Universidade Federal de Santa Maria
Curso de Ciência da Computação
Disciplina: Computação Gráfica
Primeiro Semestre de 2018
Prof. Cesar Tadeu Pozzer
Data: 23/04/2018

Trabalho 2 – Transformações Geométricas e Curvas

Ferramentas

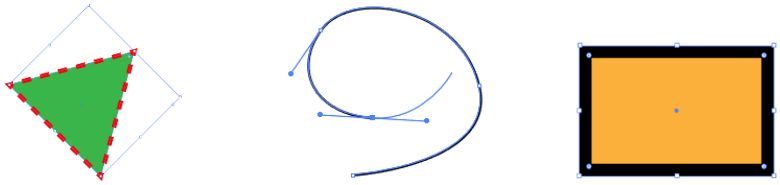
Linguagem C++, utilizando a **API Qt** (versão 5.10.0) e ferramenta **Qt Creator**, utilizando as funções de desenho da Canvas2D (demo gl_4_canvasQT) e compilando com MinGW (disponível na versão 17.12 da IDE Code::Blocks). **Não podem ser utilizadas bibliotecas auxiliares**. Não pode ser usada a API OpenGL.

Descrição

Implemente um ambiente que permita a criação, edição, salvamento e carregamento de formas vetoriais.

O programa deverá apresentar ferramentas que permitam que usuário desenhe curvas a mão livre e primitivas. Também deve ser possível editar com o mouse os pontos de controle (anchors) das formas (shapes) já criadas, bem como excluir e rotacionar as formas.

O usuário deve poder salvar todas as formas que desenhou e depois carregá-las de volta no programa.



Exemplos (do Adobe Illustrator) de uma primitiva rotacionada com sua caixa envolvente, uma ferramenta de desenho a mão livre (curva) e outra primitiva com um estilo de linha e preenchimento diferente.

Requisitos básicos

1. **Ferramentas de desenho (até 3 pts):**
   - 1.1. Primitivas:
     - 1.1.1. Linha;
     - 1.1.2. Quadrado/retângulo;
   - 1.2. Mão livre (pen tool):
     - 1.2.1. Inserir pontos de controle com o mouse;
     - 1.2.2. Desenhar curvas de Bézier, garantindo continuidade C1 entre os patches (em caso de haver mais de um patch);
     - 1.2.3. Patches com 4 pontos de controle;

1.2.4. Utilizar as equações paramétricas vistas em aula;
1.2.5. Permitir curvas abertas ou fechadas (ponto final igual ao inicial);
Dica: procurem observar como funciona em softwares de desenho vetorial (e.g. Adobe Illustrator);

2. **Salvamento e carregamento (até 2 pts):**
   2.1. Salvar todas as formas (shapes) em um arquivo binário;
   2.2. Carregar do arquivo binário e reconstruir as formas (shapes);
3. **Interação (até 3 pts):**
   3.1. Forma ativa: deve ser possível selecionar (definir foco) uma forma;
   3.2. Bounding box: a forma ativa (selecionada) deve apresentar uma caixa envolvente;
   3.3. Operações sobre uma forma ativa:
      3.3.1. Excluir;
      3.3.2. Mover (translação);
      3.3.3. Rotacionar;
      3.3.4. Editar os pontos de controle;
4. **Interface (até 1 pt):**
   4.1. Visualmente bonito e bem organizado: organizar o espaço na tela para dispor os elementos da interface.
   4.2. Didático. Busque utilizar os componentes (botões, barras, sliders, etc) já prontos e disponíveis no Qt para criar interfaces mais práticas e amigáveis.
5. **Critérios gerais:**
   5.1. Atenção às especificações do trabalho;
   5.2. Qualidade do código:
      5.2.1. **Modularização e utilização de classes para modelar as entidades do programa**;
      5.2.2. Documentação do código;
      5.2.3. Legibilidade do código. Atenção à nomenclatura de variáveis e funções;

Extras (para nota acima de 9.0)

1. Transformação de escala (0,5 pt);
2. Preenchimento de formas fechadas (até 1 pt);
3. Preenchimento de formas fechadas usando algoritmo de scanline (até 2 pt);
4. Opção de desenhar as curvas utilizando B-Spline (até 1 pt);
5. Primitivas extras: triângulo, círculo/elipse, hexágono, etc (0,25 cada, até 1 pt);
6. Permitir adicionar/remover pontos de controle em qualquer parte de uma curva já pronta (até 1 pt);
7. Escolher estilo de linha (espessura, tracejada, contínua, etc) (até 1 pt);
8. Alterar a ordem de desenho das formas (trazer para frente / enviar para trás) (até 1 pt);
9. Camadas (layers): agrupar as formas em camadas que influenciam na ordem de desenho e permitir a mudança na ordem das camadas (até 1 pt);
10. Formatação individual de cada figura (com escolha de estilo de linha, cor de linha e cor de preenchimento) (até 0,75 pt);
11. Mais ideias que acrescentem ao conteúdo explorado neste trabalho também poderão ser recompensadas.
12. Curvas com mais de um patch (até 0.5 pt)

Obs.: as primitivas rect e rectFill da Canvas2D só permitem o desenho de figuras alinhadas ao eixo, por isso recomenda-se aprender as possibilidades e limitação das funções polygon e polygonFill, para facilitar o desenho e preenchimento de formas não alinhadas ao eixo ou rotacionadas.

O trabalho deve apresentar uma lista de instruções, 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.

Recomenda-se utilizar o demo gl_4_canvasQT (disponível no site da disciplina) como base para construir o seu projeto sobre ele. Estude esse demo para aprender como funcionam as callbacks e a criação de uma interface no Qt. Caso você decida criar um projeto do zero, lembre-se de enviar o arquivo .ui que for gerado junto com os demais.

Data e Formato de Entrega:

- Data: 12/05/2018, até as 23:59. Após esse prazo o trabalho será desconsiderado.
- No e-mail e no cabeçalho do arquivo, deve conter o nome completo do aluno. O arquivo deve ser enviado para pozzer3@gmail.com e bruno.t.nasc@gmail.com, com o subject "CG T2".
- 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 arquivos de áudio (quando existirem) e o projeto (.pro). Não devem ser enviadas libs, executáveis, DLLs, arquivos .pro.user, .suo e pastas de build.
- Ex.: o arquivo fulano.rar deve conter um único diretório chamado fulano. Dentro desse diretório devem estar todos os arquivos do trabalho, incluindo o README.
- O diretório do projeto com os arquivos a serem entregues deverá ficar semelhante ao seguinte:

```
fulano/
    ├
    ├ CanvasQT.pro
    ├ README.txt
    ├ glCanvas2d.cpp
    ├ glCanvas2d.h
    ├ main.c
    └ demais arquivo .cpp, .h e .hpp do seu projeto
```

Critérios de Avaliação:

- Documentação: descrever no cabeçalho de cada arquivo a ideia geral do código e detalhes específicos de partes que mereçam uma explicação. Não comente por exemplo o que faz b++.
- README: incluir um arquivo "README.txt" contendo informações sobre quais funcionalidades foram implementadas (requisitos e extras).
- Pontualidade: Trabalhos não entregues na data não serão avaliados e receberão nota zero.
- Legibilidade: nome de variáveis, estruturação do código.
- Clareza: facilidade de compreensão – evite códigos complexos e desnecessários. Adote a solução mais simples possível.

- 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).

A**lgumas coisas sobre a entrega desse trabalho para facilitar e evitar problemas na hora da correção.**

- O projeto entregue deve estar em condições de ser compilado no Windows.
- O aluno deve implementar o seu trabalho sobre o demo gl_4_canvasQT. Esse demo já está com o projeto configurado e tem uma interface implementada na qual o aluno pode se basear para editar e fazer a interface do seu programa.
- ATENÇÃO para os caminhos de salvamento e carregamento utilizados. Não utilize caminhos absolutos (/home/fulano/minhaPasta/blablabla/...). Use caminhos relativos e, preferencialmente, mantenha-os apontando para no diretório raiz (e.g. ./meuArquivo.ext, normalmente nem precisa do "./").
- Leiam atentamente toda a especificação do trabalho.