenciar os componentes gráficos do simulador. Age como um contêiner de componente e propaga eventos entre os mesmos. **CanvasView** - É uma subclasse de QGraphicsView e é responsável por exibir o conteúdo do Canvas, realizar funções como zoom e movimentação e propagar eventos de mouse e teclado para o Canvas. **DiagramTextItem** - É uma subclasse de QGraphicsTextItem e é responsável pela função de inserção de textos no Canvas. **ComponenteBase** - É uma subclasse de QGraphicsItem e é responsável por servir de base para a criação dos componentes inseríveis no Canvas. **Resistor** - É uma subclasse de ComponenteBase e é responsável por ser o representante computacional do dispositivo elétrico resistor. **FabricaSimplesDeComponentes** - Essa classe é responsável por instanciar componentes e repassá-los para o Canvas. Essa abordagem *encapsula* a criação de objetos em um local só, deixando o sistema ter uma manutenção mais fácil. **Arrow** - É uma subclasse de QGraphicsLineItem e é responsável por criar conexões (fios) entre conectores de componentes. **Conector** - É uma subclasse de QGraphicsLineItem e quando associado a um componente o torna disponível para conexões.

RESULTADOS E/OU DISCUSSÃO (ou Análise e discussão dos resultados)
Aplicando as definições propostas da metodologia obteve-se a interface que pode ser vista na figura 2. Esta interface é a ideia final de um conjunto de ideias iniciais de layouts de interfaces que tem como objetivo usabilidade e intuitividade.

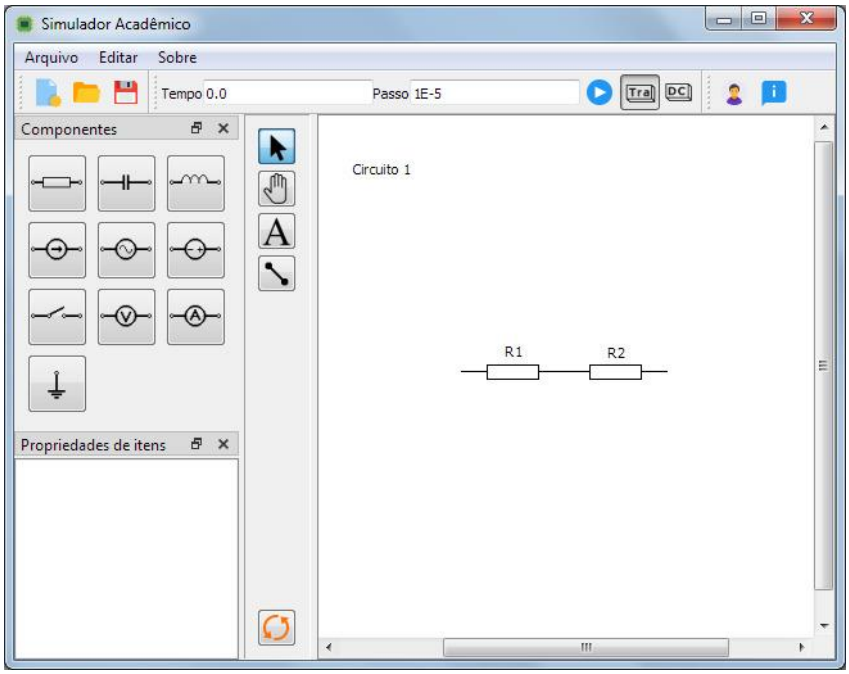

Figura 2: Layout final da interface gráfica do simulador.

O software em sua versão atual já apresenta algumas características das principais encontradas em simuladores similares no mercado. Como já foi apresentado, é possível realizar com usabilidade funções de manipulação e visualização de componentes como zoom in e zoom out, e movimentação no espaço de criação de circuitos sem perda alguma de qualidade. O projeto da interface gráfica é realizado de forma a garantir a robustez do software simulador, proporcionando uma base para implementação de diferentes tipos de análises e simulações. Mantendo sempre a ideia de extensão, a interface foi construída de uma forma que simplifica a adição de novas funcionalidades e também de novos componentes, proporcionando uma "receita" para a criação dos mesmos.

CONSIDERAÇÕES FINAIS (ou Conclusão)

Em conclusão, mesmo entre os desafios de criar uma boa modelagem orientada a objetos e os primeiros contatos com frameworks extremamente ricos e robustos, foi desenvolvido um sistema capaz de suprir de forma sucinta requisitos importantíssimos no que diz respeito ao uso de softwares simuladores, tomando a preocupação de que as funções mantivessem o quesito usabilidade e que a interface continuasse com o status extensível.

REFERÊNCIAS

SUMMERFIELD, M.; BLANCHETTE, J. C++ GUI Programming with Qt 4. 1 ed. Prentice Hall, 2006.

PACIEVITCH, Y. C++. Disponível em <https://www.infoescola.com/informatica/cpp/>. Acesso em 29 Jul. 2019.

LIPPMAN, S. B.; LAJOIE J.; MOO B. E. C++ Primer. 5 ed. New Jersey: Addison-Wesley Professional, 2012.

QT COMPANY. Graphics View Framework. 2006. Disponível em: <https://doc.qt.io/qt-5/graphicsview.html>. Acesso em: 29 Jul.2019