

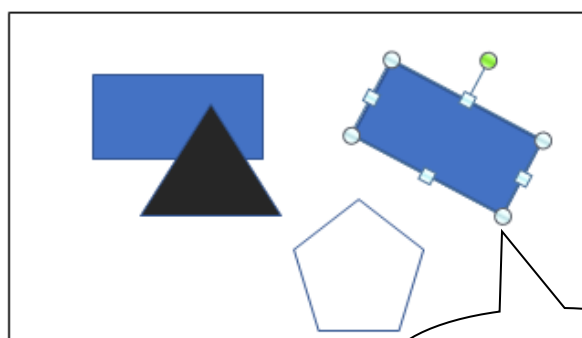
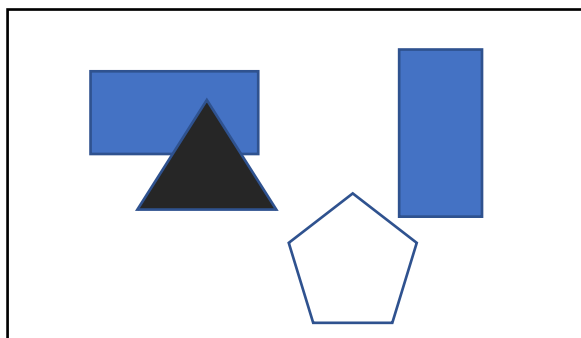
Trabalho 1 – Editor de Figuras Vetoriais

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`.

Descrição

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



Interface de
edição do Word

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 (junto com o projeto code::blocks). 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 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):

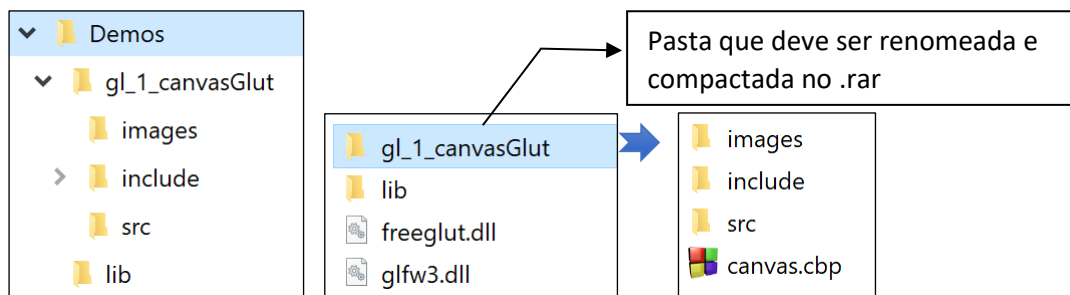
- (+0,5) Sinalizar qual figura está selecionada.
- (+1,0) Rotacionar figura em qualquer ângulo.
- (+1,5) Concatenar duas figuras (se tornando uma só) = agrupar.
- (+1,0) Permitir inserir polígonos quaisquer.
- (+2,0) Preencher a figura **qualquer** (circular, polígono, triângulo, etc) com uma imagem (BMP).
- (+1,0) Painel mostrando uma label para cada figura presente na imagem.
- (+0,5) Aplicar criptografia no arquivo.
- (+0,5) Alterar cor em espaço HSV.
- Mais extras podem ser feitos e serão avaliados.

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:

- O trabalho deve ser entregue pelo Google Classroom.
- Deve-se utilizar como base o projeto gl_1_canvasGlut disponível nos demos da disciplina, como ilustrado na seguinte figura a esquerda.



- A pasta `gl_1_canvasGlut` tem todos os códigos fonte e recursos. A pasta `lib` tem bibliotecas que são comuns a todos os trabalhos, e por isso não deve ser enviada. O mesmo vale para as `dll`.
- A pasta `gl_1_canvasGlut` **deve ser renomeada** com o nome do aluno. Ex: `Trab1Maria`, `Trab2Paulo`, `Trab3Pedro`, `Trab4JoaoPedro`, etc. Esta estrutura vai facilitar a execução dos trabalhos pelo aluno monitor. Todos os arquivos do trabalho devem estar dentro desta pasta, que deve ser a única pasta enviada, compactada em formato `.rar`, cujo nome deve ser o nome do aluno. Ex: `FulanoSobrenome.rar`.
- Dentro desta pasta (`Trab1Maria`, ...), como mostrado na figura a direita, existe:
 - o projeto `code::blocks`,
 - a pasta `src` que contém os códigos fonte (`.c`, `.cpp`, `.h`), e
 - a pasta `imagens` que pode conter arquivos `BMP` (opcional).
- Esta estrutura de pastas não pode ser modificada.
- **Não devem ser enviadas `lib`, `exe`, `obj`, `DLL`, `pdf`, `doc`.**
- Retire todo código não utilizado no trabalho (arquivos, métodos, variáveis, etc), bem como `printf`is de depuração.

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).