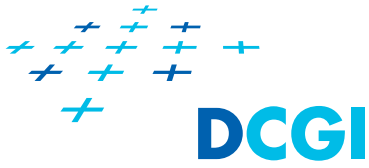


Programování grafiky
ÚVOD

Petr Felkel

Katedra počítačové grafiky a interakce, ČVUT FEL

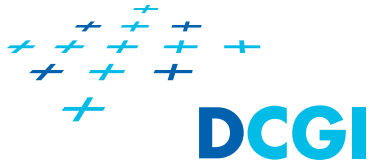
místnost KN:E-413 (Karlovo náměstí, budova E)

E-mail: felkel@fel.cvut.cz

S použitím materiálů Bohuslava Hudce, Jaroslava Sloupa, Ondřeje Karlíka a Vlastimila Havrana

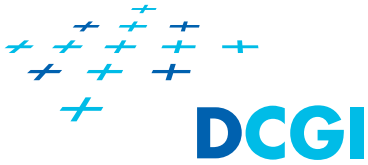
Poslední změna: 20.2.2013

Cíle předmětu programování grafiky



- Porozumět základům počítačové grafiky, např.:
  - Modelování objektů v povrchové reprezentaci
  - Osvětlování
  - Nanášení textur
  - Animacím
- Umět naprogramovat
  - Jednoduché interaktivní aplikace
  - Založené na OpenGL
  - Nezávislé na platformě (PC / Mac / vestavné systémy / web,…)

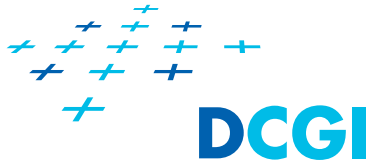
Organizace



- Přednášky
  - Teorie počítačové grafiky – základy pro grafické aplikace běžící v reálném čase (real time)
  - Syntéza obrazu s použitím OpenGL od verze 3.1
  - Demonstrační příklady a kódy jednotlivých technik
- Cvičení
  - Detaily o OpenGL a GLSL
  - Jednoduché úlohy
  - Semestrální projekt (program, web, prezentace)
- Podrobnosti na webu předmětu
  http://service.felk.cvut.cz/courses/A7B39PGR/

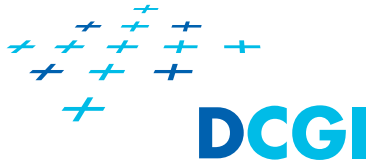
Ukázky semestrálních prací

Ukázky



- Semestrální práce
- Demo na webu

Krátký úvod do počítačové grafiky



- Různé pohledy na počítačovou grafika
- Obory v počítačové grafice
- Pixely a antialiasing
- Geometrické modelování
- Textury
- Syntéza obrazu
- Problém viditelnosti
- Negeometrické vlastnosti scény
- Kvalita výsledného zobrazení

Úvod do PG zpracován částečně na základě textu F. Duranda, MIT, 2007.

Dělení počítačové grafiky dle účelu [Karlík]

- 2D grafika
  - DTP – noviny, časopisy
  - Grafická uživatelská rozhraní- GUI
  - Výkresová dokumentace
- 3D modelování
  - CAD/CAM systémy
- **3D grafika** [Karlík]
  - Vizualizace
  - Filmová grafika
  - Realtime grafika - hry
  - …

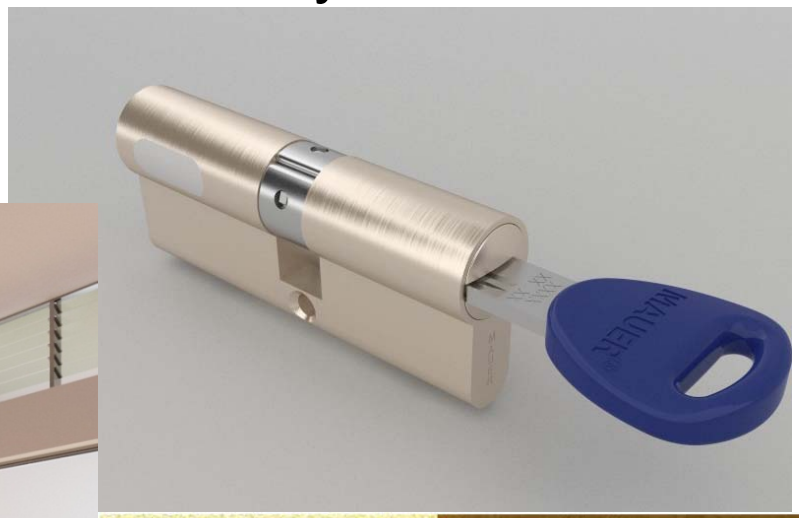
http://www.cs.utah.edu/npr/

Vizualizace

DCGI

[Karlík]

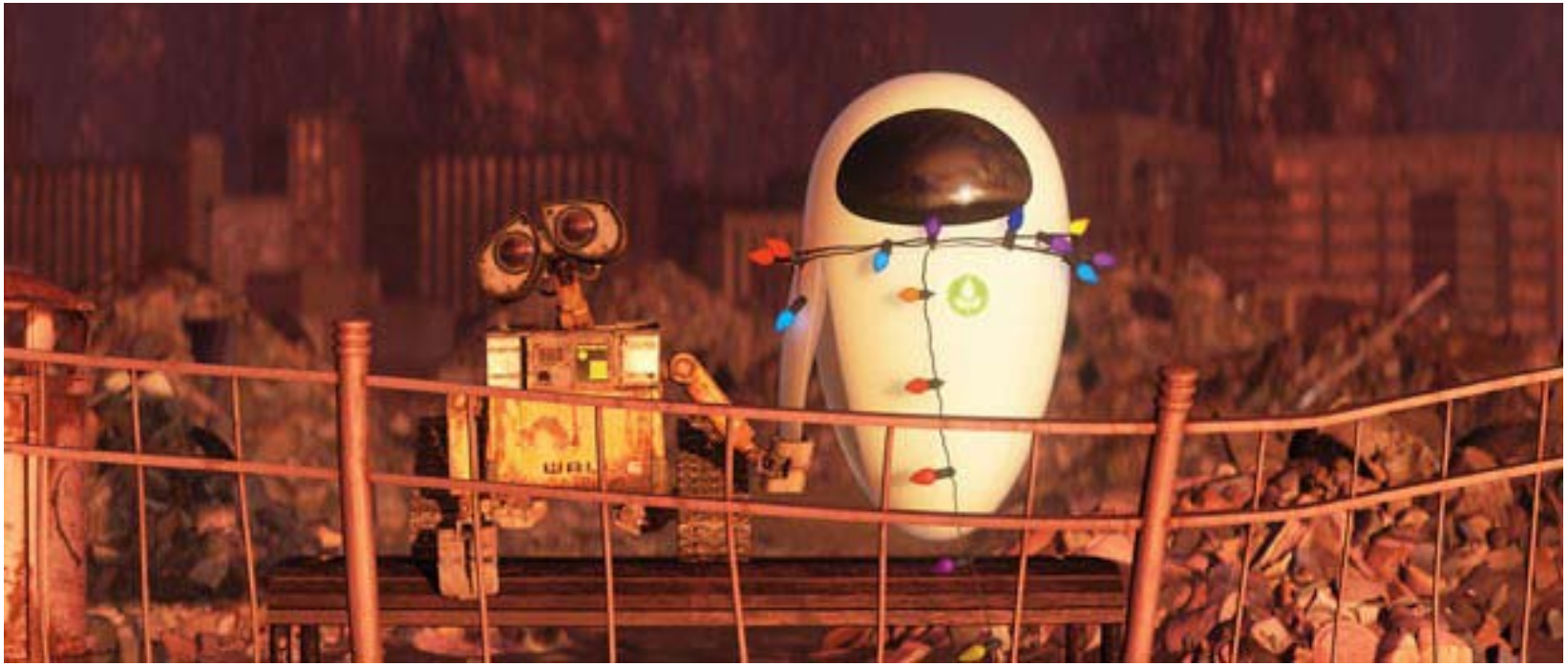
- Reklama, vizualizace produktů, architektury
- Fotorealismus
- Delší výpočetní čas



Filmová grafika

[Karlík]

- Obrovský rozsah dat
- Umělecká kontrola nad výsledkem
- Renderman + vlastní SW studií



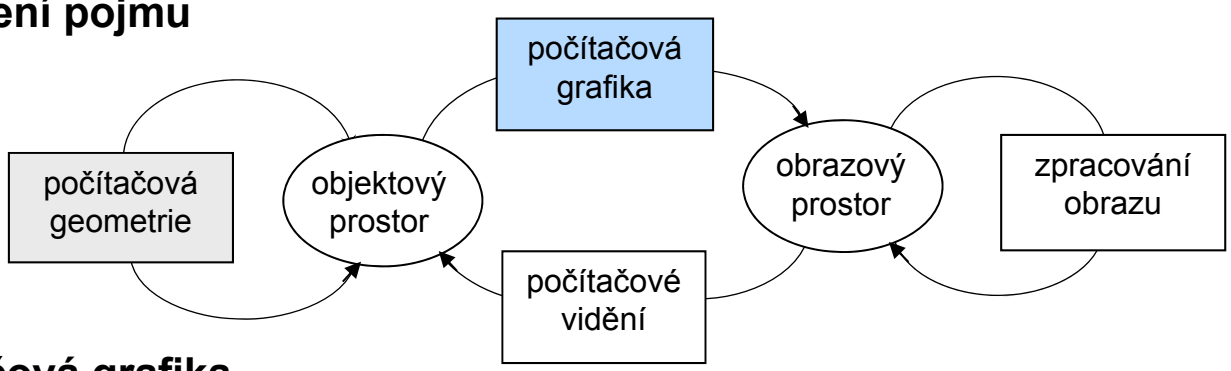
Realtime grafika

[Karlík]

- Nízký počet polygonů
- Rasterizace
- GPU
- 30+ FPS
- Fake

Vztah počítačové grafiky a vidění

- **Vymezení pojmu**



- **Počítačová grafika**

  transformace dat z objektového (3D) do obrazového prostoru (2D)

- **Počítačová geometrie**

  zpracování a geometrických a topologických dat v objektovém prostoru

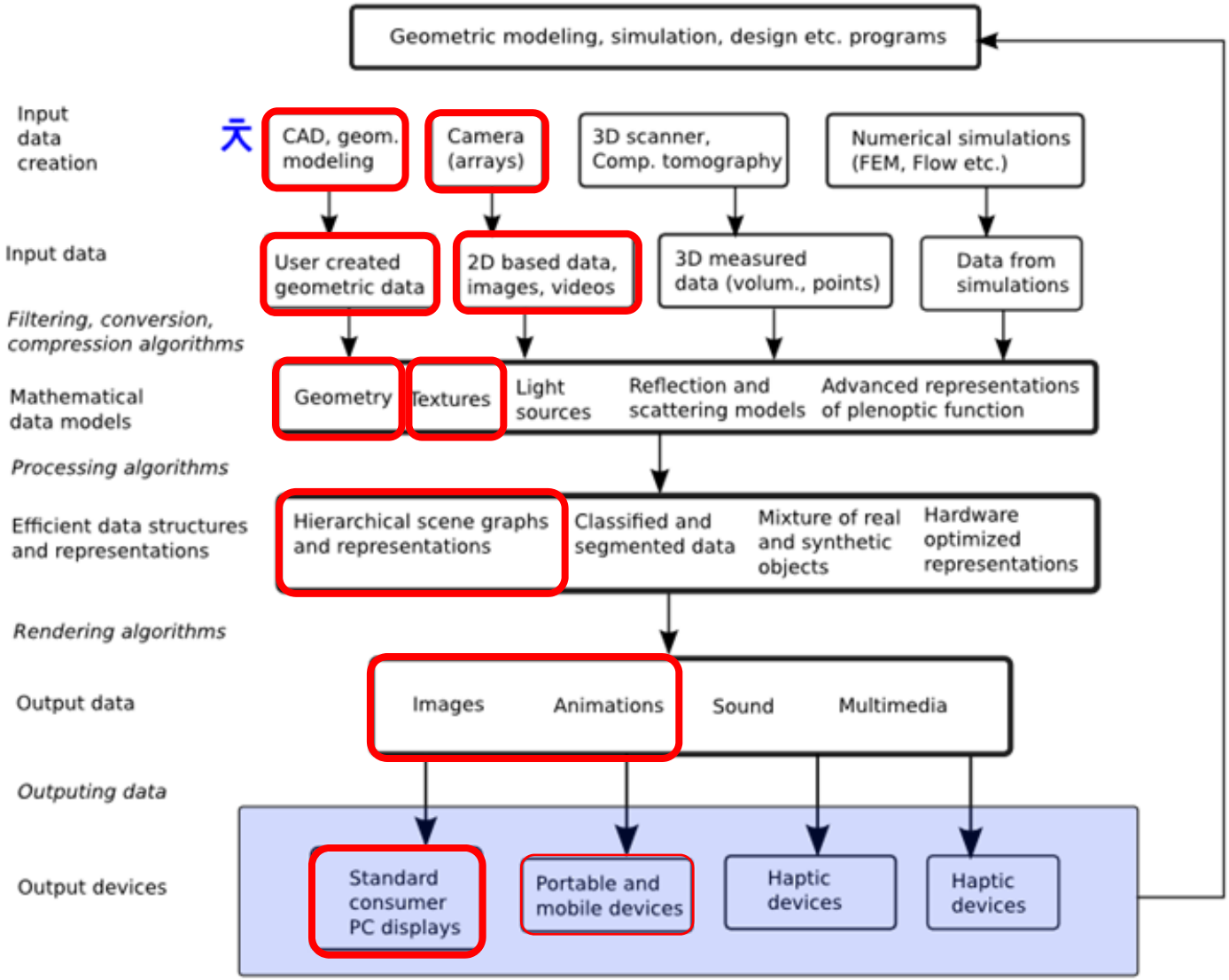
- **Zpracování obrazu**

  Zpracování nestrukturovaných rastrových obrázků
  (zlepšování kvality obrazu, vyhledávání obrysů, ..)

- **Počítačové vidění**

  Na základě analýzy obrazu se hledají objekty a vztahy mezi objekty v objektovém prostoru.

Oblasti počítačové grafiky podrobně



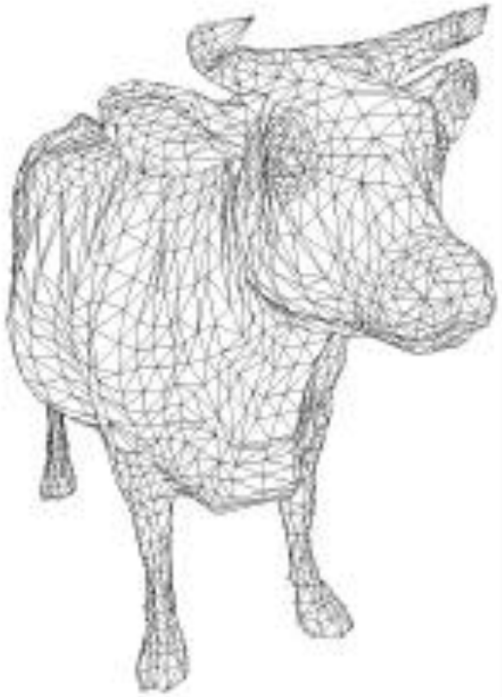
Obrázek, pixely a antialiasing

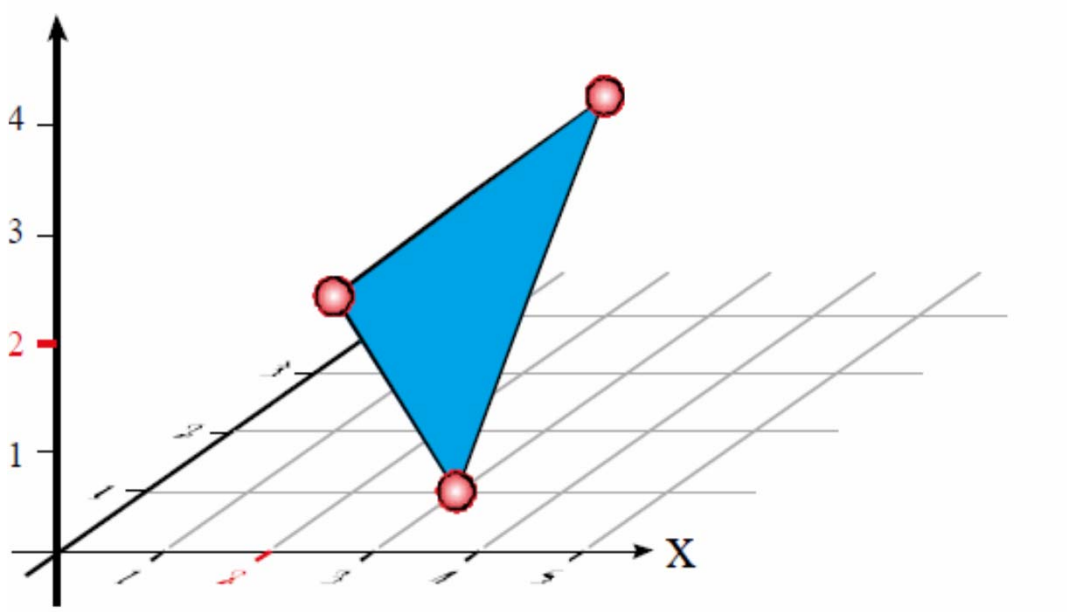
- Obrázek
  - Omezené rozlišení (Daný počet pixelů)
  - Zubaté okraje, artefakty
  - Nutno vyhlazovat

bez vyhlazování

s vyhlazováním

Geometrické modelování x zobrazování

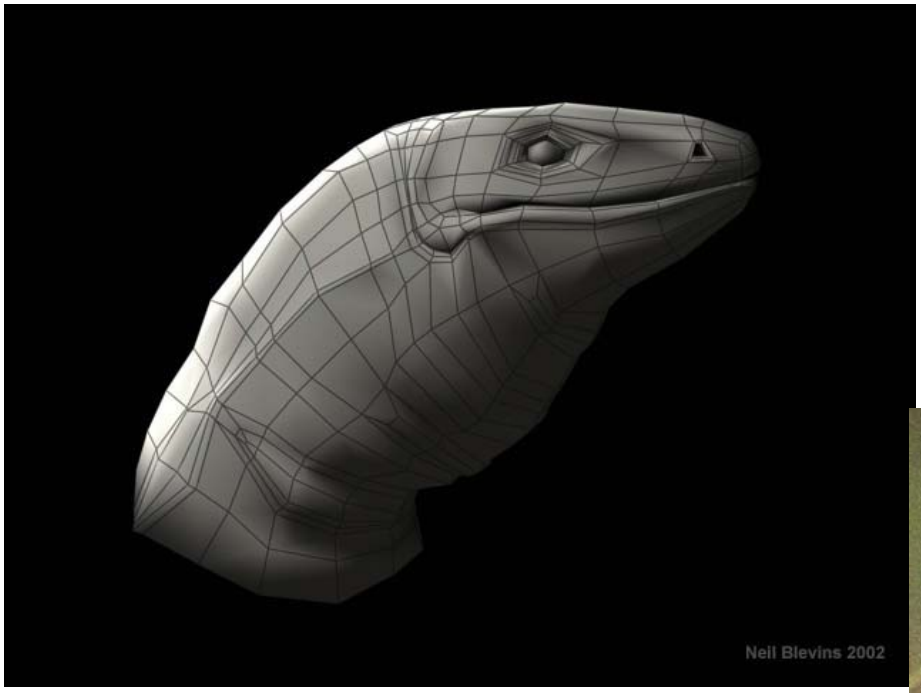




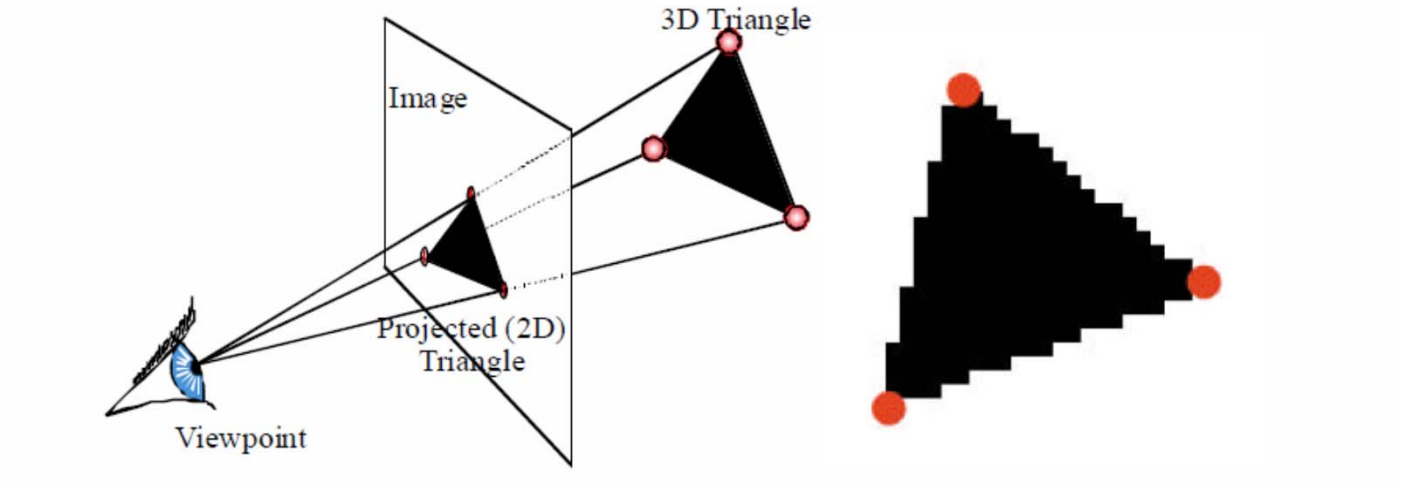
Modelování – často komplexní plochy

Zobrazování – trojúhelníky ve 3D (teselace)

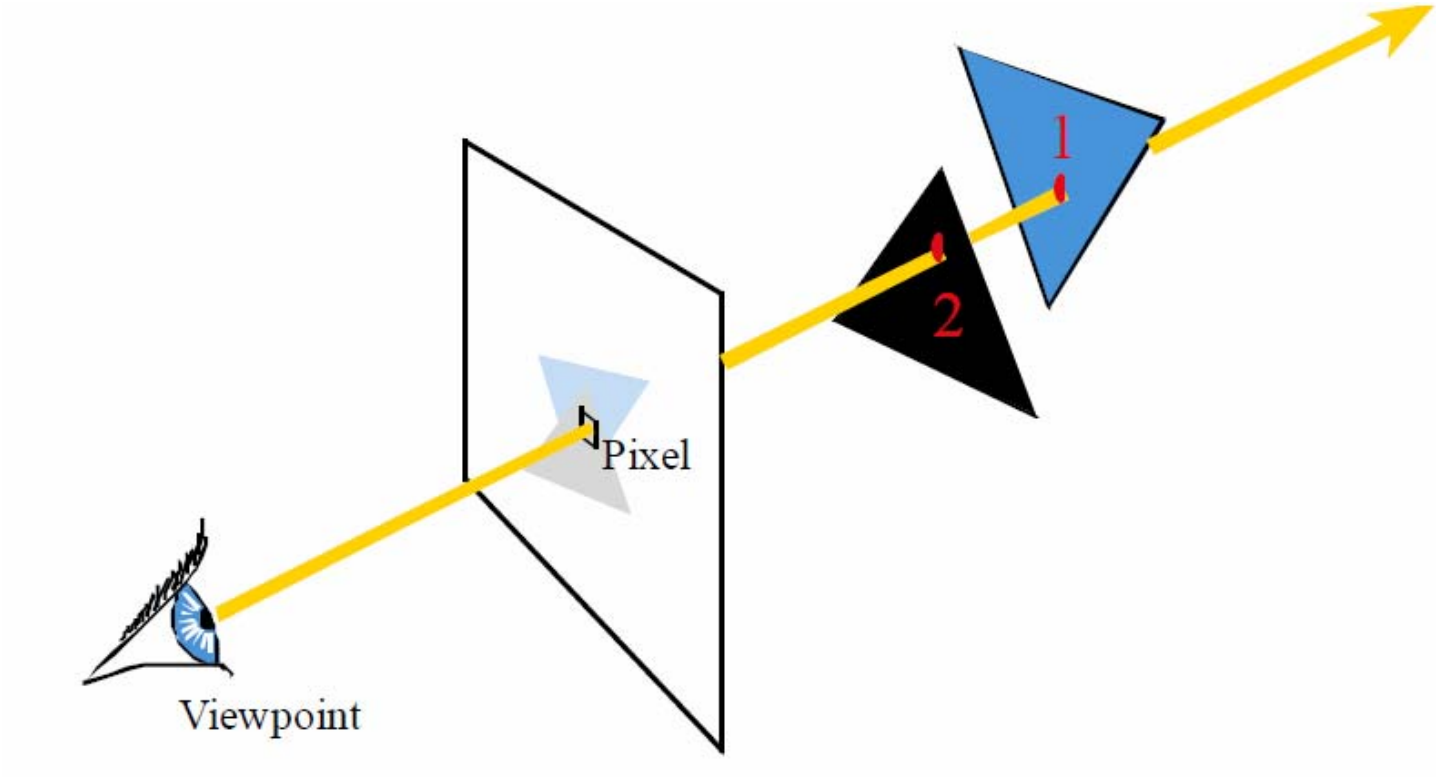
Textury



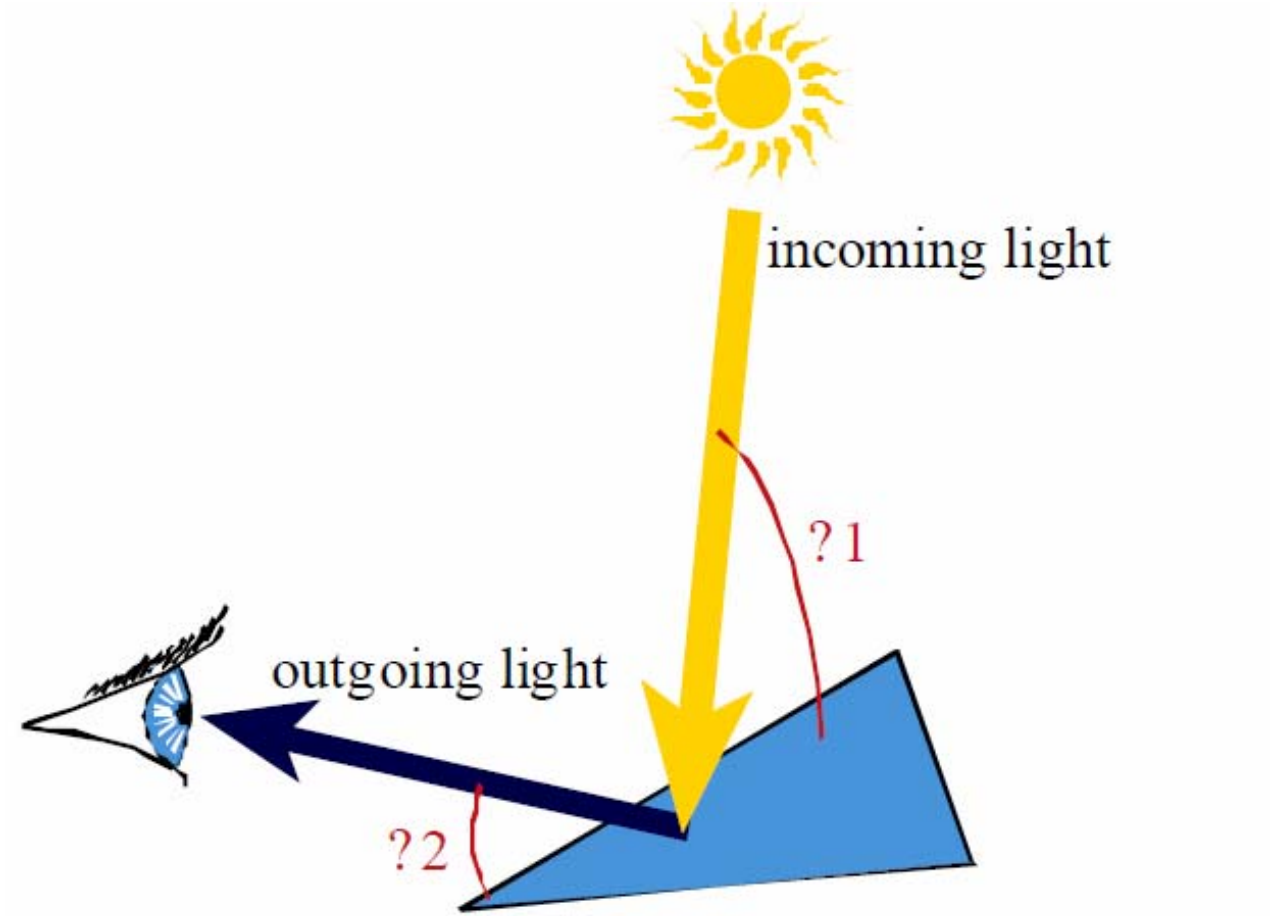
Základ syntézy obrazu



Problém viditelnosti

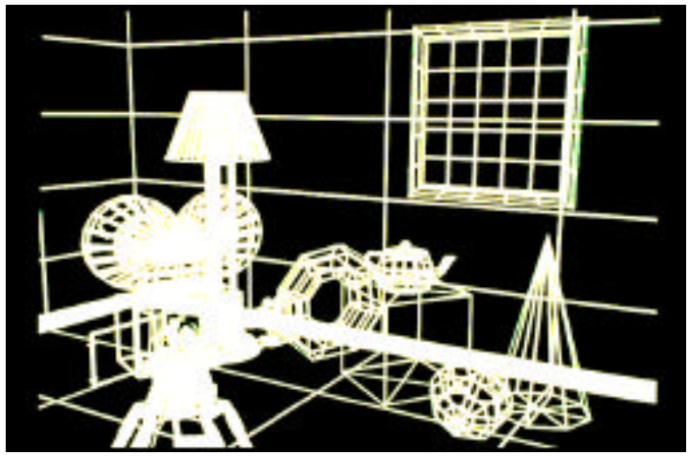


Negeometrické vlastnosti objektů

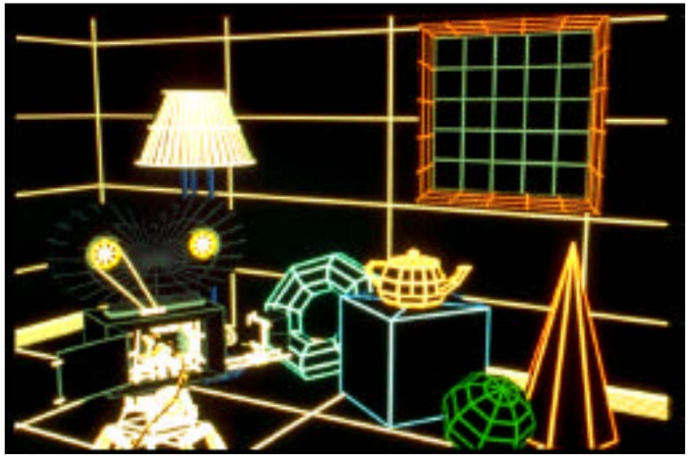


Různá kvalita syntetizovaného obrazu (obrázky fy PIXAR)

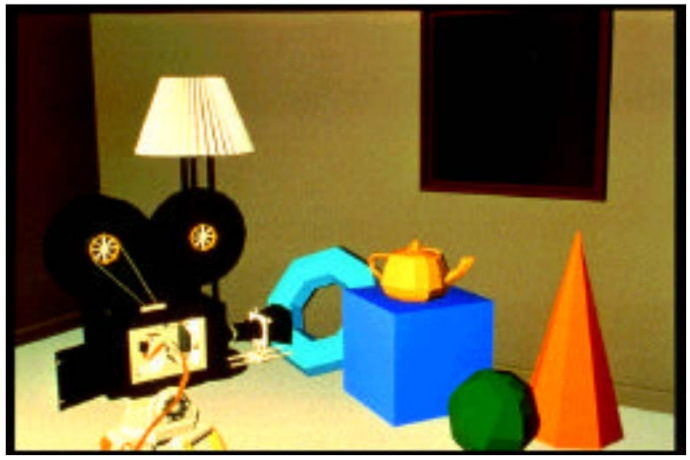
Polygonální model – drátové zobrazení



Polygonální model –
drátové zobrazení s viditelností



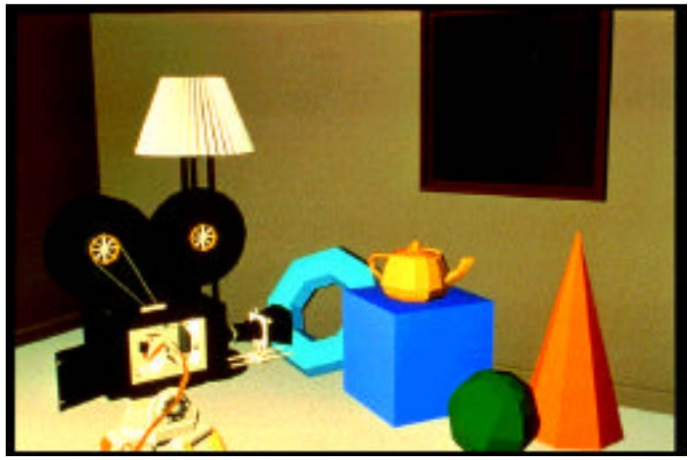
Polygonální model – konstantní stínování



Polygonální model – interpolované stínování

Různá kvalita syntetizovaného obrazu (obrázky fy PIXAR) II

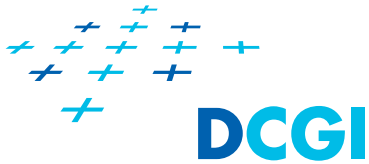
Polygonální model – konstantní stínování



Polygonální model – interpolované stínování

Mapování textur

Stíny

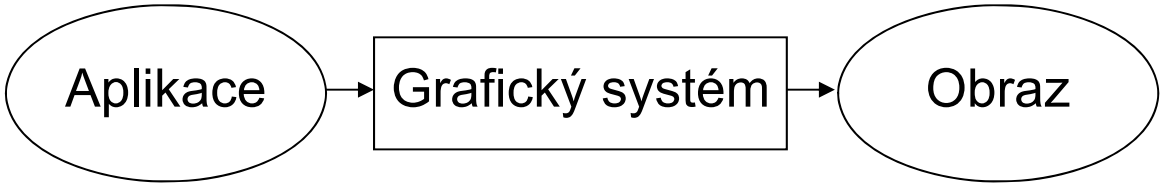


Grafická rozhraní, knihovna OpenGL, GLUT

Grafický systém

[Hudec: Skripta ZPG, 2007]

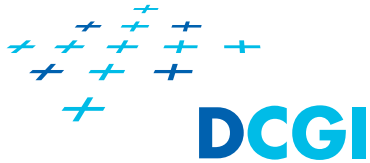
= speciální programy pro zpracování grafických úloh



Typické funkce:

1. Generování obrazu a reprezentace scény
   (elementy, atributy, struktura obrazu)
2. Transformace mezi souřadnými systémy a ořezávání
3. Řízení vstupu, výstupu
4. Uchování obrazu pro pozdější použití
   - archivní a grafické soubory, obrázky, animace

Grafická rozhraní a standardy



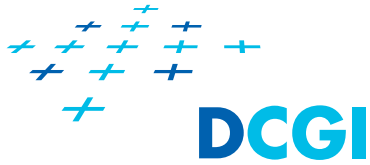
Rozhraní

- definována mezi vrstvami systému
  (aplikace - grafický systém - logické zařízení – HW,…)

Standardizace rozhraní

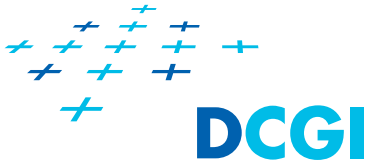
- kvetla dříve (ISO GKS, PHIGS, PHIGS+) [80],
- teď fungují hlavně *de facto* standardy (Adobe postscript and portable description format (PDF), Khronos OpenGL, OpenCL, Pixar RenderMan, atd.)

Grafická rozhraní a standardy (2)



- Přibývají **3D knihovny nejen pro hry** (*engines*)
  - komplexní knihovny nad OpenGL a/nebo DirectX
  - s širokým spektrem funkcí
    (např. fyzika, graf scény, soubory, zvuk…)
  - OGRE, Irrlicht , OpenSceneGraph,... … viz [engines]
- A **jazyky pro programování GPU**
  - OpenGL Shading Language (GLSL),
  - CUDA
  - Cg & HLSL,
  - Sh Shading Language Assembler (viz GL_ARB_fragment_program & GL_ARB_vertex_program),…
- My se budeme zabývat **OpenGL a GLSL**
  - multiplatformní

Co je OpenGL?



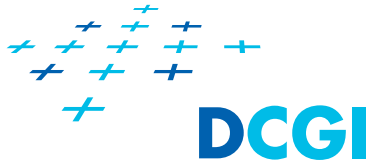
Grafická knihovna (Open Graphics Library)

(Rendering API = Application Programming Interface)

- SW rozhraní k rastrovému grafickému HW
- generuje vysoce kvalitní barevné obrázky
- nezávislé na operačním systému (OS) a na použitém HW
- nestará se o okna, události, formáty souborů,…

- HW musí mít obrazovou paměť (rastr pixelů)
  (OpenGL nepracuje na plotrech)

Knihovny kolem OpenGL

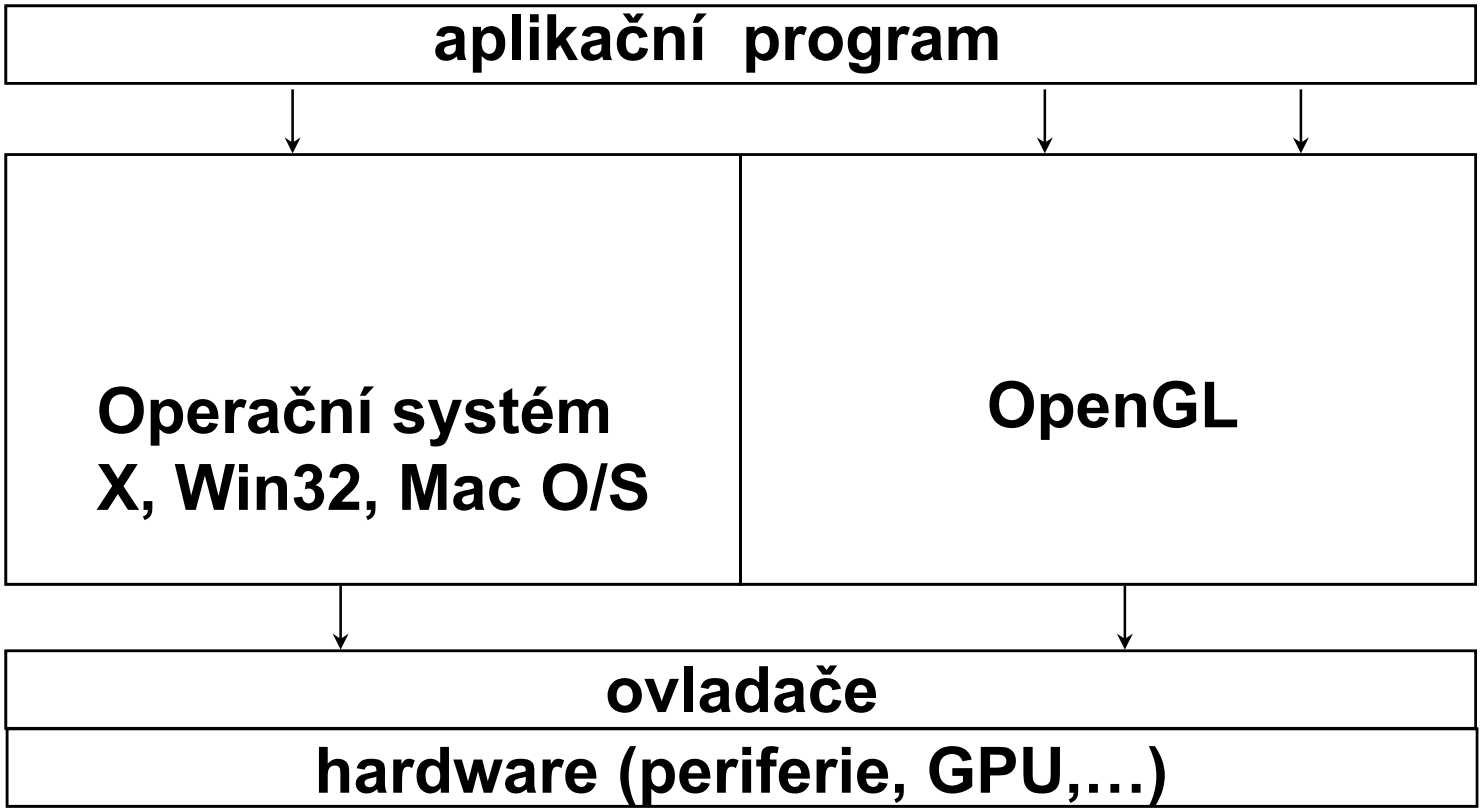


- **OpenGL (verze 3.3)**
  - minimální knihovna
  - geometrická primitiva (body, úsečky, trojúhelníky, pásy troj.…)
  - obrazová primitiva (rastrové obrázky) – textury
  - předává balíky dat grafické kartě (GPU)
  - (osvětlení, stínování, operace s pixely) –> převzaly shadery

- **GLUT** (OpenGL Utility Toolkit)
  - přenositelné okenní rozhraní (Mark Kilgard 1994-6)
  - není součástí OpenGL
  - využíváme variantu *freeglut*, která se dále vyvíjí
  - existují i jiné alternativy – např. knihovna *SDL* a *Qt*

Knihovny kolem OpenGL (2)

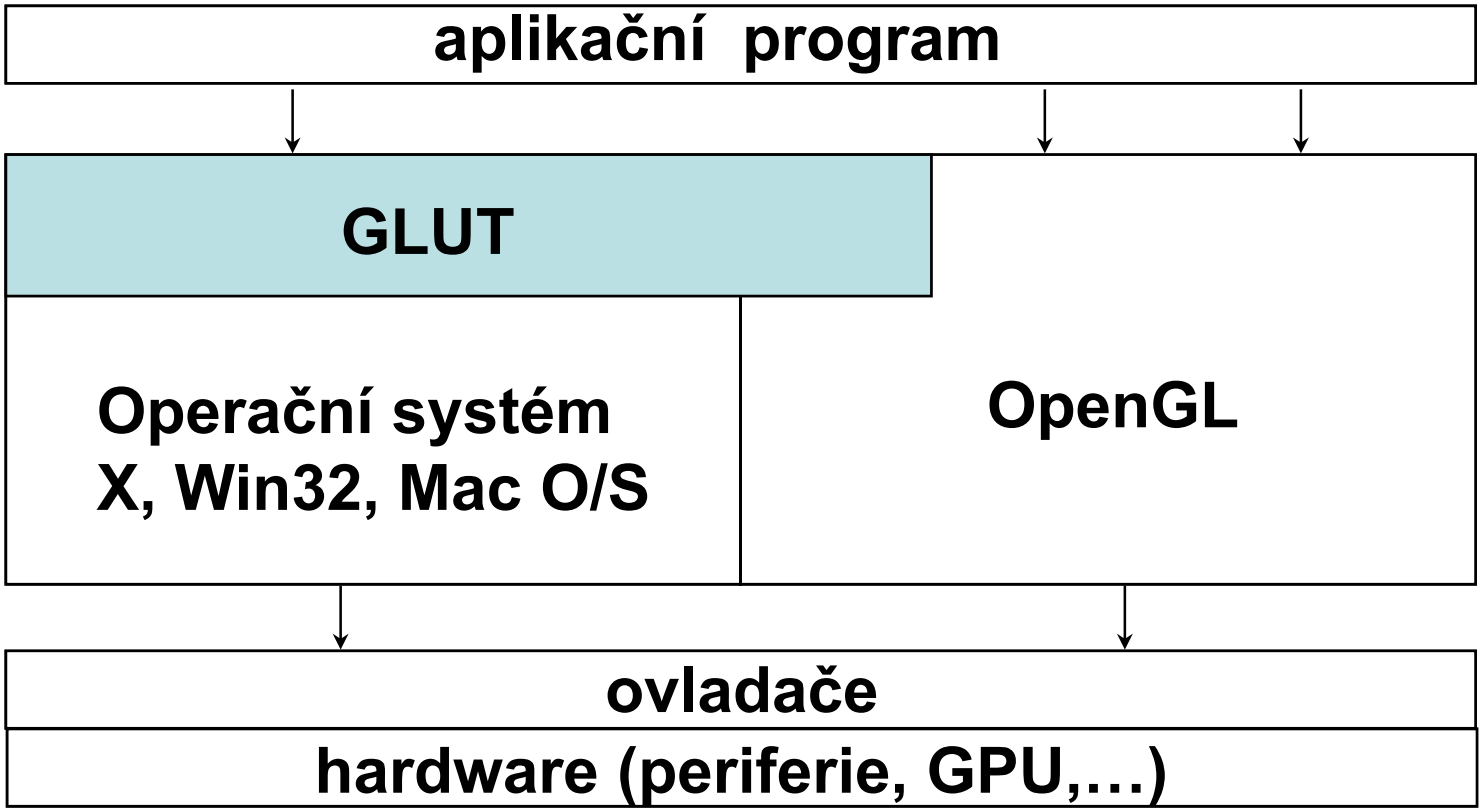
Aplikace *závislá* na operačním systému



Courtesy Siggraph2004 Course 29

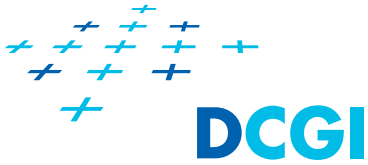
Knihovny kolem OpenGL (3)

Aplikace *nezávislá* na operačním systému



Courtesy Siggraph2004 Course 29

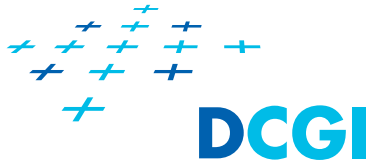
OpenGL je standard



- **Definice rozhraní**
  - ARB (Architectural Review board)
  - 1992-2006 nezávislá komise, členy: 3DLabs, Apple, ATI, Dell, IBM, Intel, NVIDIA, SGI, Sun Microsystems, chvíli i Microsoft
  - 2000 založili Khronos group (OpenGL | ES)
  - Od září 2006 – ARB Working Group v rámci konsorcia Khronos Group (placené členství, vývoj na všech platformách)
  - OpenGL™ je ochranná známka SGI

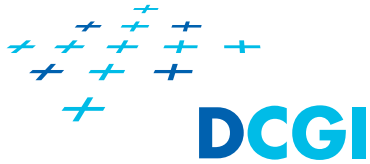
- **Implementace**
  - Výrobci HW
    (jediní potřebují licenci od SGI, pro Windows od Microsoftu)
  - Testy provádí ARB (conformance tests) – někteří členové (adopters)
- **Používání OpenGL v programech - zdarma**

Výhody OpenGL



- Jednoduchost & srozumitelnost
- Konzistentní implementace & design (ověřeno dlouholetou existencí)
- Rozsáhlá (aktualizovaná) dokumentace
- Přenositelnost
- Pro embedded systémy (OpenGL ES) i superpočítače
- Další rozvoj je dán konsorciem firem
- Otevřenost a rozšiřitelnost!!
  - výrobci HW mohou rozšiřovat funkcionalitu sami (viz GL Extensions)
  - např. v Direct3D je všechno vydáno na „milost a nemilost“ Microsoftu, který jediný spravuje a aktualizuje Direct3D

Evoluce OpenGL



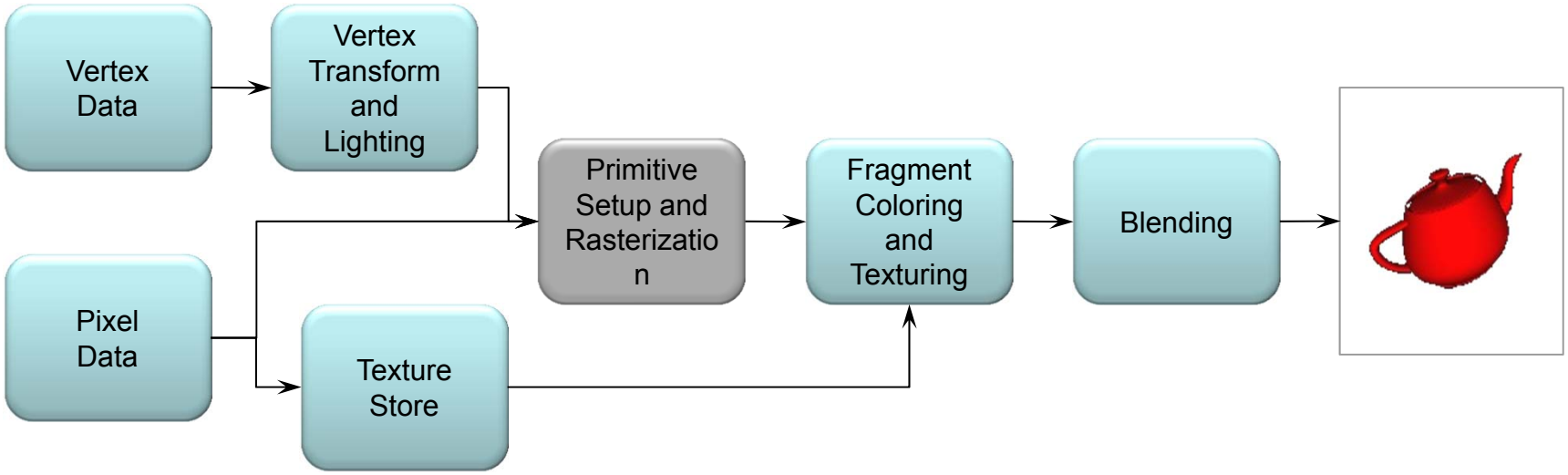
- 1994 - OpenGL 1.0 - fixní proudové zpracování (“fixed pipeline”)
- 2004 - OpenGL 2.0 - programovatelnost pomocí tzv. „shaders“
- 2008 - OpenGL 3.0 - zavedení dopředné kompatibility a možnosti zrušit staré funkce
- 2009 - OpenGL 3.1 - kompletně odstraněno fixní proudové zpracování, nutnost použití „shaders“
- 2010 - OpenGL 4.0 - řízení teselace přímo na GPU, volání funkcí (podprogramů), externí API, atd.
- 2011 – OpenGL 4.2 - atomické instrukce, instancování objektů, lokální modifikace komprimované textury atd.

- Podrobněji zpracováno na:

  http://en.wikipedia.org/wiki/OpenGL

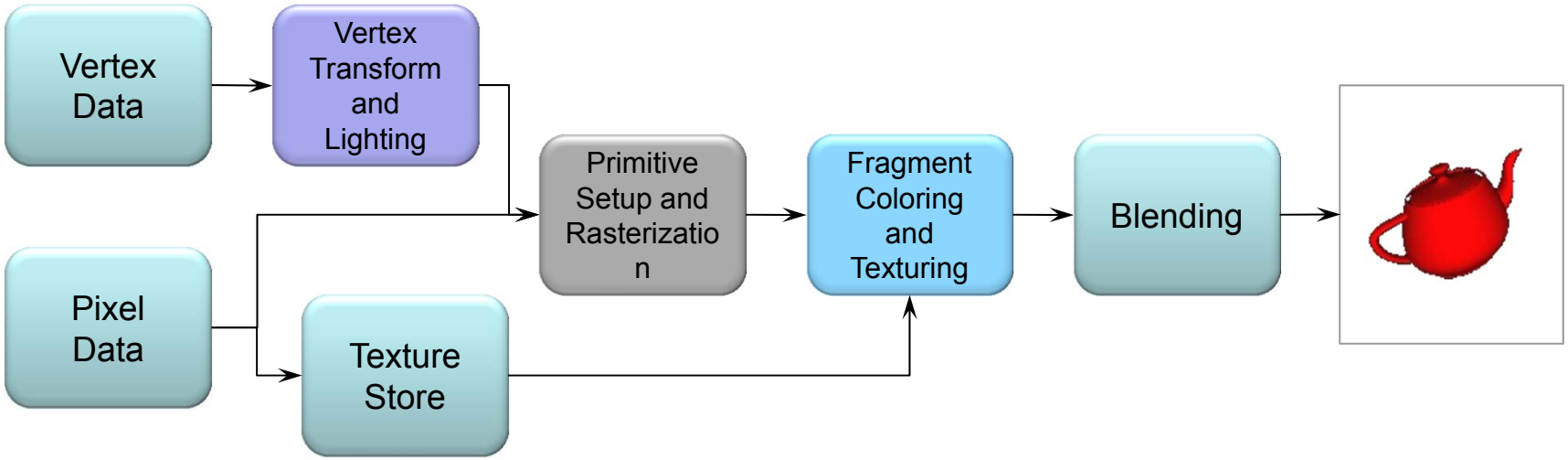
OpenGL 1.0, rok 1994

- Fixní zobrazovací řetězec (“fixed pipeline”), tj. neprogramovatelný
- Operace a jejich pořadí je zadrátované v HW
- Vydržel dlouho, až do roku 2004, po roce 2000 kritizován pro svoji nepružnost

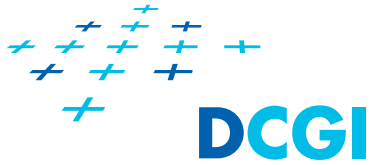


OpenGL 2.0, rok 2004

- Nové programovatelné bloky – **vertex shader** a **fragment shader**
- Fixní proudové zpracování stále programově dostupné



OpenGL 3.0, červenec 2008 – větší změny

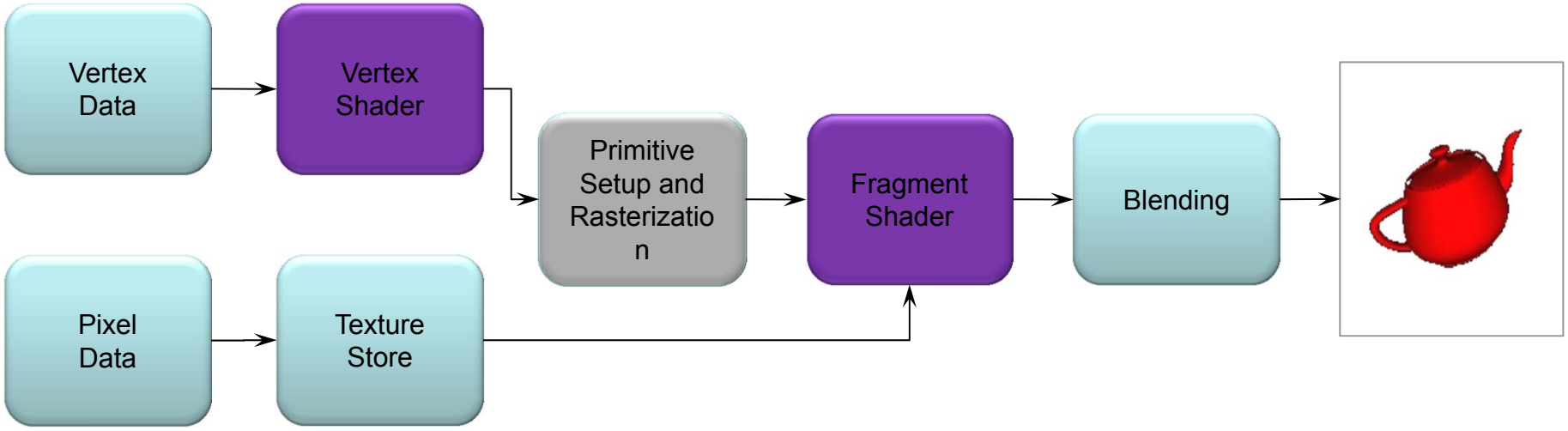


- OpenGL 3.0 zavádí "*deprecation model*"
  - Metoda pro odstranění starých funkcionalit z jazyka OpenGL
  - Zastaralé metody označeny jako zavržené – *deprecated*
  - Programátoři by je neměli používat, neboť budou v dalších verzích OGL z knihovny odstraněny
- Řetězec zpracování stejný jako verze 2.1 až do další verze OpenGL 3.1 (březen, 2009)
- Je definován typ kontextu plný a dopředně kompatibilní

Typ kontextu	Popis
Full	Všechny funkce z aktuální verze OpenGL (včetně zavržených – *deprecated*)
Forward Compatible	Pouze nezavržené funkce (tj. kontext, podobný tomu, jak to bude v dalších verzích OpenGL)

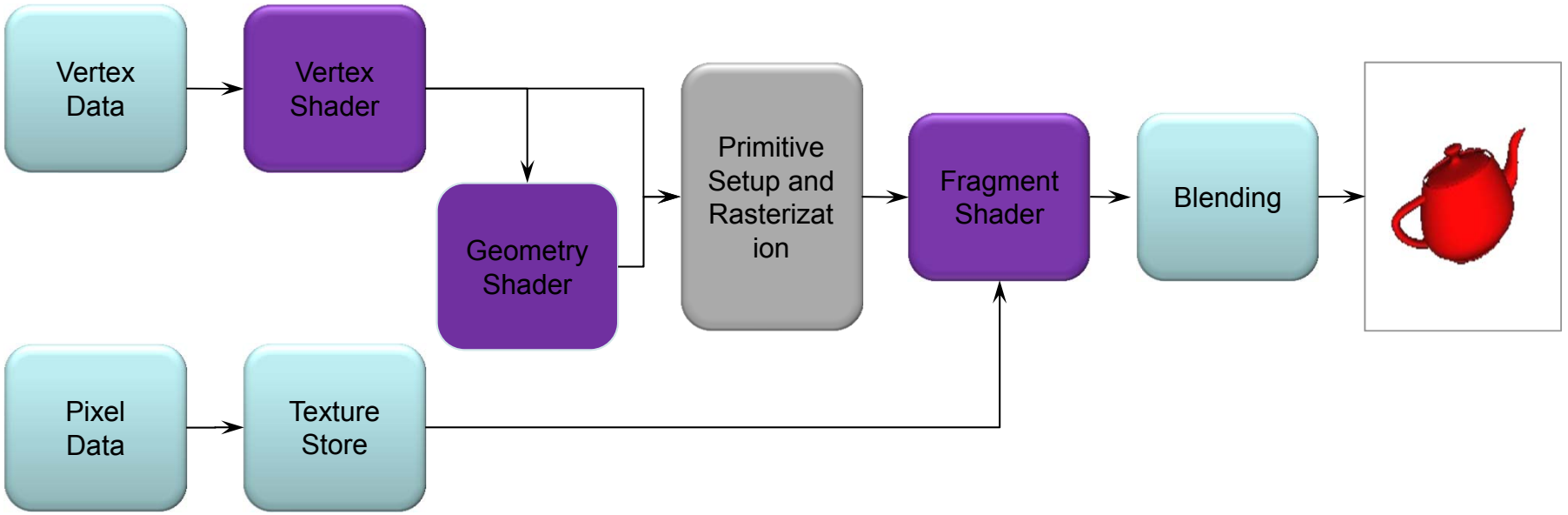
OpenGL 3.1, březen 2009

- OpenGL 3.1 odstranila fixní proudové zpracování
  - Programy musí používat tzv. shadery (*shaders*)
- Maximum dat uloženo v paměti na *GPU*
  - Data vrcholů s použitím “buffer objects” a “vertex arrays”

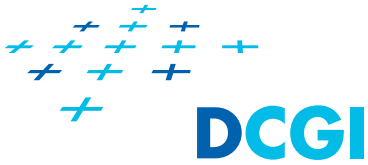


OpenGL 3.2, srpen 2009

- Nově geometrický procesor – “**geometry shader**”
- A profily (*context profiles)*
  - Určují verzi OpenGL a množinu použitelných příkazů
  - Dva typy profilu: “*core”* and “*compatible”*



OpenGL 3.1 a 3.2 – context profiles

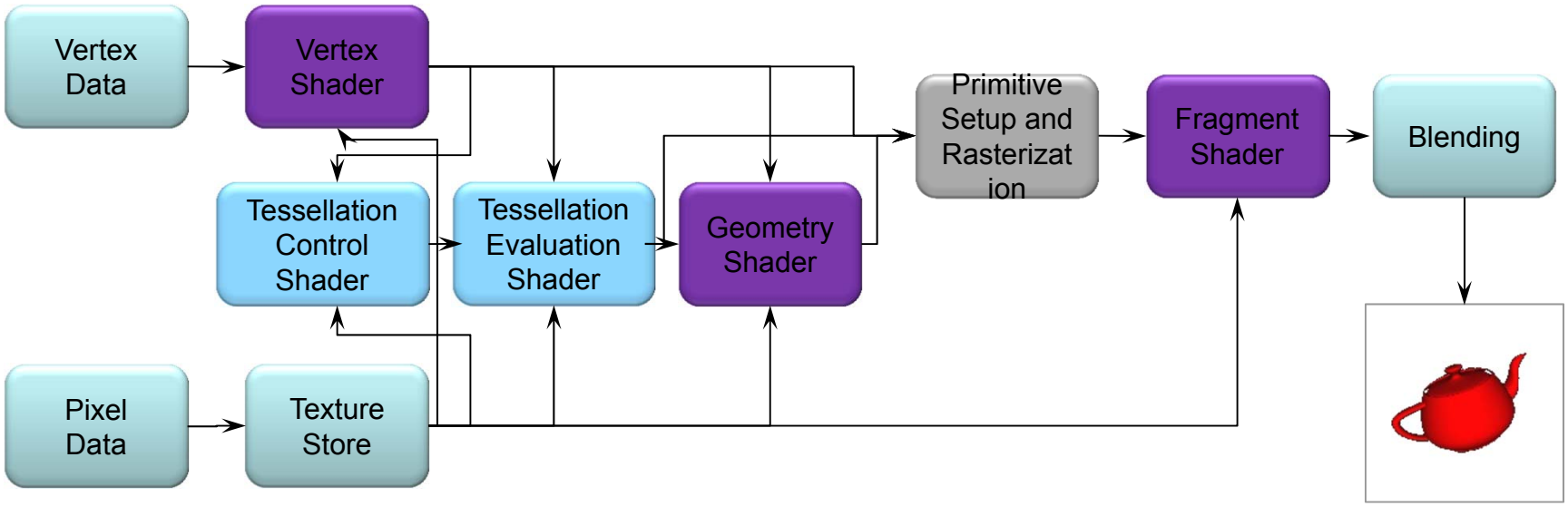


- A profily (*context profiles*)
  - Určují verzi OpenGL a množinu použitelných příkazů
  - Dva typy profilu: "*core*" and "*compatible*"

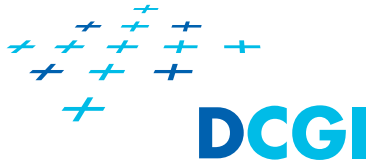
Context Type	Profile	Description
Full	core	Všechny funkce dané verze OpenGL (i ty, označené jako zavržené, ale bez zrušených funkcí)
	compatible	Všechny funkce ze všech předchozích verzí OpenGL (full v OpenGL 3.0)
Forward Compatible	core	Pouze nezavržené funkce dané verze OpenGL
	compatible	Není podporován

OpenGL 4.1, březen 2010

- Další 2 programovatelné bloky – “***tessellation-control***” *a* “***tessellation-evaluation shaders***”



Základy OpenGL



- Syntaxe příkazů OpenGL
- OpenGL jako automat (Stavový stroj – *state machine*)
- Zpracování chyb

Syntaxe příkazů v OpenGL

Konstanty („Constants“)

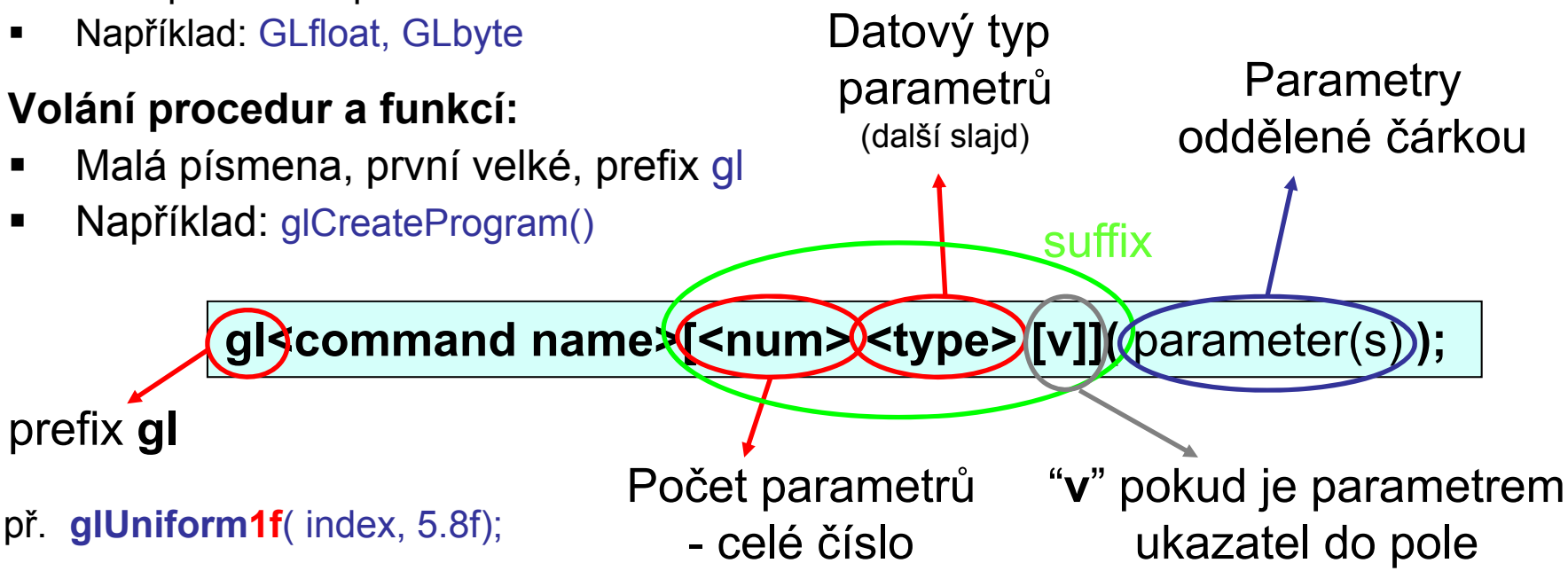
- Velká písmena, prefix GL_
- Například: GL_COLOR_BUFFER_BIT, GL_DEPTH_TEST

Datové typy

- Malá písmena s prefixem GL
- Například: GLfloat, GLbyte

Volání procedur a funkcí:

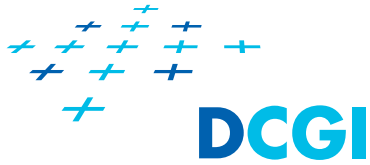
- Malá písmena, první velké, prefix gl
- Například: glCreateProgram()



př. **glUniform1f**(index, 5.8f);

```
double attrib_array[3] = {1.0, 5.0, 6.0};
glUniform3fv (index, attrib_array);
```

Datové typy parametrů příkazů

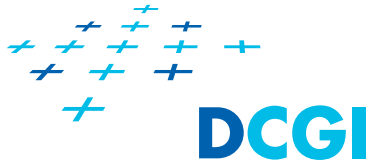


Suffix	Datový typ	Datový typ v ANSI C	OpenGL definice
b	8bit integer	signed char	GLbyte
s	16bit integer	short	GLshort
i	32bit integer	long	GLint, GLsizei
i64	64bit integer	(long long, _int64)	GLint64
f	32bit float	float	GLfloat, GLclampf
d	64bit float	double	GLdouble, GLclampd
ub	8bit unsigned int	unsigned char	GLubyte, GLboolean
us	16bit unsigned int	unsigned short	GLushort
ui	32bit unsigned int	unsigned long	GLunit, GLenum, GLbitfield
ui64	64bit unsigned int		GLunit64

OpenGL jako automat *(state machine)*

- OpenGL je automat – v každém okamžiku má definován svůj vnitřní stav
  - Vnitřní stav je definován hodnotami vnitřních proměnných
  - Pokud je nezměníme, mají své iniciální hodnoty
  - Tyto proměnné lze nastavovat a ptát se na jejich hodnotu
  - Např. zapnutí hloubkového testu

Nastavení a získání hodnoty stavové proměnné



- **Nastavení stavu – logická hodnota (on / off)**

```
glEnable(GL_MULTISAMPLE);
glDisable(GL_MULTISAMPLE);
```

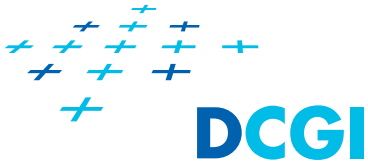
- **Získání stavu**

```
bool glIsEnabled(GL_CULL_FACE);
```

- **Získání stavu jedné či více hodnot:**

```
glGetBooleanv(GLenum  pname,  GLboolean * params);
glGetFloatv(  GLenum  pname,  GLfloat *   params);
glGetIntegerv(GLenum  pname,  GLint *     params);
glGetDoublev( GLenum  pname,  GLdouble *  params);
```

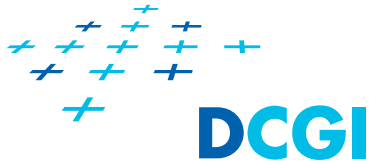
Zpracování chyb v OpenGL



GLenum **glGetError(**void**);**

- OpenGL detekuje jen minimum z možných chyb
- Kontrola všeho by byla příliš časově náročná
- Zapamatuje si první chybu, která nastala
- Každá chyba má svůj vlastní číselný kód a jemu přiřazenou symbolickou konstantu:
  - **GL_NO_ERROR** *no problem, OK*
  - **GL_INVALID_ENUM** *enum value out of range.*
  - **GL_INVALID_VALUE** *numeric argument is out of range*
  - **GL_INVALID_OPERATION** *illegal operation in current state*
  - **GL_INVALID_FRAMEBUFFER_OPERATION** *offending command for current state*
  - **GL_OUT_OF_MEMORY** *not enough memory to execute the command*

OpenGL odkazy



- http://www.opengl.org/
  dokumentace, odkazy na programy,…
- http://www.khronos.org/
  vývoj, schvalování implementací (OpenGL compliant)
- http://www.sgi.com/
  licence pro HW

- Ondřej Karlík - Corona renderer: Making of production-ready renderer in 3dsMAX, Přednáška, **Speciální seminář z počítačové grafiky,** 20. 10. 2011