Universidade Federal de Santa Maria
Curso de Ciência da Computação
Disciplina: Computação Gráfica
Primeiro Semestre de 2019
Prof. Cesar Tadeu Pozzer
Data: 05 / 04 / 2019

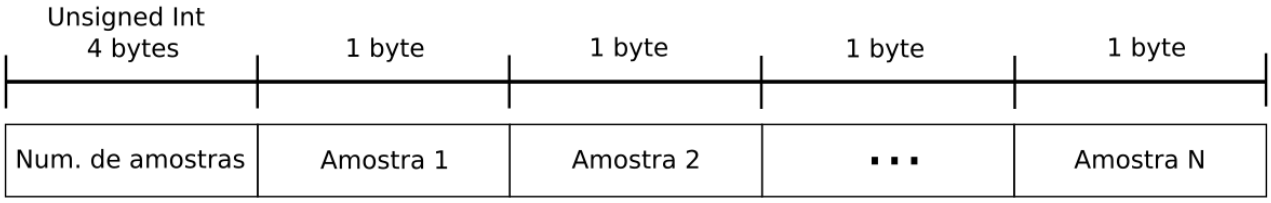
Trabalho 1 - Transformada Discreta de Cosseno

Ferramentas:

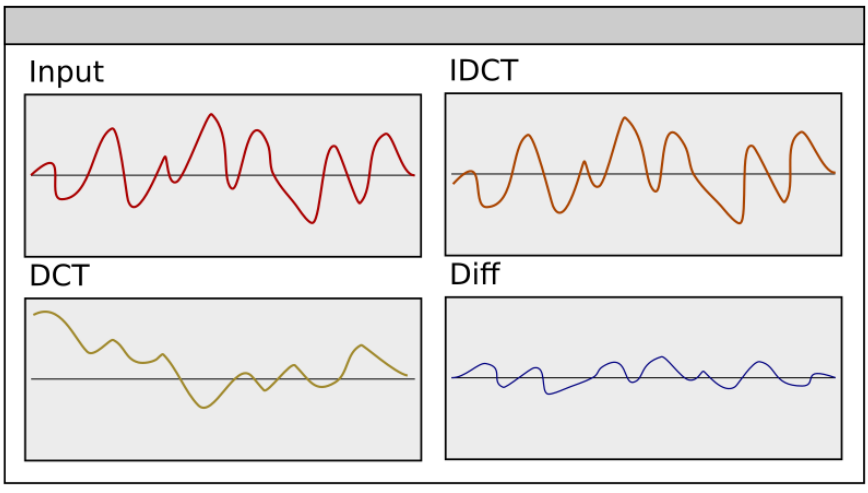
Linguagem C++, utilizando a API Canvas2D (disponível no site da disciplina) e IDE Code::Blocks, compilando com MinGW (disponível na versão 17.12 da IDE Code::Blocks).
Não podem ser utilizadas bibliotecas auxiliares. Não pode ser usada a API OpenGL.

Descrição:

Desenvolva um programa que aplique a Transformada Discreta de Cosseno (DCT) e sua inversa (IDCT) em um sinal amostrado unidimensional. O programa deve ler as amostras de sinal de um arquivo binário "input.dct", onde o cabeçalho é um dado do tipo Unsigned Int (4 bytes) representando a quantidade de amostras a serem lidas, e as amostras são armazenadas sequencialmente, cada uma ocupando 1 byte (cada amostra deverá ter valor entre -127 e 128). O número de amostras deve variar entre 8 e 64. O número de funções base deve ser igual ao número de amostras do arquivo.



O programa deve aplicar a DCT sobre essa entrada, aplicar a IDCT para reconstruir os dados originais, e calcular a diferença entre a entrada original e as amostras reconstruídas. Quatro diferentes arrays de amostras devem ser armazenados separadamente, e plotados na tela em forma de gráficos. Observe que o gráfico da idct pode ter valores grandes, que devem ser escalados.



O tamanho da janela pode variar entre 480x360 e 1600x900. O resultado reconstruído deve ser salvo em um arquivo chamado “output.dct”.

O programa deve estar estruturado em classes. Ex: deve ter uma classe parametrizável para desenhar gráficos. Deve ter uma instância para cada tipo de gráfico (input, idct, dct, diff, etc).

Requisitos do programa:

- Leitura das amostras do arquivo de entrada no formato especificado. Os dados lidos serão do tipo byte, mas devem ser armazenado em memória utilizando um array de Int, variando entre -127 e +128.
- Aplicação da DCT sobre as amostras de entrada.
- Aplicação da transformada inversa (IDCT) para reconstrução dos dados originais.
- Calcular a diferença entre a entrada e os dados reconstruídos.
- Plotagem dos gráficos das quatro diferentes amostragens. O tamanho do gráfico deve se adaptar horizontalmente à quantidade de amostras lidas.
- Salvar o resultado reconstruído em um arquivo “output.dct”, no mesmo formato do arquivo original.

Extras (para nota acima de 8,0):

- (+1) Exibir as N funções bases usadas na DCT.
- (+1) Implementação de uma interface interativa (botões para carregar o input, aplicar DCT, aplicar IDCT, sliders para visualizar melhor as amostras, etc.). Quanto menos cliques melhor.
- (+1) Usar vetor de quantização 1D (e plotar o resultado).
- (+2) Carregar a imagem lena128_24bits.bmp (em anexo). Aplicar a DCT e IDCT e exibir na tela as imagens de entrada e saída. Obs.: para ler o arquivo de entrada deverá ser utilizada, exclusivamente, a classe Bmp disponível no demo gl_14_texture (disponível no site da disciplina).
- (+1) Usar matriz de quantização 2D na imagem, reconstruir a imagem original e exibir o resultado na tela.

- (+1) Aplicar algum método de compressão nos dados quantizados (em 1D ou 2D) e mostrar na tela a taxa de compressão.

- Mais ideias que acrescentem ao conteúdo explorado neste trabalho também poderão ser recompensadas.

Entrega:

Data de entrega:
18/04/2019, até as 23:59. **Após esse prazo o trabalho será desconsiderado.**

Formato de entrega:
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 adrian.k.47@hotmail.com, com o subject "CGT1".

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: códigos fonte (.c e .h), imagens (.bmp), arquivos de entrada e saída (.dct) e arquivo de projeto do Code Blocks (.cbp). Não devem ser enviadas .libs, executáveis e DLLs em geral.

Antes do envio, teste se o projeto contido no seu .rar funciona em qualquer diretório que ele seja colocado.

Critérios de avaliação:

- 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.
- README: incluir um arquivo "README.txt" contendo informações sobre quais funcionalidades foram implementadas (requisitos e extras) e instruções de uso do programa caso o aluno julgue necessário ou caso tenha sido implementado uma funcionalidade extra que exija explicação.
- 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 mandar e-mail para o monitor (adrian.k.47@hotmail.com) se tiver alguma dúvida relacionada a descrição ou implementação do trabalho.

Você pode discutir estratégias e ajudar o colega na implementação, porém evite passar código fonte. Programas semelhantes terão a nota 0 (zero).