Computação Gráfica - Exercícios

Jorge Augusto Salgado Salhani

Setembro, 2023

1 Lista 1 - Conceitos básicos

1.1 Pode-se afirmar que a Computação Gráfica está diretamente relacionada com outras áreas que envolvem a manipulação de imagens no computador. Quais são elas e como se relacionam com a CG?

Diversas áreas atualmente utilizam de ferramentas gráficas para criação ou análise, e com isso, são dependentes da computação gráfica. Podemos citar área de marketing digital, com uso de interfaces como Photoshop, e também engenharia e arquitetura, com AutoCAD. Ambas auxiliam na representação de modelos reais por meio de linhas e figuras geométricas que, por meio de CG, podem ser manipuladas.

Em outro aspecto temos também interfaces de criação e reprodução de jogos e filmes, os quais são fortemente dependentes da manipulação e processamento de imagens ao longo do tempo, assim como a forma como objetos de diferentes materiais interagem com a luz em relação a um observador. Ferramentas como Unity e OpenGL são alguns exemplos que auxiliam estes processo.

1.2 Quais as etapas básicas de um sistema típico de Reconhecimento de Padrões em imagens? Descreva brevemente cada uma dessas etapas.

A área de Reconhecimento de Padrões, em resumo, destina-se a obter informações ou categorizar imagens obtidas do mundo real por ferramentas digitais utilizando análise estatística a partir de uma base de dados pré-estabelecida. Um sistema típico de Reconhecimento de Padrões em imagens contém 6 elementos principais. Em ordem são

1. **Aquisição**: como e a partir de quais equipamentos serão obtidas as imagens do mundo real para meios digitais: câmeras

2. **Pré-processamento**: operações realizadas sobre a imagem original para que análises seguintes sejam mais acuradas, sem alteração substancial: aumento de contraste, transformação em escala de cinza

3. **Processamento**: operações realizadas sobre a imagem pré-processada para melhor análise, podendo agora alterar substancialmente a imagem: redução de ruído, reconhecimento de bordas

4. **Análise**: reconhecimento de padrão em si, com análise que depende do contexto do problema. Com as imagens processadas, características importantes podem ser reconhecidas e estruturadas: procurar bifurcações e terminações (em reconhecimento de digitais), procurar métricas de distancia dos olhos, tamanho de nariz (em reconhecimento facial)

5. **Extração de características**: Construção de modelos matemáticos que utilizam dos dados estruturados na etapa de análise que determinam especificamente o objeto: triangulação de terminações (reconhecimento de digitais) e de posições relativas entre os olhos, nariz e boca (reconhecimento facial)

6. **IA / Reconhecimento de Padrões**: Calculo estatístico e comparação com base de dados previamente construído (base de conhecimento) para que seja possível categorizar o objeto obtido na primeira etapa de forma acurada: identificação final da pessoa com certa margem de erro (desejadamente baixa)

1.3 Qual a principal diferença de escopo entre a Visualização Científica e a Visualização de Informação? Dê exemplos de aplicações em cada um deles.

Visualização Científica é uma área destinada à representação gráfica de dados de modo a torná-los mais intuitivos em nível de compreensão e interação, para que novos conceitos possam ser estudados e avaliados. Ou seja, a interpretação deve ser mais intuitiva que outras representações dos dados em análise. São exemplos modelos aerodinâmicos de veículos, construção de modelos 3D do corpo humano, atividade cerebral, enovelamento de proteínas.

Por outro lado, Visualização de Informação destina-se à representação gráfica de dados que buscam explicitar algum fenômeno específico dentro dos possíveis resultados que o conjunto de dados pode conter. Ou seja, a interpretação é complexa e requer avaliação especializada. São exemplos histogramas, gráficos e mapas, onde cores, tamanho e formatos são relevantes para sua compreensão.

1.4 Na década de 80 surgiram os primeiros pacotes gráficos, ou APIs. Atualmente temos diversas APIs gráficas, dentre elas a OpenGL. Qual a principal função da APIs gráficas?

API é um acrônimo para Interface de Programação de Aplicações (*Application Programming Interface*), e sua grande vantagem de uso acontece devido à portabilidade de execução de programas para hardwares diferentes.

Hardwares distintos operam de modos diferentes em relação ao processamento gráfico, e APIs tal como a OpenGL, apresentam comunicação padrão com cada um deles. Assim, é possível construir aplicações via OpenGL e esta, por sua vez, será responsável por inserir as instruções conforme o hardware sob o qual a aplicação está sendo executada.

1.5 Faça uma tabela comparativa enumerando as vantagens e desvantagens da representação de imagens utilizando as tecnologias matricial vs vetorial.

Caso	Matricial	Vetorial
Representação	Pixels	Caminhos
Atualização de frame	Independe da complexidade da imagem	Depende da complexidade da imagem
Conversão em scan	Pontos e polígonos devem ser convertidos em pixels	Não é necessário conversão
Memória utilizada	Imagens de alta qualidade demandam alta memória	baixa memória necessária
Ajuste de tamanho	Rescala da imagem pode distorcê-la	Sem distorções

1.6 Considere os seguintes monitores matriciais com resoluções de 640x480, 1280x1024 e 2560x2048. Qual o tamanho do *framebuffer* (em bytes) necessário para cada um desses sistemas, se cada pixel tem 12 bits de profundidade? E se cada pixel tiver 24 bits de profundidade?

O framebuffer é a memória na qual mantemos a imagem e os metadados da imagem. Para cada pixel, uma série de metadados é armazenado.

Dessa forma, temos que

Resolução	Profundidade	Memória
640x480	12	640x480x12 = 3686400 bits
640x480	24	640x480x24 = 7372800 bits
1280x1024	12	1280x1024x12 = 15728640 bits
1280x1024	24	1280x1024x24 = 31457280 bits
2560x2048	12	2560x2048x12 = 62914560 bits
2560x2048	24	2560x2048x24 = 125829120 bits

1.7 Suponha que um sistema matricial RGB foi projetado para ter uma tela de 8x10 polegadas, com resolução de 100 pixels por polegada em cada direção. Se desejamos armazenar 6 bits por pixel no framebuffer, quanta memória (em bytes) será necessária?

Sendo 100 pixels por polegada, temos 800x1000 pixels. Se desejamos 6 bits por pixel no framebuffer, nossa memória deverá ter 800x1000x6 = 4800000 bits, ou 600000 bytes

1.8 Quanto tempo seria necessário para carregar um framebuffer de 640x480 pixels com 12 bits por pixel, se a memória utilizada permite transferir 105 bits/s? Quanto tempo seria gasto para carregar um framebuffer de 24 bits por pixel e resolução de 1280x1024 pixels a esta mesma taxa de transferência?

Sendo a memória total armazenada no framebuffer igual a 640x480x12 = 3738249 bits, então para uma taxa de transferência de 105 bits/s, serão necessários 35602.28 segundos.

1.9 As três principais etapas conceituais do pipeline de rendering utilizado pelo OpenGL consistem em: Transformações sobre vértices, Projeção e Rasterização. Sumarize o que acontece em cada uma delas.

O pipeline de rendering do OpenGL consiste em uma sequência de passos entre um modelo e sua representação em uma tela 2D. Os três passos são

- **Transformações sobre vértices**: Neste momento, todos os vértices que compõe o objeto a ser representado em tela são definidos em relação ao próprio objeto (OCS - Object Coordinate System) para, em seguida, serem representados em relação à câmera (CCS - Camera Coordinate System).

  Como em relação à câmera podemos já definir quais são as regiões do objeto que serão visíveis, este processamento (culling) também ocorre neste momento. No culling, temos também o view frustum culling (remoção de objetos fora do campo de visão da câmera), o clipping (recorte de objetos fora do campo de visão da câmera) e o back-face culling (remoção de primitivas não visíveis à câmera)

- **Projeção**: Nesta etapa, as primitivas e seus vértices são projetados (paralela ou perspectivamente) sobre o plano cuja normal coincide com o eixo de visão da câmera. Nesse sentido é possível representar todo o conteúdo do objeto (potencialmente em 3D) em um display 2D que será apresentado ao usuário

- **Rasterização**: Por fim, as primitivas que permanecem em tela são representadas como um conjunto de fragmentos que podem ser computadas como pixels (em relação à sua posição 2D e ao seu conteúdo, dado

pela profundidade de cada pixel) e armazenadas no framebuffer, que será utilizado para atualizar a matriz de pixels do monitor

1.10 Qual a principal primitiva gráfica utilizada no pipeline do OpenGL? Porque?

A principal primitiva é o triângulo, pois representa a menor estrutura 2D planar, e portanto, de mais fácil representação e manipulação

1.11 Sugira uma estrutura de dados para representar uma malha de triângulos

Uma estrutura de dados possível seria por meio de listas encadeadas de alguma primitiva (vértices ou faces, e.g) e suas respectivas relações com seus vizinhos próximos (caso vértices, quais três outros que formam uma primitiva triangular, juntamente com o vetor normal ao seu plano, e.g.).

Comumente é utilizada uma lista de faces indexada a diversos elementos topológicos (faces, arestas e vértices). Podemos citar a estrutura Winged Edge.

1.12 O que é um shader? E um vertex shader?

Shader consiste em um processo de determinação, via coordenadas de e relações entre os elementos topológicos de um objeto, de seus atributos de luz, cor, opacidade, textura, entre outras. Um vertex shader define estas mesmas operações exclusivamente aplicadas sobre os elementos vértices do objeto

1.13 Dê exemplos de outras APIs gráficas (que não a OpenGL) e comente em que contexto são usadas.

DirectX, WebGL, Vulkan, Mantle, ...