Hardware pro počítačovou grafiku NPGR019

3D akcelerátory - historie a architektura

Josef Pelikán
Jan Horáček

http://cgg.mff.cuni.cz/
MFF UK Praha

2012

Pokroky v hardware

- **3D akcelerace** běžná i v konzumním sektoru
- zaměření na **hry, multimédia**
- **vzhled** - kvalita prezentace
  - velmi důmyslné techniky texturování a stínování
  - kombinace mnoha textur, modularita zpracování
- vysoký **výkon**
  - nejmodernější čipové technologie pro výrobu GPU,**masivní paralelismus**
  - velmi rychlé **paměti** (GDDR5)
  - výjimečné sběrnice mezi GPU a CPU - **AGP, PCI-E**

Pokroky v software

- dvě hlavní **knihovny** pro 3D grafiku
  - **OpenGL** (SGI, open standard, nyní Khronos) a **Direct3D** (Microsoft)
  - přístup je podobný, API velmi ovlivněno hardwarem
- nastavení parametrů a **úsporný přenos dat** do GPU
  - maximální sdílení společných datových polí
- **programování grafického řetězce**
  - revoluce v programování 3D grafiky
  - **vertex-shader**: zpracování vrcholů sítě
  - **tesselation**: rozdělování plátů na menší trojůhelníky, deformace podle textury, atp.
  - **geometry-shader**: generování elementů
  - **fragment-shader** (pixel-shader): zpracování jednotlivých pixelů před vykreslením

Vývojové nástroje

- příjemné pro programátory i umělce
- **vyšší jazyky** pro programování GPU
  - Cg (nVidia), HLSL (DirectX), GLSL (OpenGL)
  - velmi podobné
- **kompozice grafických efektů**
  - kompaktní popis celého efektu (GPU programy, odkazy na data) v jednom souboru
  - DirectX **.FX** formát, nVidia **CgFX** formát
  - nástroje: Effect Browser (Microsoft), **FX Composer** (nVidia), **RenderMonkey** (ATI)

Historie I: Silicon Graphics

- první masově rozšířené grafické pracovní stanice
  - hardwarově akcelerovaný grafický subsystém
  - SW nástroje a aplikační programy, "grafika == SGI"
- vybrané stanice (`http://www.g-lenerz.de/`)
  - 1983: **Iris 1000** - grafický terminál k VAXu (Motorola 68000 @ 8 MHz, Geometry Engine, . . .)
  - 1984: **Iris 2000** (grafická stanice, Clark GE), Iris 3000
  - 1986: **Profesional Iris 4D** (první s MIPS procesory)
  - 1988: Power series, **GTX**, **Personal Iris 4D**
  - 1991: **Iris Indigo** (asi nejpopulárnější stanice SGI)
  - 1992: **Iris Crimson**

Historie II: Silicon Graphics, SGI

- další stanice a grafické systémy:
  - 1992: **Indigo R4000** (64bit), **Reality Engine**
  - 1993: **Onyx** (server s RE2)
  - 1993: **Iris Indigo**^{2} (Extreme graphics)
  - 1996: **O2** (levná stanice), **InfiniteReality** engine
  - 1997: **Octane** (2 procesory)
  - 2000: **Octane2** (Vpro gr.)
  - 2002: **Fuel**
  - 2003: **Tezro** (poslední MIPS, SGI podporuje IRIX do 2013)
  - 2008: **Virtu** (x86-based, nVidia)

Historie III: Komoditní produkty

- **první grafický akcelerátor** do domácího počítače
  - 1996: **3Dfx Voodoo**
  - doplňuje grafickou kartu ("pass-through")
  - bilineární filtrování texelů
  - API **Glide**
- první karta se **SLI**
  - 1998: **3Dfx Voodoo^2**
  - **Scan Line Interleave** - dvě karty spolupracují na vykreslení obrazovky
  - jedna kreslí sudé řádky, druhá liché
  - nutnost aby obě karty měly stejnou informaci o scéně (tzn. velikost scény limitovaná velikostí paměti jedné karty)
  - později celou firmu koupila nVidia včetně této technologie

Historie IV: Komoditní produkty

- první karty od **nVidia**
  - 1997: nVidia **Riva 128**
  - 1998: nVidia **Riva TNT** ("TwiNTexel")
- první **HW T&L** ("Transform and Lighting")
  - 1999: nVidia **GeForce 256**
- rok **2000**:
  - nVidia GeForce 2
  - ATI Radeon
- rok **2001**: **programování GPU**
  - **DirectX 8.0** (vertex shaders, fragment shaders, 1.0, 1.1)
  - nVidia GeForce 3 Titanium
  - DirectX 8.1 (PS 1.2, 1.3, 1.4)
  - ATI Radeon 8500 (TruForm)

Historie V: Komoditní produkty

- rok **2002**: **rozšířené programování GPU**
  - **DirectX 9.0** (VS, PS 2.0)
  - nVidia GeForce 4 Titanium (DX 8)
  - ATI Radeon **9700** [Pro]
- rok **2003**
  - DirectX 9.0 kompatibilní karty jsou již levné (VS, PS 2.0)
  - nVidia **GeForce FX**
  - ATI Radeon **9800**
- rok **2004**: **rozšířené programování GPU**
  - **DirectX 9.0c** (VS, PS 3.0)
  - **OpenGL 2.0**
  - nVidia **GeForce 6800**
  - ATI Radeon **X800**

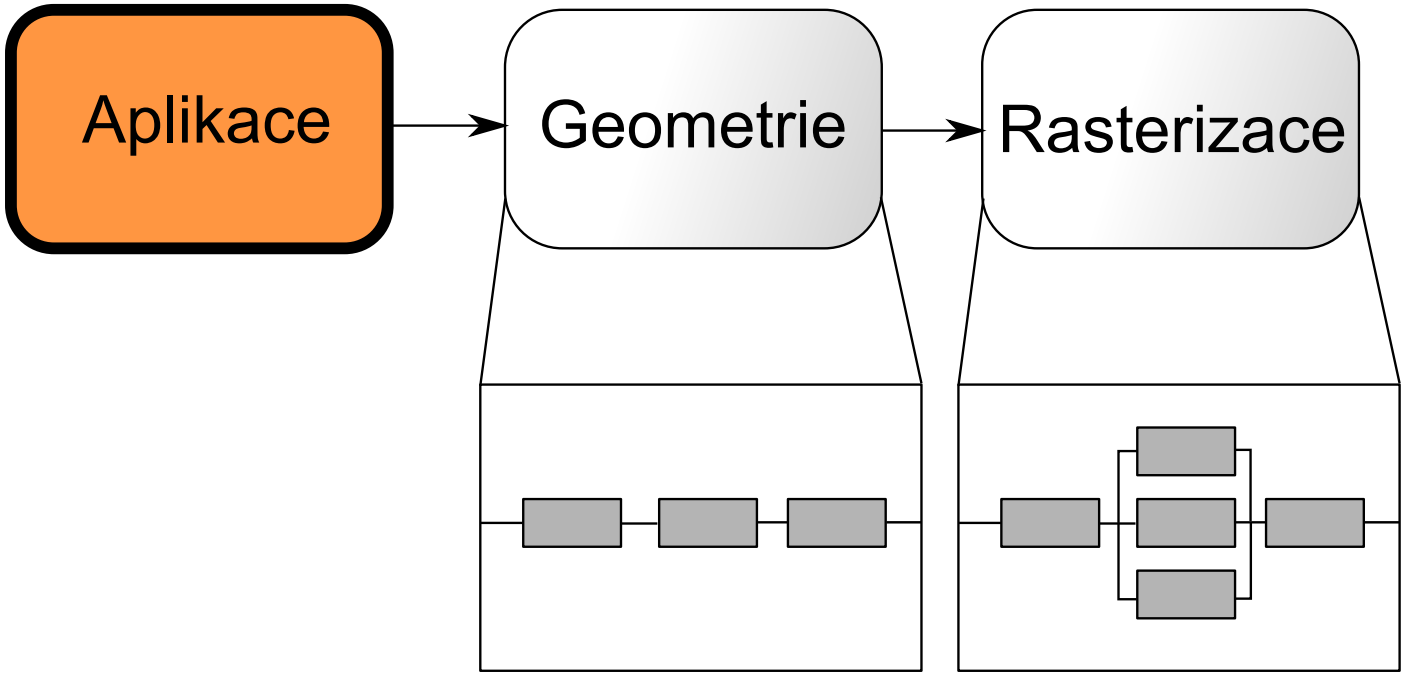
Historie VI: Komoditní produkty

- rok **2005**:
  - **PCI-Express** sběrnice
  - nVidia uvádí obnovenou **SLI**, ATI podobné **CrossFire**
  - nVidia GeForce **7800**
  - ATI Radeon **X550, X850**
- rok **2006**:
  - **OpenGL 2.1**
  - **DirectX 10** (Widows Vista) - **geometry shaders**
  - nVidia GeForce **7600, 7900**
  - ATI Radeon **X1800, X1900**
- rok **2007**
  - **CUDA** - nVidia zpřístupňuje GPU pro obecné úlohy
  - nVidia GeForce **8800**
  - ATI Radeon **HD 2400, 3850**

Historie VII: Komoditní produkty

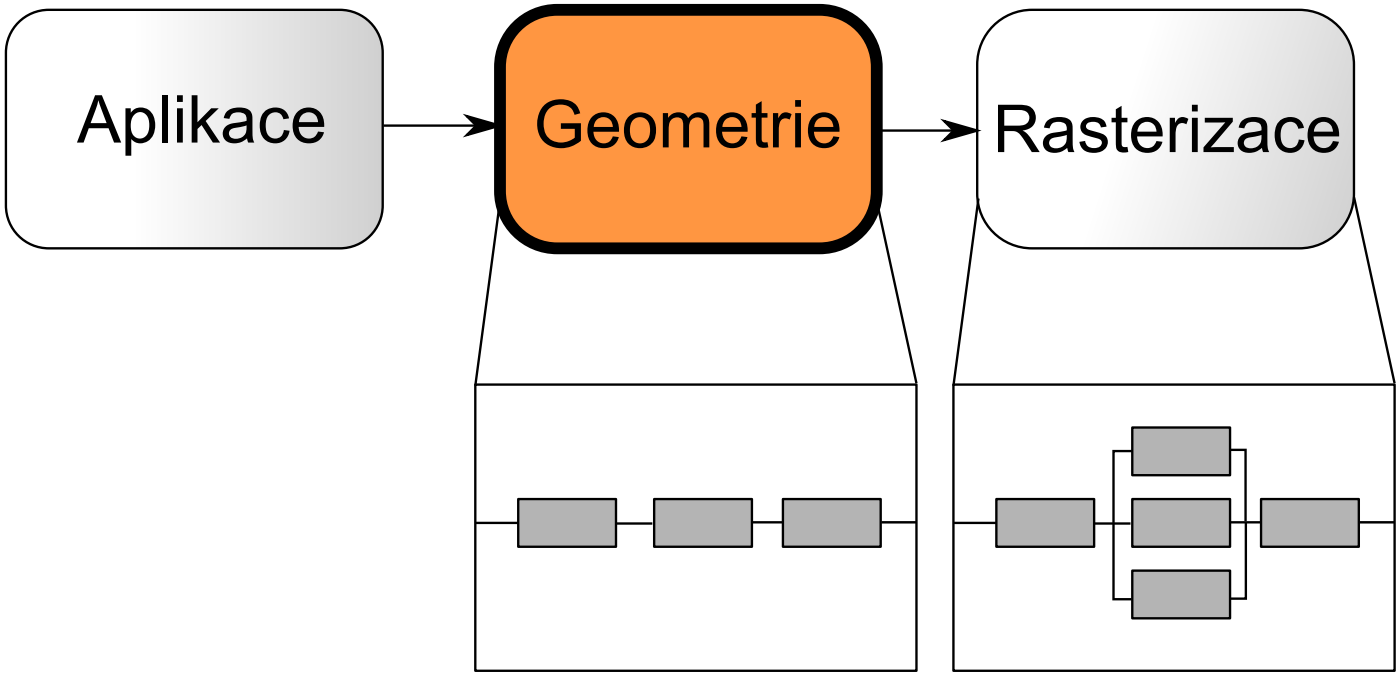
- rok **2009**:
  - **OpenGL 3.2**
  - **DirectX 11**
  - GPU **tesselation**
  - architektura nVidia **Fermi**
- rok **2010**
  - **OpenGL 4**
  - **OpenCL** - multiplatformní programování GPU pro obecné úlohy
- **další výrobci**:
  - Matrox, 3DLabs, S3, PowerVR (Kyro), SiS
  - Intel - **největší** výrobce grafických čipů

Řetězec zpracování 3D grafiky I



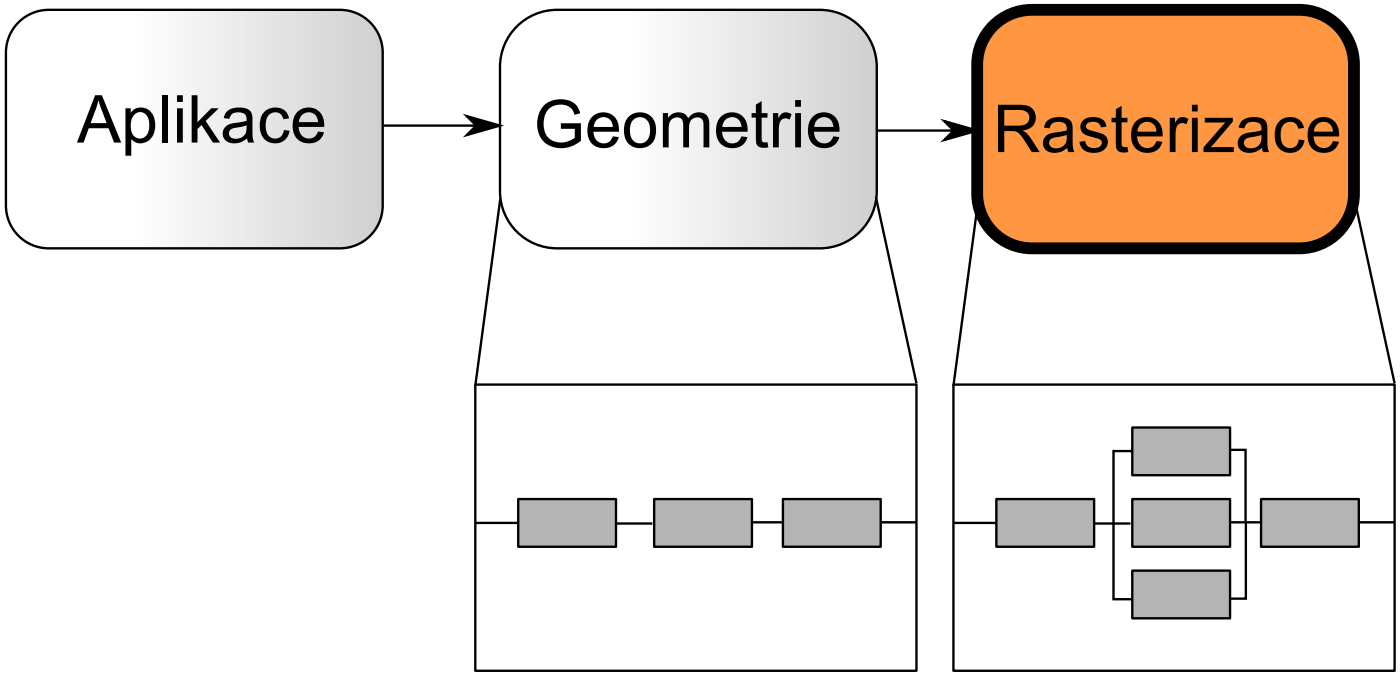
- **Aplikace**:
  - **reprezentace 3D dat** (virtuální uložení na disku i v paměti), parametrizace, šablony, ...
  - **chování objektů**: fyzikální simulace, umělá inteligence
  - **interakce** uživatele, mezi objekty (kolize, deformace), ...

Řetězec zpracování 3D grafiky II



- **Geometrie**:
  - **modelové transformace** (pomáhají aplikační vrstvě)
  - **projekční transformace** (perspektiva), **ořezávání**
  - **výpočet osvětlení** (příp. jen příprava jistých vektorů, . . .)
  - dlouhý řetězec (pipeline), řetězců může být více

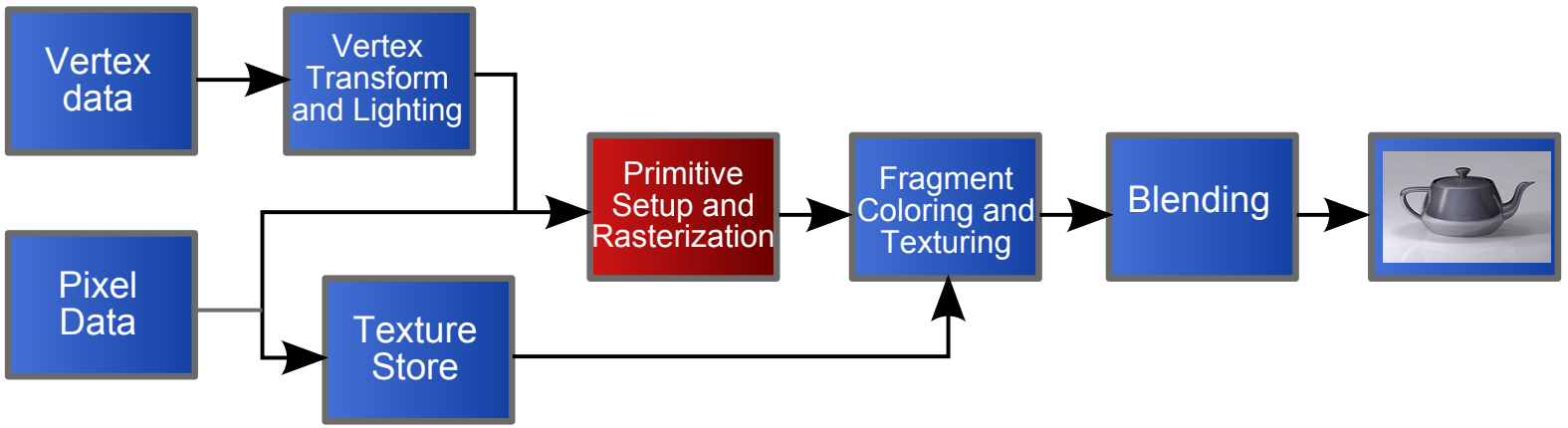
Řetězec zpracování 3D grafiky III



- **Rasterizace** (vykreslení):
  - převod objektů scény do **fragmentů** a jejich zpracování do jednotlivých **pixelů**
  - výpočet viditelnosti (*depth buffer*), mapování textur a jejich kombinace, interpolace barev, průhlednost, mlha

Schéma OpenGL (Fixed Function Pipeline)

- **HW T&L** (*Hardware Transform and Lighting*)
  - IrisGL: dříve, display list
  - od nVidia GeForce 256 (1999)
- **Rasterizace** se na čipu dělá již dlouho
  - stanice Silicon Graphics
  - API: IrisGL apod.
  - **multitexturování**: až druhá polovina 90.let

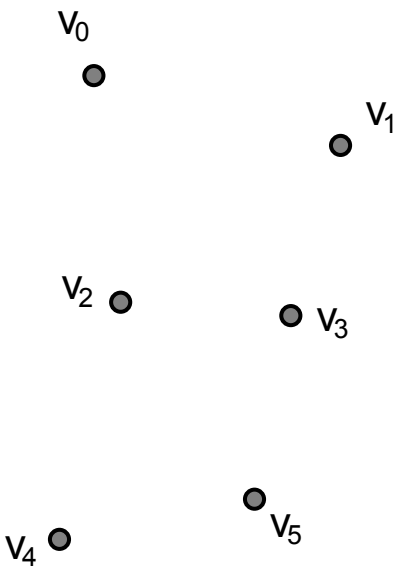


OpenGL Fixed Functionality Pipeline

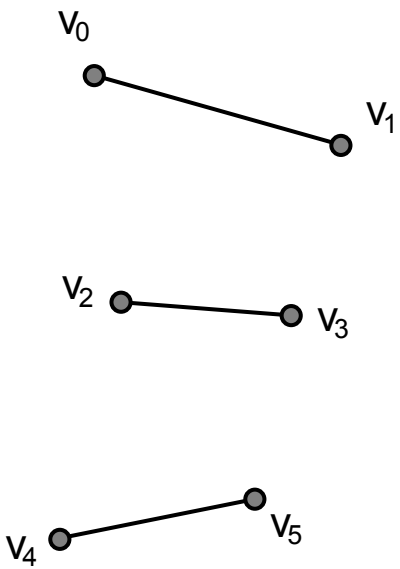
OpenGL: geometrická primitiva

- typy geometrických primitiv:
  - **bod, úsečka**, lomená čára, smyčka
  - polygon, **trojúhelník**, proužek trojúhelníků, vějíř trojúhelníků, čtyřúhelník, proužek čtyřúhelníků
  - **pozor**: v nových verzích přibylo několik dalších (např. *patch*, bude v přednášce o OpenGL 4), ovšem některá primitiva byla odstraněna např. čtyřůhelník
- zpracování **jednotlivých vrcholů**
  - glVertex, glColor, glNormal, glTexCoord, ...
  - neefektivní (mnoho volání funkcí) a dnes už není v *core profile*
- **pole vrcholů**
  - glDrawArrays, glMultiDrawArrays, glDrawElements, ...
  - glColorPointer, glVertexPointer, nebo **prokládání**

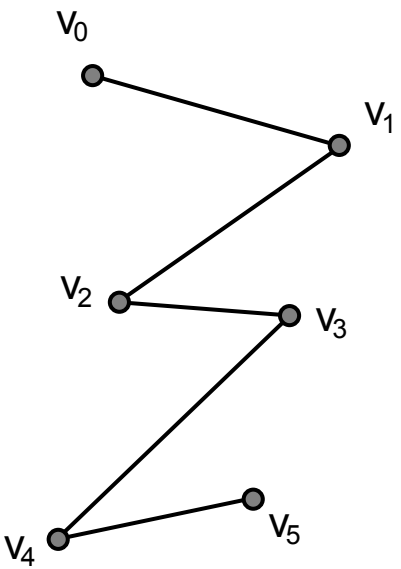
Geometrická primitiva I



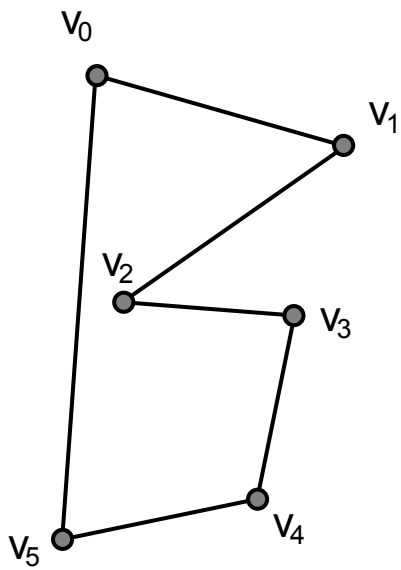
GL_POINTS



GL_LINES

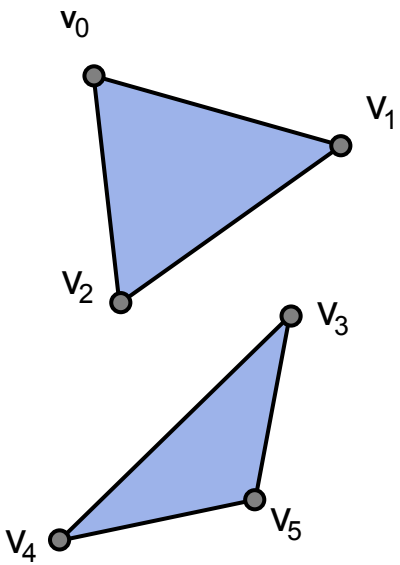


GL_LINE_STRIP

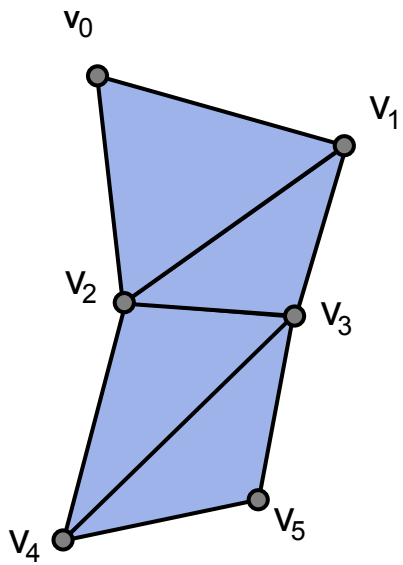


GL_LINE_LOOP

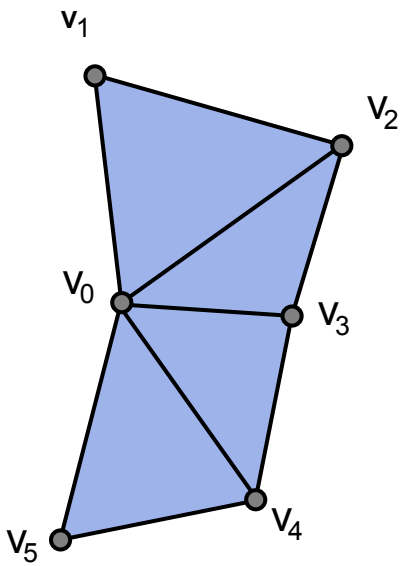
Geometrická primitiva II



GL_TRIANGLES



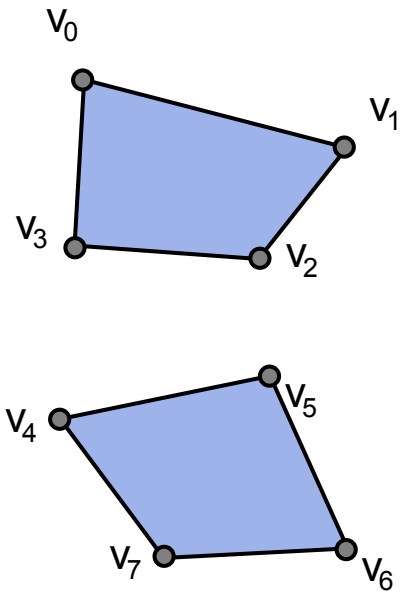
GL_TRIANGLE_STRIP



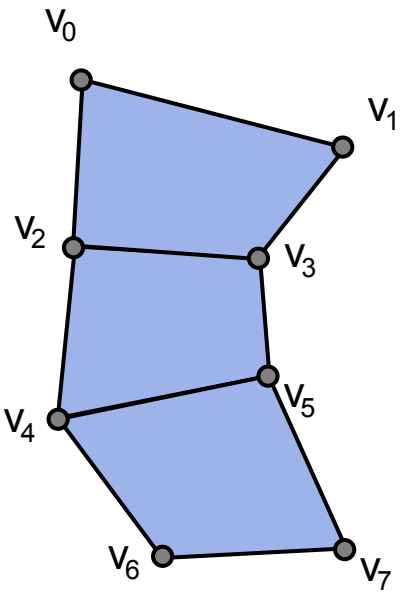
GL_TRIANGLE_FAN

Geometrická primitiva III

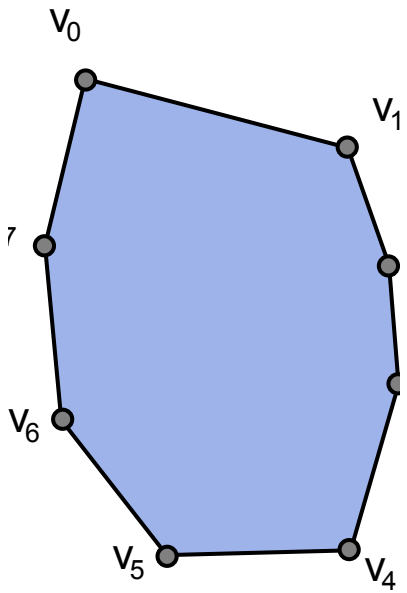
- v nových verzích OpenGL již nejsou
- existují ještě nová primitiva, budou v přednášce o OpenGL 4



GL_QUADS



GL_QUAD_STRIP



GL_POLYGON

Geometrická data na serveru

- až od **OpenGL 1.5**
- dnes ovšem **velmi** důležité v OpenGL
- buffer **na straně grafického serveru** obsahující geometrická data
  - založení bufferu: **glGenBuffers**, **glBindBuffer**
  - **zadání dat** z pole: glBufferData, glBufferSubData
  - **mapování** do paměti aplikace: glMapBuffer, glUnmap ...
- práce s **klientskou** pamětí nebo bufferem:
  - glColorPointer, glNormalPointer, glVertexPointer, ...
  - dnes se uz nepoužívá, skoro vše musí být na straně serveru (zůstalo pouze glVertexAttribPointer)

Zpracování vrcholů

- **transformace vrcholů** modelovacími a projekčními maticemi
- glMatrixMode
- glLoadIdentity, glLoadMatrix, glMultMatrix
- glRotate, glScale, glTranslate, ...
- **osvětlovací atributy**
  - glLight, glLightModel, glMaterial

Sestavení a zpracování primitiv

- **sestavení** (Assembly)
  - určení, kolik vrcholů primitivum potřebuje
  - shromáždění balíčku dat a odeslání dál
- **zpracování** primitiv
  - ořezávání (*clipping*)
  - projekce do zorného objemu (*frustum*) - dělení w
  - projekce a ořezání do 2D okénka (*viewport*)
  - odstranění odvrácených stěn (*culling*) - jednostranná vs. oboustranná primitiva

Rasterizace, fragmenty

- **rasterizace** = vykreslení vektorových primitiv
  - rozklad geometrických objektů na **fragmenty**
  - geometrické objekty: body, úsečky, trojůhelníky
- **fragment**
  - **rastrový element**, který potenciálně přispívá k barvě nějakého pixelu
  - velikost: stejná nebo menší než u pixelu (anti-aliasing)
  - "balíček dat" procházející rasterizační jednotkou GPU:
    - vstup/výstup: **x,y,z** (pouze hloubku lze měnit!)
    - texturovací souřadnice t_0 až t_n
    - lesklá a difusní barva, textury, koeficient mlhy, uživatelská data, ...
    - výstupní barva **RGB** a neprůhlednost α (frame-buffer op.)

Interpolace ve fragmentech

- atributy fragmentů se automaticky **interpolují z hodnot ve vrcholech**
  - hloubka (**z** nebo **w**)
  - texturové souřadnice
  - barvy (lesklá a difusní složka)
  - uživatelské atributy, ...
- rychlé **HW interpolátory**
- **perspektivně korektní** interpolace
  - jen [**x**,**y**] se mění lineárně
  - ostatní veličiny vyžadují jedno dělení na každý fragment

Zpracování fragmentů

- **texturovací operace**
  - maximálně **optimalizované** operace
  - výběr barvy z texturovací paměti
  - interpolace texelů:
    - lineární, mip-mapping, anisotropic filtering, ...
    - speciální HW pro načítání a interpolaci texelů
  - kombinace několika textur (výběr z mnoha operací)
  - zvláštní efekty (bump-mapping, environment mapping)
- výpočet **mlhy**
  - podle hloubky **z**
- **kombinace** primární a sekundární barvy (diff., spec.)

Upotřebení fragmentů (per-fragment op.)

- **lokalizace** fragmentu
  - do kterého pixelu patří?
- test **uživatelského ořezání** (glScissor)
- test odmítnutí podle **průhlednosti** (glAlphaFunc)
- test **šablony** (*stencil test*, glStencilOp)
- test **hloubky** (*depth test*, glDepthFunc)
- **kompozice barvy** (podle případné průhlednosti)
- **dithering** (příp. konverze barvy pro frame-buffer)
- **logická operace** (glLogicOp)

Globální operace s frame-bufferem

- **frame-buffer** se skládá z několika **bufferů**
  - front,back,left,right, . . . (double-buffering, stereo)
  - nastavení aktuálního kreslícího bufferu (glDrawBuffer)
- **inicializace** bufferů (glClear)
  - glClearColor, glClearDepth, glClearStencil, glClearAccum
- řízení přenosu dat do **grafického serveru**
  - glFlush: neblokující volání, všechny dosavadní příkazy se ukončí co nejdříve (příkazy mohou být bufferované, např. pro lepší výkon)
  - glFinish: blokující, dokončí celou kresbu v daném kontextu, čeká dokud vše na serveru není hotovo

Přenos rastrových obrázků

- **aplikační paměť → frame-buffer**
  - dříve se přímo dalo použít vykreslování: glDrawPixels, glBitmap
  - efektivnější uložit do textury a vykreslit pomocí texturovaného polygonu
- **aplikační paměť → texturová paměť**
  - jen přes *unpacking* a *pixel transfer*
  - glTexImage, glTexSubImage
- **přenos uvnitř OpenGL**
  - glBlitFramebuffer (dříve glCopyPixels)
  - glCopyTexImage, glCopyTexSubImage: cílem je textura
- **frame-buffer → aplikační paměť**
  - glReadPixels: pomalá operace i na rychlých sběrnicích, je třeba vyprázdnit pipeline

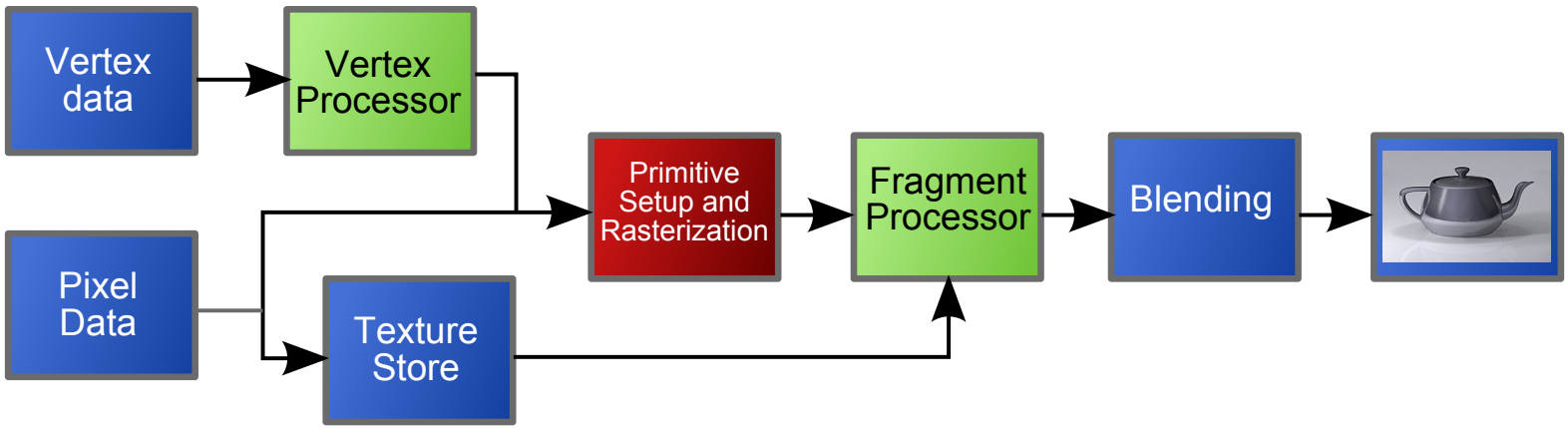
Konverze a rastrové operace

- **pixel unpacking**
  - převod z formátu aplikace do OpenGL (*coherent stream of pixels*, *group of pixels*)
  - {RGB[α] — depth — stencil}[][]
  - zdrojový formát, délka rozkladové řádky, offsety
  - nastavení: glPixelStore
- **pixel packing**
  - opačný převod

Schéma OpenGL (Programmable Pipeline)

- přidáno oficiálně v **OpenGL 2.0** (pomocí rozšíření i dříve)
- zůstal i fixní řetězec
- stejné až do roku 2009 - OpenGL 3.1

- **vertex shading** vylepšil fixní T&L
- **fragment shading** vylepšil zpracování fragmentů



Vertexový procesor

- nahrazuje **modul zpracování vrcholů**:
  - transformace vrcholů
  - transformace a normalizace normálových vektorů
  - výpočet/transformace texturovacích souřadnic
  - výpočet osvětlovacích vektorů
  - nastavení materiálových konstant do vrcholů
- **nemůže ovlivnit**
  - **počet vrcholů** (nelze přidat ani ubrat vrchol)
  - typ/topologii geometrických primitiv
    - částečné řešení: degenerace primitiv

Fragmentový procesor

- nahrazuje **modul zpracování fragmentů**:
  - aritmetické operace s interpolovanými hodnotami
  - čtení dat z textur
  - aplikace textur
  - výpočet mlhy
  - závěrečná syntéza barvy fragmentu
  - možnost modifikace hloubky fragmentu (ale vyřadí *early depth test*)
- **nemůže ovlivnit**
  - **počet fragmentů** (nelze přidat ani ubrat fragment)
  - **polohu fragmentu** na obrazovce [x,y]

Programování procesorů v GPU

- **Vertex shader**
  - kód zavedený ve vektorovém procesoru
- **Fragment shader**
  - kód zavedený ve fragmentovém procesoru
- aplikační programátor **může** tyto kódy měnit
  - **HW nezávislé** programovací jazyky
  - **mikrokód pro GPU** se kompiluje až v době běhu aplikace (omezení - různé profily/verze)
  - low-level instrukce (jazyk se podobá assembleru)
  - nebo **vyšší jazyky**: Cg, HLSL, GLSL

OpenGL 3

- odstraněna FFP a přímý mód (glBegin/glEnd)
- nový systém kontextových profilů - způsob odstraňování *staré* funkcionality
- textury v plovoucí řádové čárce
- HW instancing
- **geometry** shaders
  - možnost generovat/odstraňovat vrcholy i primitiva
  - hned za vertex shaderem
- GLSL **1.3-1.5**
  - nativní bitové operace a práce s celými čísly
  - rozšíření mnoha dalších parametrů

OpenGL 4

- nová primitiva (GL_PATCHES), trojúhelníky a čáry se *sousedností* (GL_TRIANGLES_ADJACENCY), . . .)
- **HW tesselation**
  - dva nové shadery: **Tesselation Control** Shader, **Tesselation Evaluation** shader
- přímé propojení s externími *výpočetními* API, např. **OpenCL** bez CPU
- možnost ukládat binární verzi shaderů
- GLSL **4.00**
  - podpora pro 64-bitovou plovoucí čárku
  - shaderové podprogramy
  - oddělené shaderové řetězce

Literatura

- Tomas Akenine-Möler, Eric Haines: **Real-time rendering, 2nd edition**, A K Peters, 2002, ISBN:1568811829
- OpenGL Architecture Review Board: **OpenGL Programming Guide: The Official Guide to Learning OpenGL**, Addison-Wesley, nejnovější vydání (aktuálně 8. vydání pro OpenGL 4.1)
- Randi J. Rost, Bill Licea-Kane: **OpenGL Shading Language, 3rd Edition**, Addison-Wesley,
- Randima Fernando, Mark J. Kilgard: **The Cg Tutorial**, Addison-Wesley, 2003, ISBN: 0321173481

Literatura

- The Khronos Group: **The OpenGL Graphics System: A Specification (Core/Compatibility profile)**, `http://www.opengl.org/registry/`
- Christophe Riccino: **OpenGL reviews**, `http://www.g-truc.net/post-opengl-review.html`
- Wikipedia: `http://en.wikipedia.org/wiki/OpenGL`
- OpenGL tutorials: `http://nehe.gamedev.net`