Computação Gráfica e Interfaces

2016-2017

Fernando Birra

WebGL

2016-2017

Fernando Birra

Objetivos

- Perspetiva histórica dos sistemas gráficos (APIs)
- Diferenças entre modo direto e com retenção
- OpenGL como máquina de estados
- Organização da API (tipos, funções)
- Análise de programas de exemplo

Evolução das APIs Gráficas

- 1973: IFIPS formou dois comités para se criar um standard para aplicações gráficas
  - Graphical Kernel System (GKS)
    - Core
- GKS veio a ser adotado como standard ISO e mais tarde (anos 80) como standard ANSI
- GKS não era de fácil extensão para 3D (GKS-3D)
  - Demasiado atrasado face à evolução do hardware

Evolução das APIs Gráficas

- Programmers Hierarchical Graphics System (PHIGS)
  - Surgiu na comunidade CAD (Computer Aided Design)
  - Modelo baseado numa base de dados gerida pelo sistema (retained mode)

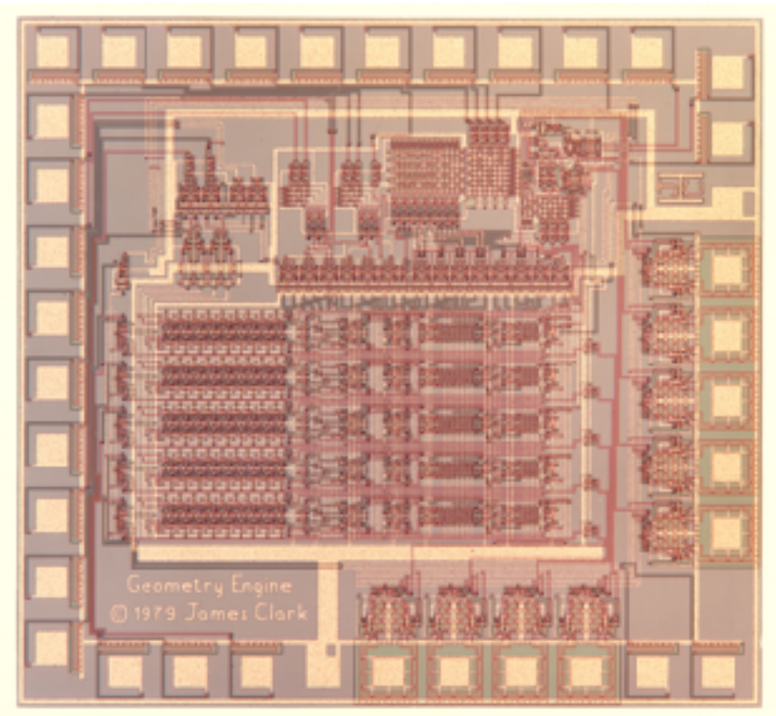
- X Window System
  - Desenvolvido pela DEC/MIT
  - Arquitetura cliente/servidor para gráficos
    - aplicação corre numa workstation remota (cliente) que envia os pedidos para o terminal gráfico local do utilizador (servidor de gráficos)

Evolução das APIs Gráficas

- A empresa Silicon Graphics Inc (SGI) revolucionou as workstations gráficas ao desenvolver a primeira placa gráfica que implementava o pipeline gráfico completamente em hardware (1982)

- A acompanhar o hardware revolucionário foi introduzida uma biblioteca para a sua programação, denominada GL

- GL permitia o desenvolvimento de aplicações 3D interativas de forma relativamente simples



Evolução das APIs Gráficas

- Em 1992 surge o OpenGL, uma API inspirada no sucesso do GL e independente da plataforma hardware:
  - De fácil uso
  - Suficientemente de baixo nível para estar perto do hardware
  - Omitia a gestão de janelas e do input para ser completamente independente da plataforma

Evolução do OpenGL

- Inicialmente controlado por um organismo autónomo - Architecture Review Board (ARB)
- Entre os membros do ARB incluíam-se:
  - SGI, Microsoft, HP, 3DLabs, IBM, Nvidia,…
- Atualmente controlado pelo designado Kronos Group
- Até à versão 2.5 era relativamente estável:
  - mantinha a retro-compatibilidade
  - ia incluindo de forma evolutiva os avanços do hardware (texturas 3D, vertex e fragment shaders)
- Possui um mecanismo de extensões para features específicas e eventualmente "*vendor specific*".

OpenGL Moderno

- A performance foi aumentada pelo uso do GPU em detrimento do CPU
- Os GPUs são controlados (programados) através de programas especiais denominados por *shaders*
- O trabalho das aplicações é o de enviar para o GPU os dados…
- …e pedir ao GPU para efetuar/sintetizar a imagem

Desenho em Modo Imediato

- Os dados relativos aos objetos vão sendo enviados para o GPU à medida que vão sendo produzidos no CPU

- Criam um *bottleneck* entre o CPU e o GPU pois a largura de banda é subaproveitada

- Este modo foi removido por completo nas versões 3.1 de OpenGL e na versão 2.0 do OpenGL ES

- Cada vértice era assinalado por uma chamada `glVertex`

Desenho em Modo com Retenção

- Os dados relativos aos vértices são colocados num array

- Os dados são enviados para o GPU duma só vez, mas…

- em vez de serem imediatamente desenhados e descartados pelo GPU…

- … ficam armazenados no GPU e poderão ser redesenhados com um mero pedido e praticamente sem consumo de bandwidth CPU-GPU

OpenGL 3.1

- Totalmente baseado em *shaders*
  - Sem *shaders* por omissão
  - Cada aplicação tem que possuir, no mínimo, um *shader* de vértices e outro de fragmentos
- Sem modo imediato
- Menos variáveis de estado
- A maior parte das funções introduzidas pelo OpenGL 2.5 foram removidas
- Não é obrigatória a retro-compatibilidade
  - Em algumas implementações é possível usar uma extensão para suportar código antigo.

OpenGL (família)

- OpenGL ES
  - Versão minimalista/simplificada e destinada a dispositivos *embedded*.
  - Version 1.0 é uma simplificação de OpenGL 2.1
  - Version 2.0 é uma simplificação de OpenGL 3.1
    - baseado em *shaders*
- WebGL
  - Implementação Javascript de OpenGL ES
  - suportado nos *browsers* modernos
- OpenGL 4.1, 4.2, …
  - Inclui *features* mais avançadas: shaders de novos tipos *tesselation*, *geometry* e *compute*