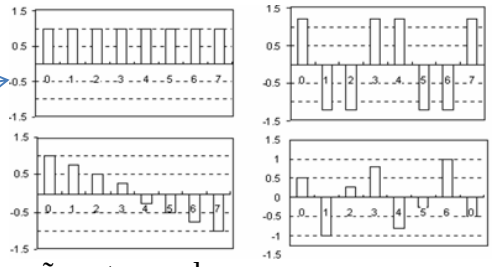
Universidade Federal de Santa Maria
Curso de Ciência da Computação
Disciplina: Computação Gráfica
Primeiro Semestre de 2017
Prof. Cesar Tadeu Pozzer
Data: 24/03/2017

Trabalho 1 - Transformada Discreta de Cosseno

Ferramentas: Linguagem C++, utilizando a API Canvas2D (disponível no site da disciplina - gl_3_canvasGlut) e IDE Code::Blocks, compilando com MinGW (disponível na versão 16.01 da IDE Code::Blocks). **Não podem ser utilizadas bibliotecas auxiliares.** Não pode ser usada a API OpenGL.

Descrição:

Desenvolva um programa que aplique a Transformada Discreta de Cosseno (DCT) e sua inversa (IDCT) em um sinal amostrado unidimensional. O sinal deve ter (no mínimo) 8 amostras (entre -100 e 100). Essas amostras devem ser geradas automaticamente, com opção de atualização e amostras predefinidas (ex: função seno).



O programa deve plotar:

1. As 8 formas de onda ortogonais (continua / amostrada)
2. A função a ser amostrada escalada de forma apropriada
3. A função reconstruída
4. A diferença entre a função original e a reconstruída
5. O valor de cada coeficiente de cada frequência junto a função ortogonal

Interações da aplicação

1. Poder regerar o sinal de entrada (randômico) ou selecionar função predefinida
2. Desabilitar alguma função base no processo de reconstrução do sinal

Requisitos

1. Deve ser de fácil utilização
2. Todo em tempo real: uma seleção implica na exibição do resultado sem precisar clicar em um botão “processar”
3. Visualmente bonito e bem organizado – organizar o espaço na tela para dispor os gráficos, etc.
4. Didático. Criar componentes (botões, barras, etc) e interfaces amigáveis.
5. Os gráficos poderão ser plotados separadamente ou no mesmo gráfico com linhas de cores diferentes. A escala do gráfico deve se auto ajustar em função dos valores das amostras e tamanho da tela.

Extras (Para nota acima de 9.0):

1. (+1) Usar vetor de quantização.
2. (+3) Aplicar algoritmo de compactação
3. (+3) Opção para fazer tudo em 2D (pode ser imagem em tons de cinza). As imagens de entrada e saída devem ser exibidas na interface.
4. Etc.

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.

Data e Formato de Entrega:

- Data: 10/04/2017, até as 23:59. Após esse prazo o trabalho será desconsiderado.
- No email e no cabeçalho do arquivo, devem conter o nome completo. O arquivo deve ser enviado para pozzer3@gmail.com, rtrindade@inf.ufsm.br com o subject "CG T1".
- O programa deve ser enviado em um arquivo compactado fulano.rar (fulano = login ou nome 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, imagens e arquivos de áudio (quando existirem) e o projeto. Não deve ser enviadas libs, executável e DLLs em geral.

Critério 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.txt: 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).

=== Conteúdo do trabalho a ser entregue ===

Como explicado no pdf, o que deve ser enviado é

- código fonte do trabalho e arquivos da pasta include
- arquivo de projeto do codeblocks (.cbp)
- arquivo README.txt com dados do aluno e descrição geral do trabalho

Arquivos .suo, .layout e .depend não são necessários
A árvore de arquivos ficará parecida com essa (substituam meu login pelo de vocês):

rtrindade
```
├── include
│   └── GL
│       ├── freeglut_ext.h
│       ├── freeglut.h
│       ├── freeglut_std.h
│       └── glut.h
├── README.txt
├── rtrindade.cbp
└── src
    ├── gl_canvas2d.cpp
    ├── gl_canvas2d.h
    ├── main_cpp.cpp
    └── ** outros arquivos .cpp e .h **
```

Comprimam dentro de um arquivo login.rar

=== Instruções para desenvolvimento no linux ===
Para o projeto funcionar no codeblocks em alguma distribuição linux variante do Debian ou do Ubuntu, basta mudar as seguintes linhas no arquivo do projeto (final .cbp) (dá pra ser editado com qualquer editor de texto):

versão do site (pra windows):
```
        <Linker>
            <Add library="..\lib\libglu32.a" />
            <Add library="..\lib\libopengl32.a" />
            <Add library="..\lib\freeglut.lib" />
        </Linker>
```

versão pra linux:
```
       <Linker>
            <Add option="-lGL" />
            <Add option="-lGLU" />
            <Add option="-lglut" />
        </Linker>
```

Para que essas options/flags funcionem, os seguintes pacotes precisam ser instalados:
```
sudo apt-get install libgl1-mesa-dev libglu1-mesa-dev freeglut3-dev
```

Para instalação do codeblocks:
```
sudo apt-get install codeblocks xterm
```

Como o arquivo é basicamente um xml, dá pra comentar usando <!-- e --> se preferirem.
Usem isso somente pra desenvolvimento (se preferirem).
Pra versão de entrega, precisa obrigatoriamente ser a versão pra windows.