Relatório técnico do trabalho

Version 1.0

Gustavo Lopes Rodrigues, Lucas Santiago de Oliveira

25 de novembro de 2021

Conteúdo

1 Introdução

Esse programa feito para a disciplina de Computação gráfica, tem como objetivo integrar C++, Qt e OpenGL, para fazer uma aplicação que seja capaz de abrir arquivos .off (Object File Format), e ter a capacidade de manipulá-los. O programa em grande parte consiste na implementação do artigo *"Interactive Graphics Applications with OpenGL Shading Language and Qt"*, com apenas algumas melhoras a sua interface, e a adição a uma funcionalidade de melhoria, que é a leitura de malhas mistas.

2 Organização do código

Como dito anteriormente, utilizamos C++ como a linguagem de programação, o Qt como o framework para construir a interface visual na qual o usuário interage com, e o OpenGL, é a API gráfica que renderiza os objetos, processa shaders e texturas. As seções a seguir seram separadas da seguinte forma: primeiro apresentaremos a comunicação entre a interface visual do Qt e o código, e em seguida iremos apresentar o código em si e sua estruturação

2.1 Qt

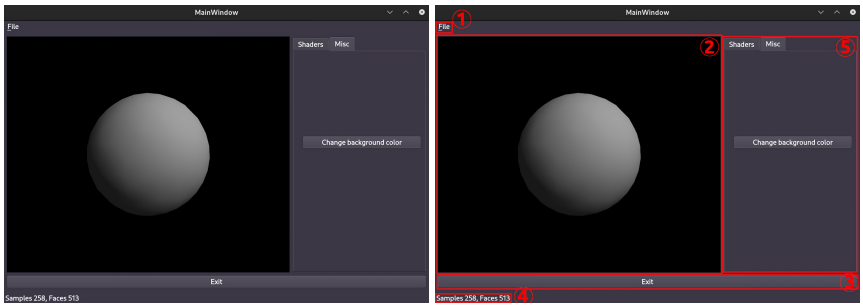

Figura 1: Principal interface do programa, e a direita temos a mesma interface porém com marcações para sinalizar os principais componentes

A área de interação do Qt é composta de Widgets, desde os botões, até o canvas do OpenGL, todos os elementos são Widgets que podem ser dimensionados na tela e mais importante, podem usar sinais para comunicar entre-si.

2.1.1 File

Primeiro temos um QMenuBar, responsável em expandir e mostrar as duas opções presentes:

- **Open** - Um QMenu com um QAction, um sinal conectado com o visualizador dos objetos(GLWidget). Quando acionado(triggered), realiza a ação de abrir o gerenciador de arquivos e abrir um arquivo .off.
- **Screenshot** - Também um QMenu com um QAction, que conecta ao GLWidget. Quando acionado(triggered), realiza a ação de abrir o gerenciador de arquivos e permitir o usuário salvar o frame atual do GLWidget como uma imagem .png.

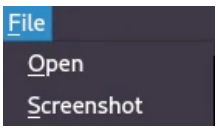

Figura 2: O botão **File** quando expandido

Além disso, o QMenuBar possui **Keybinds**, atalhos com o teclado do usuário, permitindo com que todas as iterações com o QMenuBar seja mais rápida, eis a seguir os keybinds implementados:

- **Alt + f** $\rightarrow$ clicar no botão *File*
  - **Alt + f + o** $\rightarrow$ abrir um arquivo
  - **Alt + f + s** $\rightarrow$ salvar uma captura de tela

2.1.2 Visualizador de Objetos 3D

O principal componente da aplicação, a janela onde os objetos 3D são exibidos é um GLWidget, onde a grande parte da implementação da API do OpenGL.

Esta janela é o componente que também possui a grande parte do código implementado, e isso inclui todo processo dos objetos, texturas e shaders, além de leitura de entradas do teclado e mouse do usuário.

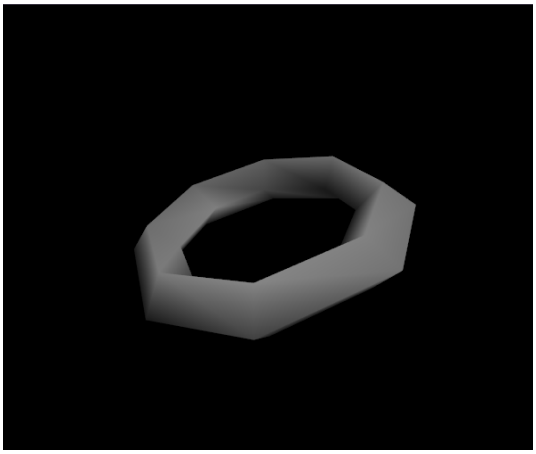

Figura 3: Objecto **octtorus.off** quando carregado

Entraremos em mais detalhe sobre os inputs do teclado e mouse, mas vamos brevemente comentar quais são as possíveis entradas.

Teclado:

- **Número 1 do teclado** $\rightarrow$ aplicar o algoritmo de *Gouraud*
- **Número 2 do teclado** $\rightarrow$ aplicar o algoritmo de *Phong*
- **Número 3 do teclado** $\rightarrow$ aplicar uma *Texture*
- **Número 4 do teclado** $\rightarrow$ aplicar uma textura com *Normal*
- **Esc** $\rightarrow$ fechar a aplicação

Mouse:

- **Scroll** - Realizar ampliação(zoom) no objeto atualmente selecionado.
- **Botão direito + movimentar o mouse** - Realizar uma operação de rotação no objeto.

2.1.3 Botão de saída

QPushButton widget que possui um sinal ligado a janela principal do programa, pedindo o mesmo para o programa ser fechado.

Figura 4: O botão para sair do programa

2.1.4 Barra de status

Uma barra de status que fica no canto inferior da tela, informando usuários de quantos samples e faces foram carregadas do objeto 3D.

Este possui um sinal ligado ao GLWidget, que recebe do mesmo, uma QString, com as informações mencionadas.

Figura 5: Barra de status com informações do **octtorus.off**

2.1.5 Abas de edição

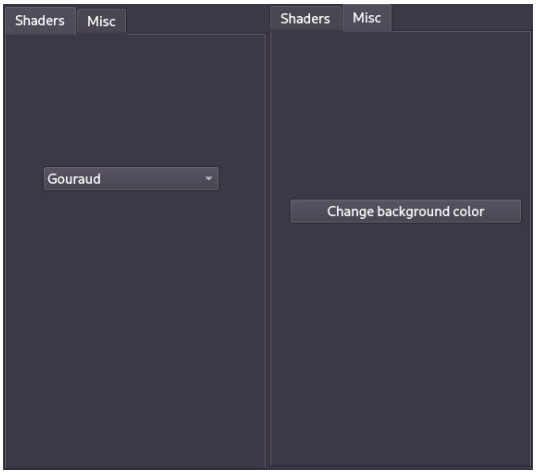

Figura 6: As duas abas de edição

Para armazenar as possíveis, utilizamos um gerenciador de abas(QTabWidget), que armazenará duas abas(QWidget): uma responsável em guardar operações com shaders, e outra para trocar cor do fundo do GLWidget.

- **Shaders** - Possui QComboBox, que apresenta varios textos com nome dos possíveis shaders no programa. Após selecionar um, será acionado um sinal que será enviado para o GLWidget, pedindo para mudar para o shader especificado pelo usuário.
- **Misc** - Possui um QPushButton, que acionará a paleta de cor do Qt, permitindo que o usuário escolha uma cor, para ser a nova cor do fundo.(mais informações na seção do código)

2.2 Código

Agora que a comunicação na interface do Qt foi detalhada, podemos passar a explicar a comunicação dentro do código em si.

3 Detalhando módelos matemáticos