Universidade Federal de Santa Maria
Curso de Ciência da Computação
Disciplina: Computação Gráfica
Primeiro Semestre de 2021
Prof. Cesar Tadeu Pozzer
Data: 28/05/2021

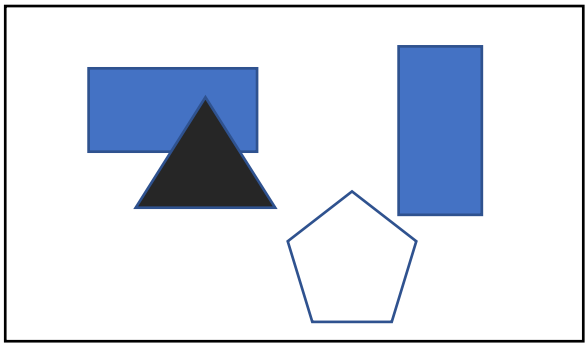
Trabalho 1 – Editor de Figuras Vetoriais

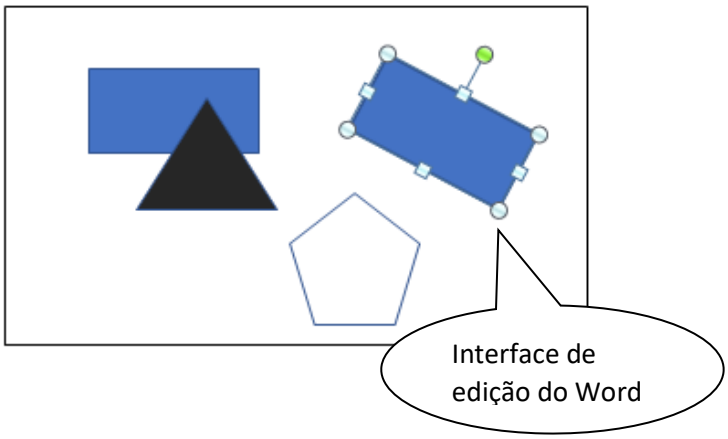
Ferramentas

Linguagem C++, utilizando a API Canvas2D (disponível no site da disciplina) e IDE Code::Blocks, compilando com MinGW (disponível na versão 17.12 da IDE Code::Blocks). **Não podem ser utilizadas bibliotecas auxiliares**. Não pode ser usada a API OpenGL. Desenvolva o trabalho sobre o demo gl_1_CanvasGlut. Antes de enviar, retire todas as funções e arquivos não utilizados. Procure não modificar os arquivos gl_canvas2d.

Descrição

Desenvolva um programa em C++ para fazer manipulações figuras vetoriais, algo semelhante às ferramentas que o Microsoft Word oferece.





O programa deve permitir:

- Inserir/excluir figura.
- Cor da figura.
- Preenchimento da figura (Bônus).
- Editar tamanho da figura.
- Editar orientação da figura (giros de 90 graus).
- Enviar para frente/traz.
- Salvar em arquivo e carregar de arquivo.

O programa deve dispor de botões para realizar as diversas operações. Pode-se (e sugere-se) que seja implementado um "manager" de botões, panels, checkbox, sliders, scrools, etc. Utilize seus conhecimentos de programação. Esse manager (uma classe) poderá ser usado nos demais trabalhos da disciplina.

Quando o programa for inicializado ele deve automaticamente carregar o arquivo “figuras.gr” que deve estar localizado no diretório raiz da aplicação. Se o arquivo não existir, o editor deve inicializar vazio. As figuras devem ser salvas em “figuras.gr”. Já envie um arquivo demo para ser carregado e testado, e que contenha todos os recursos suportados.

Cada aluno deve criar o seu formato de arquivo, que pode ser texto ou binário. Segue uma **sugestão** de formato texto. Sugiro usar um formato binário.

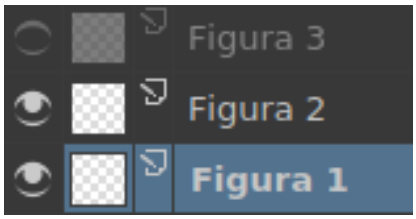
```
2 //indica que existem duas figuras no arquivo
1 10 10 30 30 90 1 0 0 //1 = quadrado, as 4 coordenadas dos cantos, rotação e cor vermelha
2 10 10 66 0 0 1 //2 = círculo, coordenadas do centro, raio e cor azul.
```

O trabalho deve apresentar uma lista de instruções (readme), explicando de forma como o usuário deve interagir com o programa. Enumere no início do código fonte (arquivo main.cpp) os quesitos que foram implementados.

Faça um planejamento de como os elementos serão distribuídos dentro da canvas2D. O programa não precisa ser bonito, mas tem que ser bem organizado e de fácil utilização.

Extras (para notas acima de 9,0):

- (+0,5) Sinalizar qual figura está selecionada.
- (+1,0) Rotacionar figura em qualquer ângulo.
- (+1,5) Concatenar duas figuras (se tornando uma só) = agrupar.
- (+1,0) Permitir inserir polígonos quaisquer.
- (+2,0) Preencher a figura qualquer (circular, polígono, triangulo, etc) com uma imagem (BMP).
- (+1,0) Painel mostrando uma label para cada figura presente na imagem.
- (+1,0) Checkbox permitindo deixar determinada figura invisível sem removê-la.



Objetivos do trabalho

Aprender interação com o mouse, transformações geométricas básicas, manipulação de arquivos (textos/binários), elaboração de formato de arquivo, desenvolvimento de aplicativos gráficos, programação orientada a objetos.

Formato de Entrega:

- Envie pelo Google Classroom.

- O programa deve ser enviado em um arquivo compactado fulano.RAR (fulano = login do aluno). Dentro deste arquivo deve haver um diretório com o mesmo nome do arquivo e, dentro deste diretório, os arquivos do trabalho. Deve-se enviar somente: código fonte (.cpp, .h, .hpp), imagens e o projeto (.cbp). Não devem ser enviadas libs, executáveis, DLLs. Deve-se enviar a Canvas2D.

- Antes do envio, certifique-se de que o projeto contido no seu .rar funciona em qualquer diretório que ele seja colocado e não dependa de outros arquivos não incluídos no envio do trabalho.

Critérios de avaliação:

- Evite códigos replicados. Eles vão descontar muita nota. Eles costumam aparecer em códigos para tratamento de cliques em botões, etc.
- Documentação: descrever no cabeçalho de cada arquivo a ideia geral do código e comentar o que cada método e classe faz.
- Clean code: estrutura do código e nomeação de métodos, classes e variáveis devem ser fáceis de ler e entender. Procurar manter o código o mais simples e organizado possível. Utilizar diferentes arquivos para diferentes classes.
- Pontualidade: Trabalhos não entregues na data não serão avaliados e receberão nota zero.
- Funcionalidade: o programa deve satisfazer todos os requisitos. Programas que não compilarem ou que não atenderem nenhum requisito receberão nota 0 (zero).
- Você pode discutir estratégias e ajudar o colega na implementação, porém evite passar código fonte. Programas semelhantes terão a nota 0 (zero).