



# 3D Intro

- 2D nebo 3D?

Pacman (1980)



Wolfenstein 3D (1992)



# Sprite



Doom 1 (1993)

# 2D vs. 3D

Duke Nukem 3D (1996)



Descent (1995)



[https://thumbs.gifycat.com/HardtofindNeatChuckwalla-size\\_restricted.gif](https://thumbs.gifycat.com/HardtofindNeatChuckwalla-size_restricted.gif)

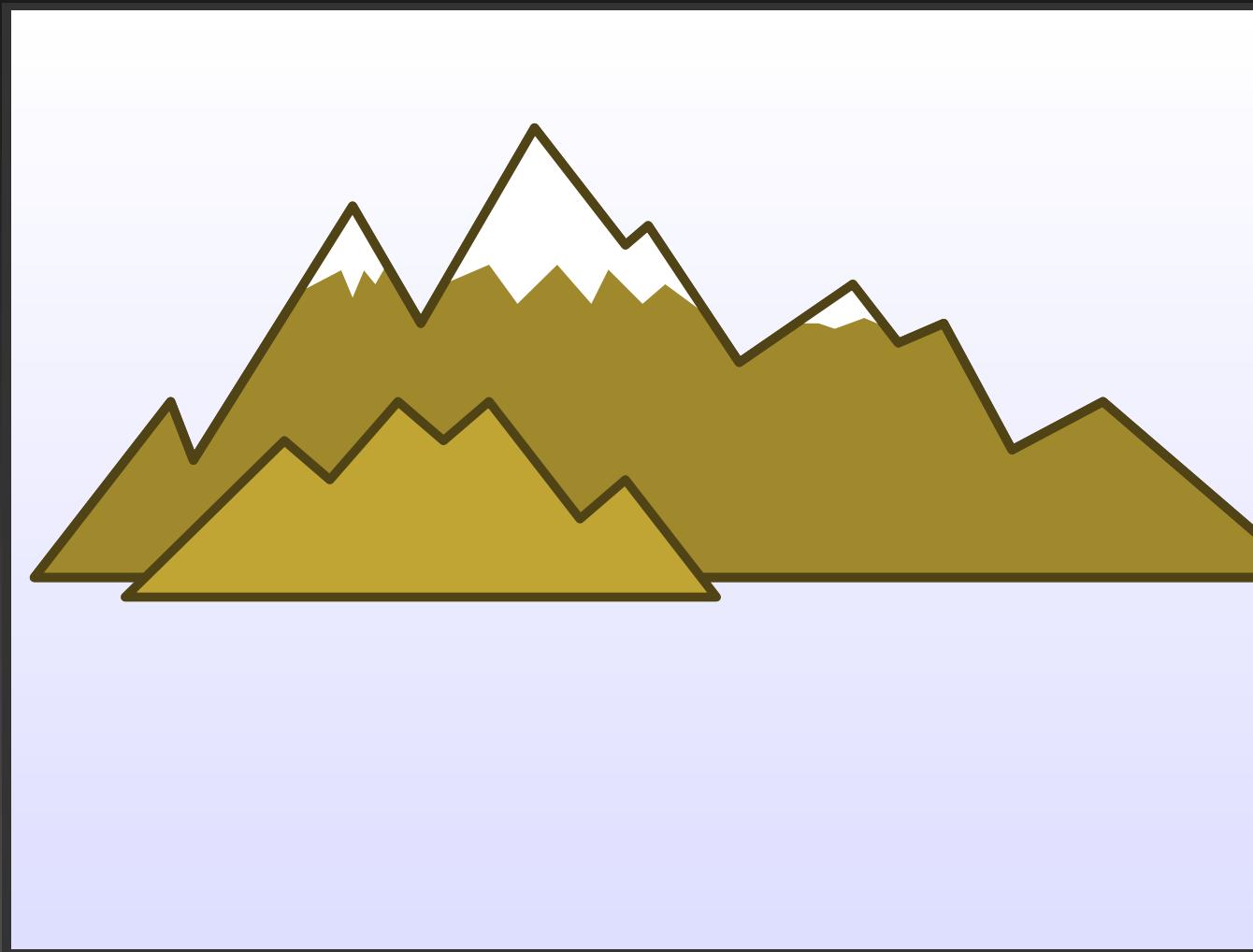


<https://steamuserimages-a.akamaihd.net/ugc/203053761726302995/50811AF808D461C5120DB83662F735E981440A0D/>

<https://media.giphy.com/media/CzTu1z70W0tfa/source.gif>

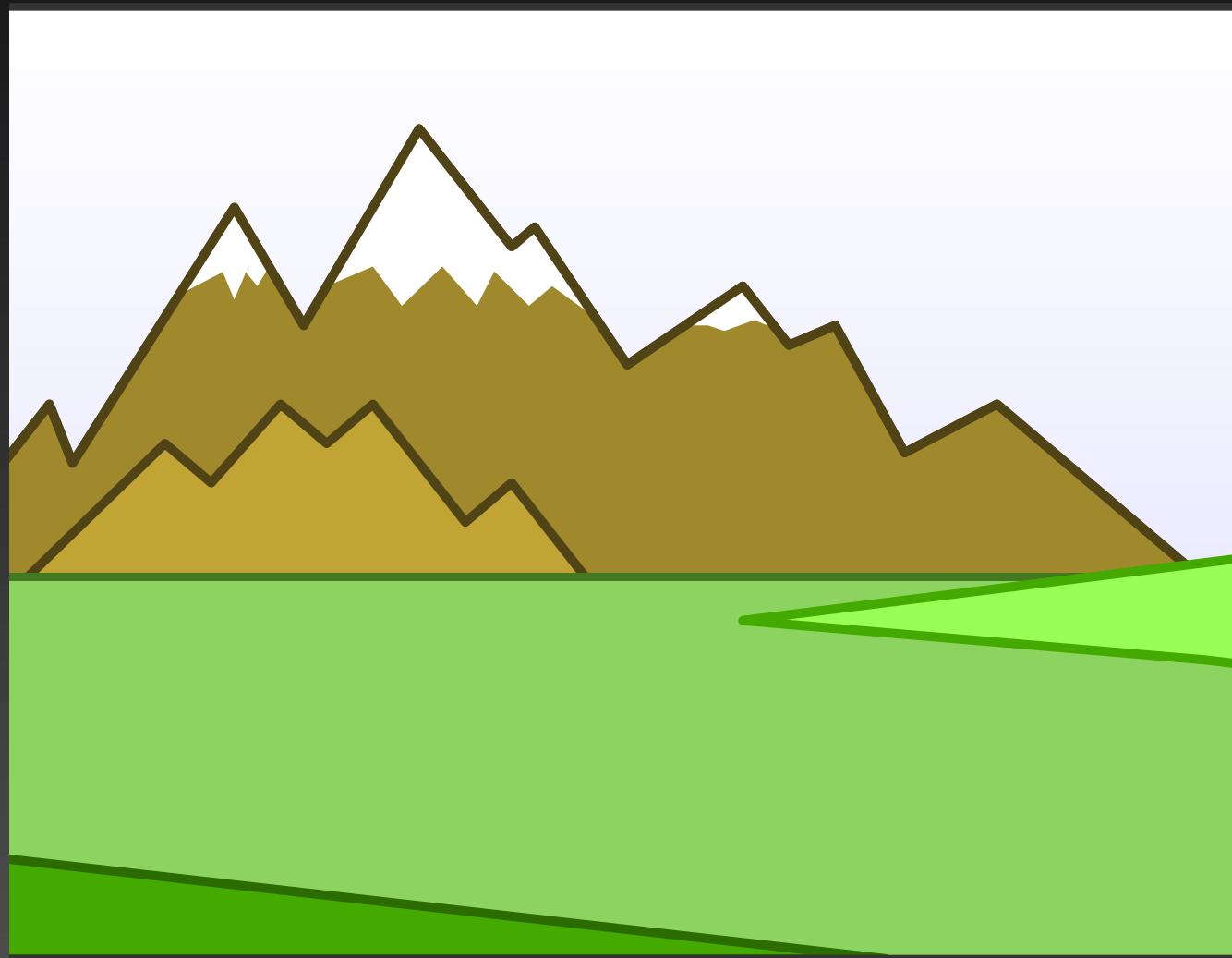
# Malířův algoritmus

- pro pseudo 3D



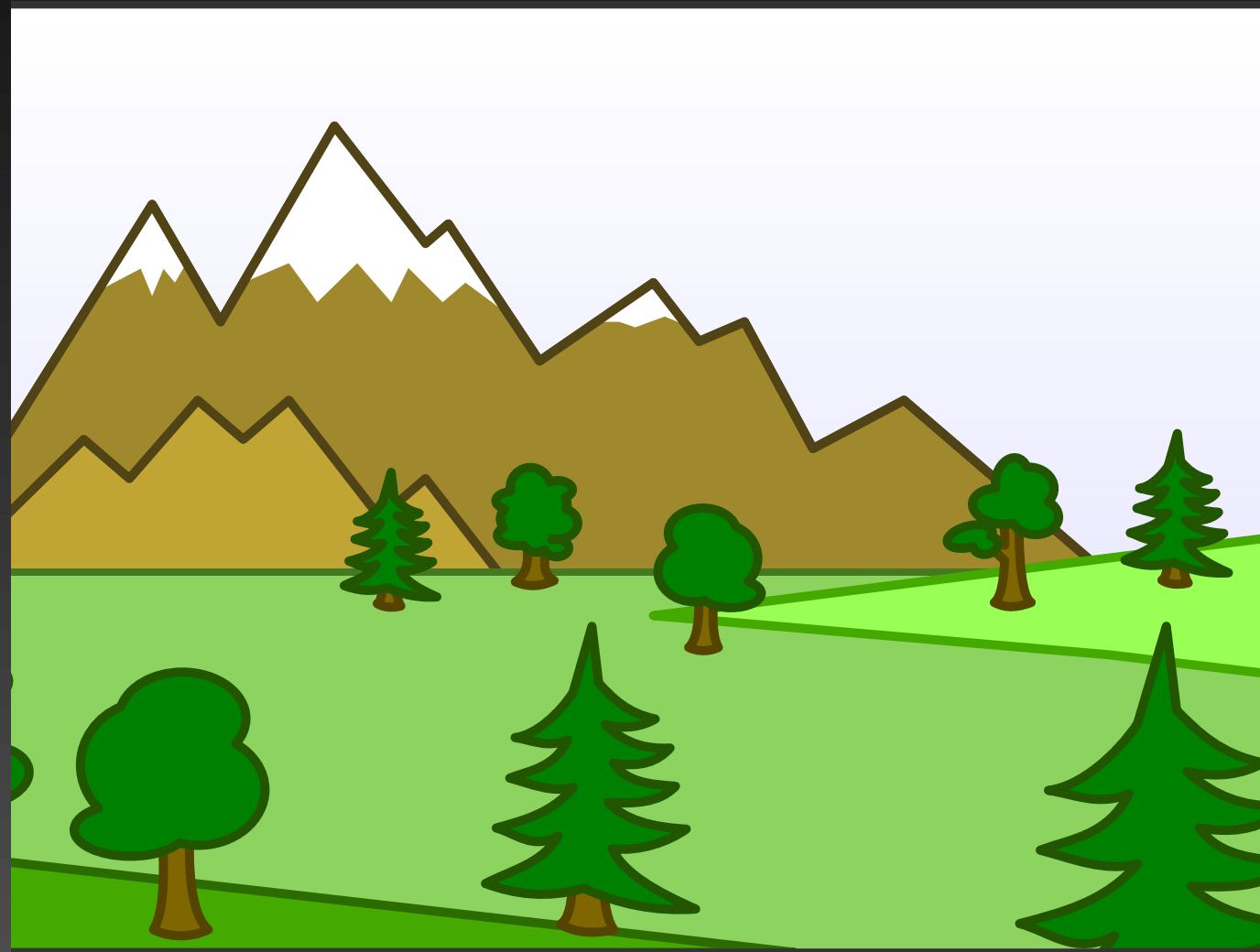
# Malířův algoritmus

- pro pseudo 3D



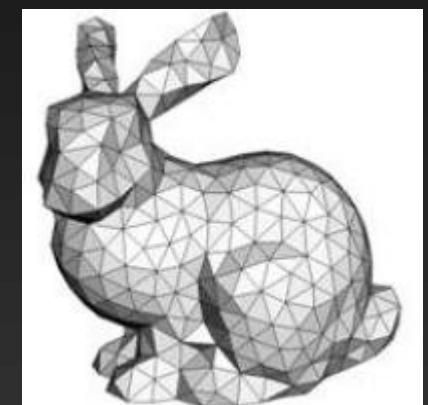
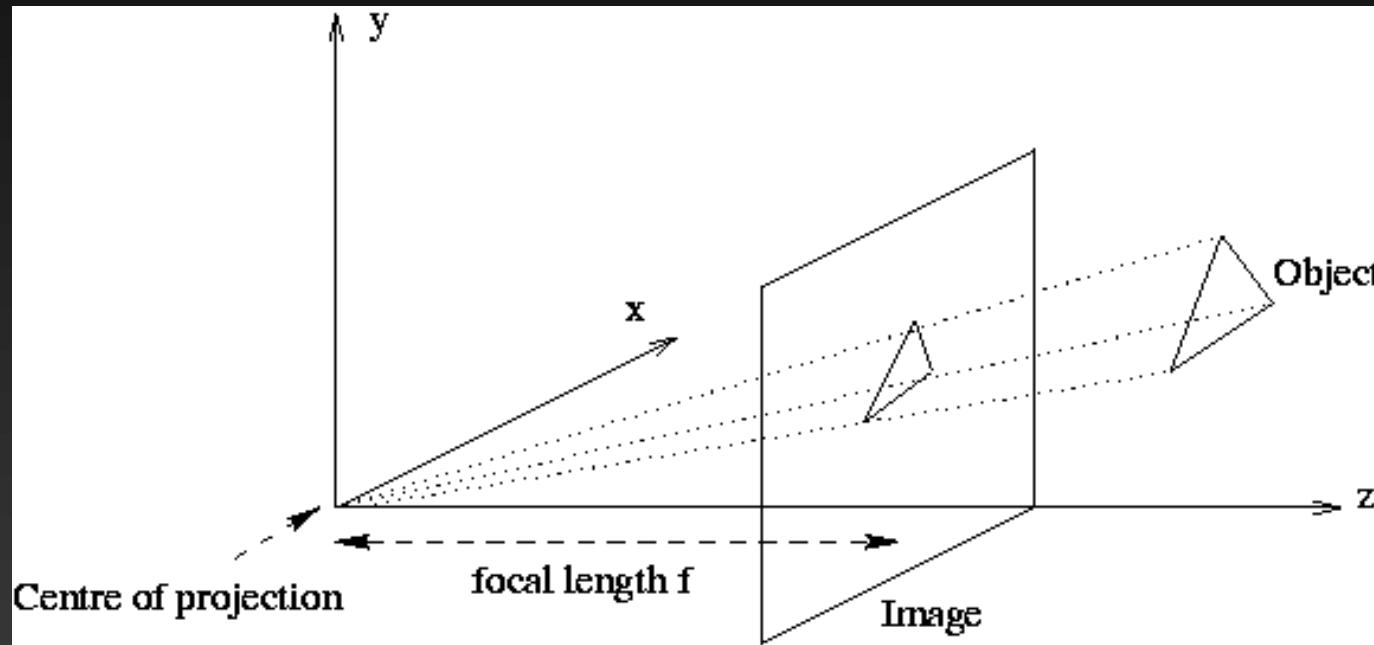
# Malířův algoritmus

- pro pseudo 3D



# 3D

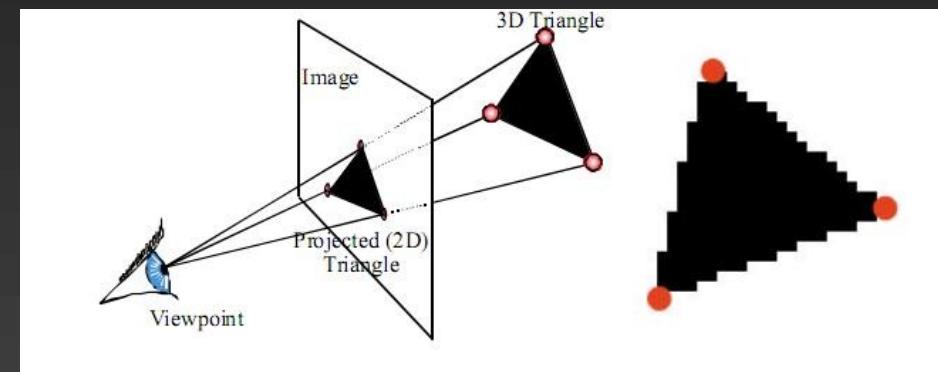
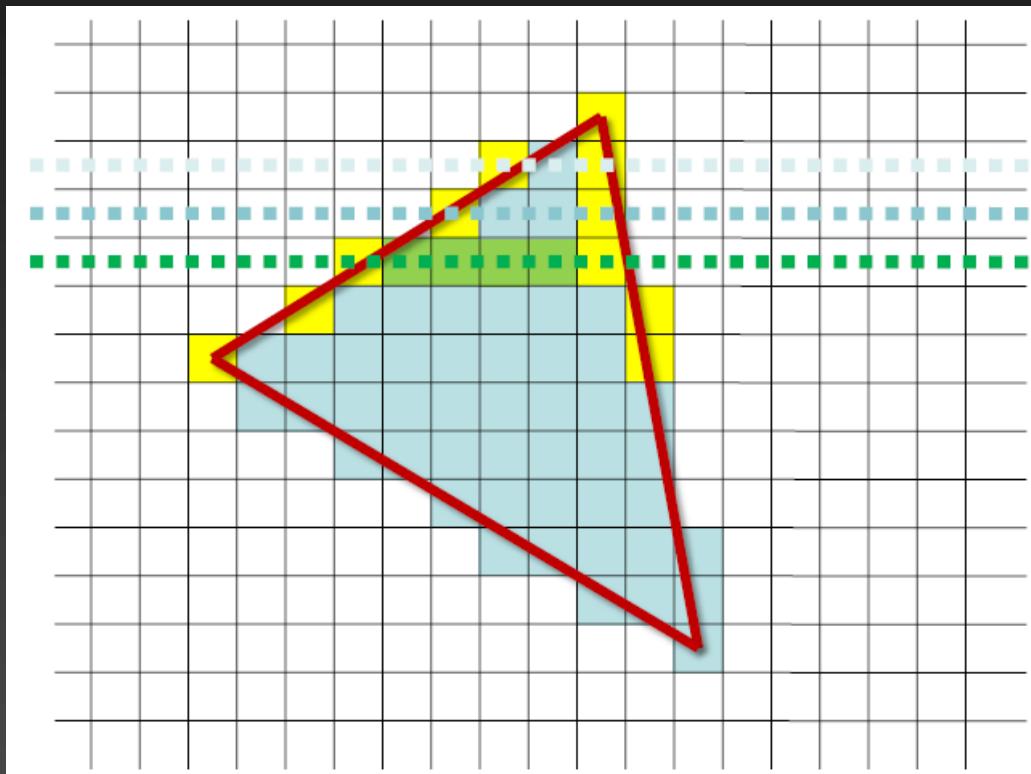
- Sít' trojúhelníků, projekce 3D → 2D, ...



$$x' = \mathbf{M}x \quad \begin{bmatrix} x' \\ y' \\ 0 \\ w' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

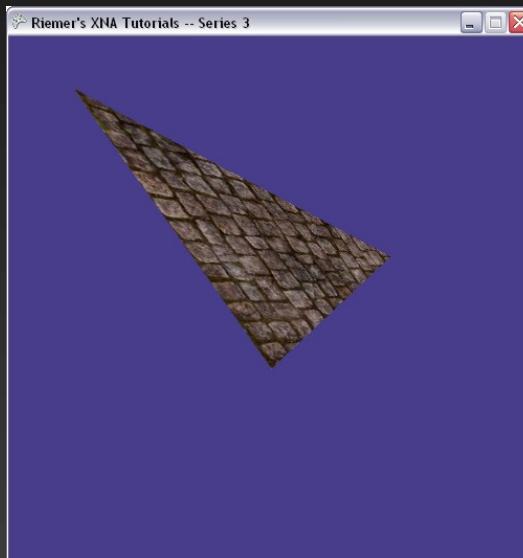
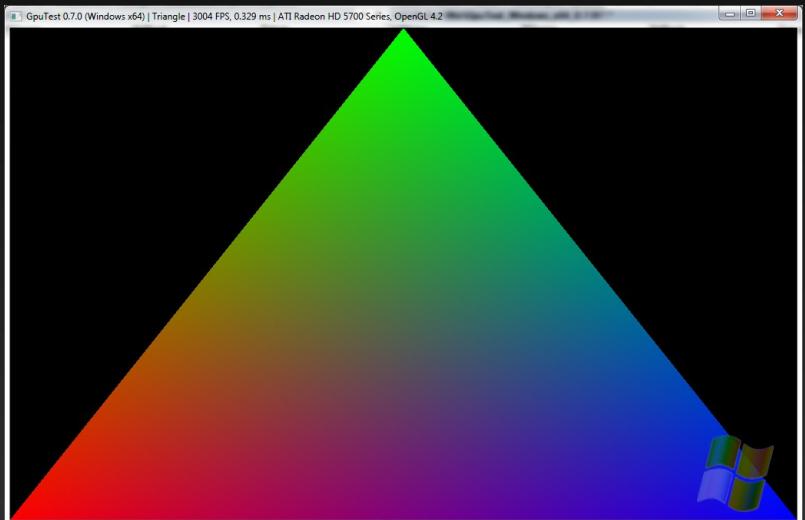
# 3D

- Rasterizace – např. scanline algoritmus
  - interpolace souřadnic



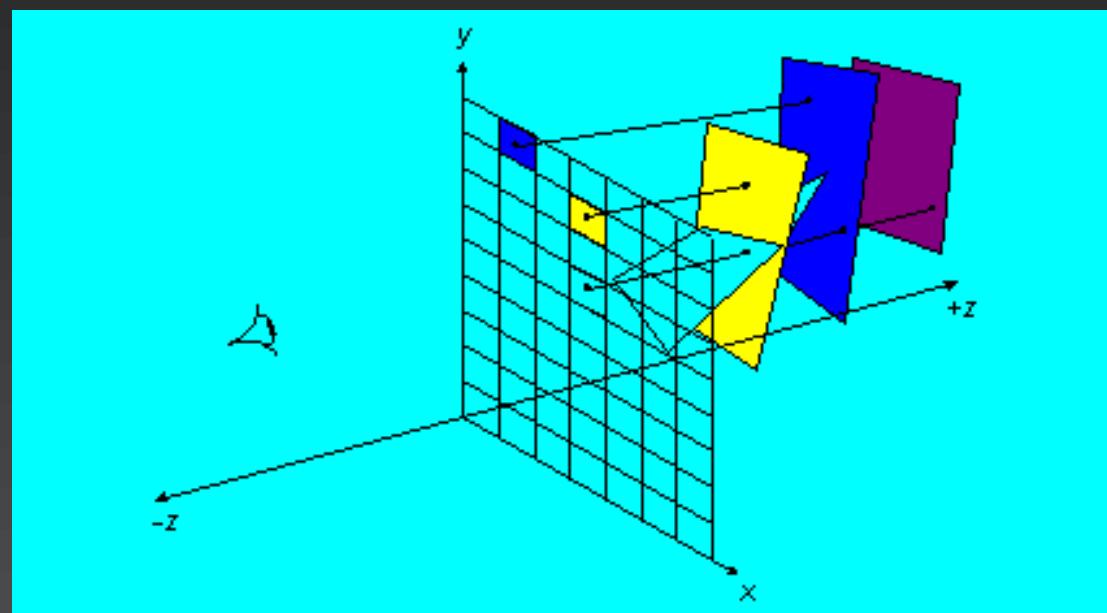
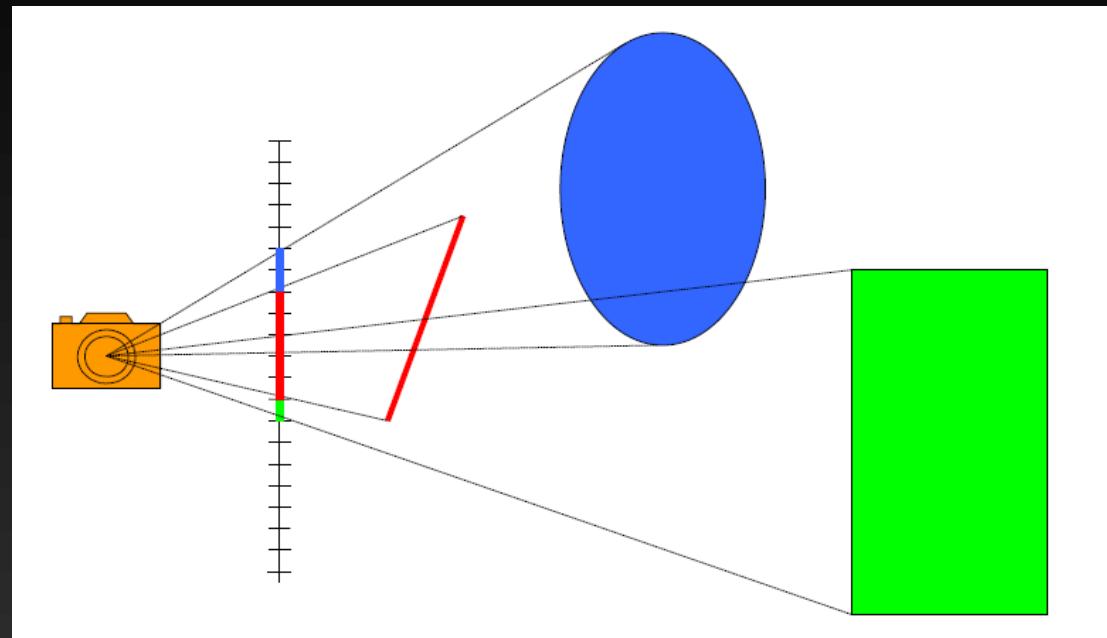
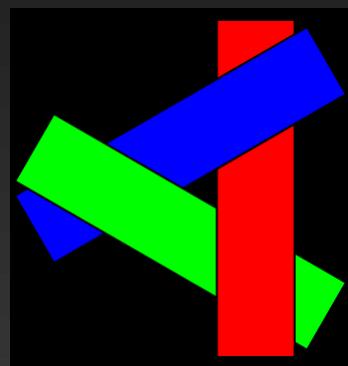
# 3D

- Obarvení, textury, ...



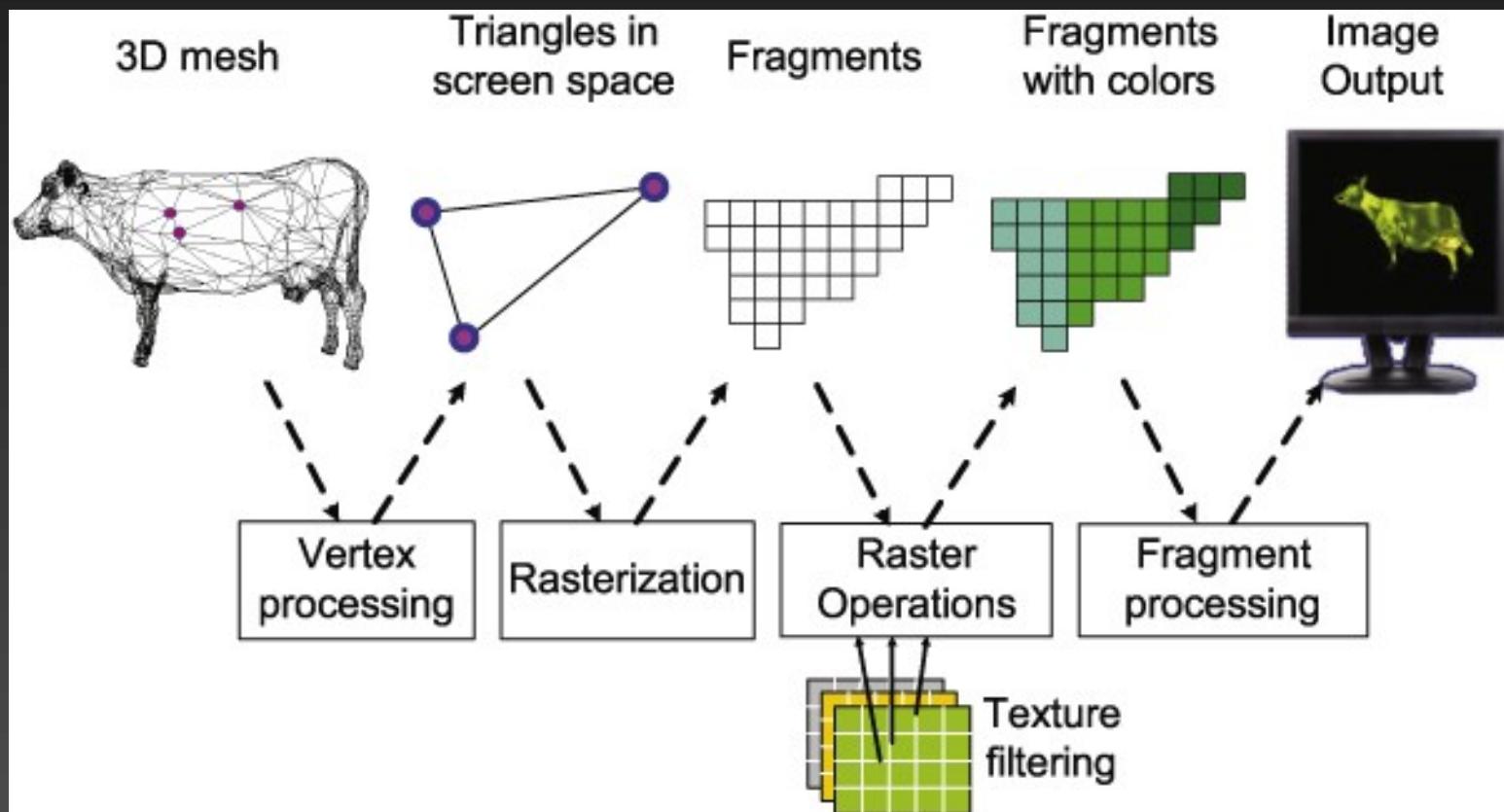
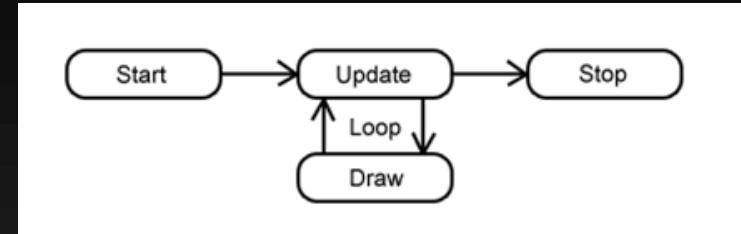
# 3D

- Paměť hloubky ...



# Soudobá real-time 3D grafika

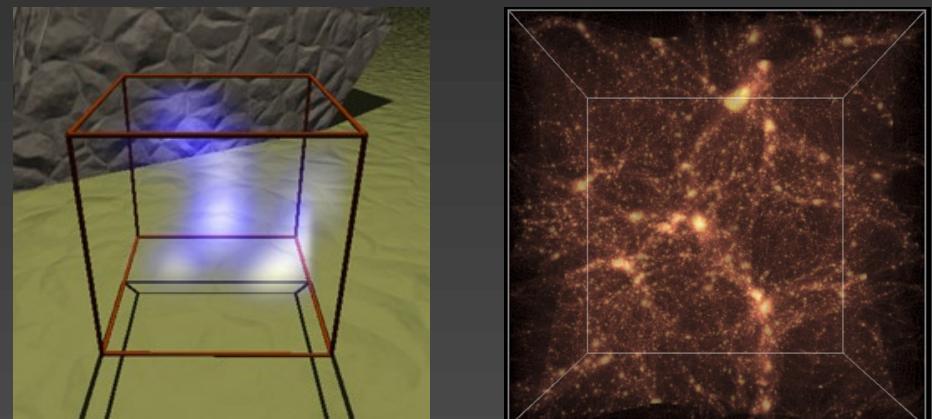
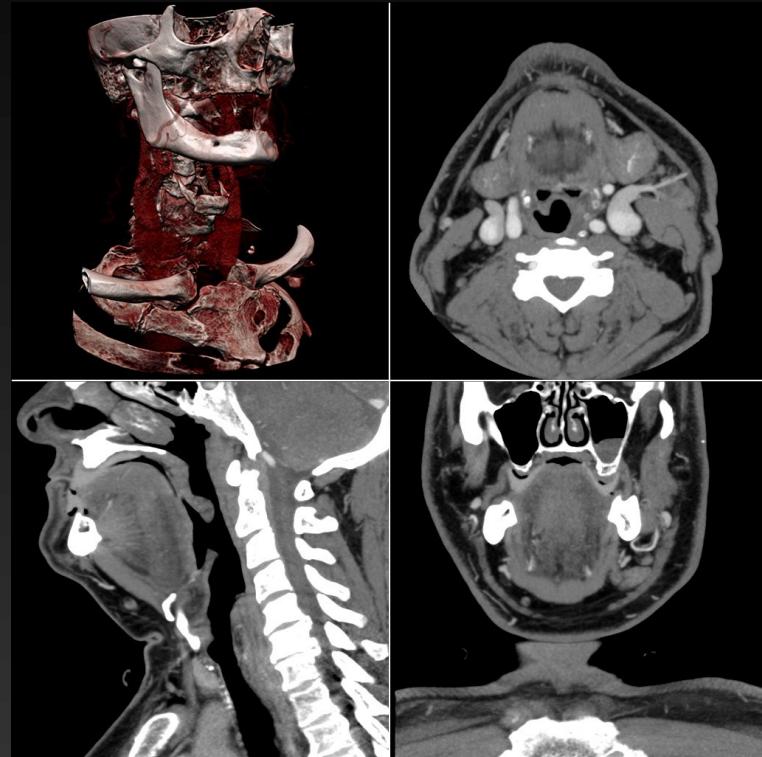
- 3D vektorový model z trojúhelníků
- transformace a projekce 3D → 2D
- rasterizace
- obarvení fragmentu
- řešení hloubky a průhlednosti pro každý fragment



# 3D reprezentace

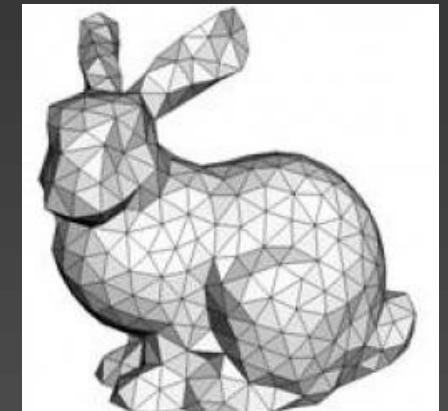
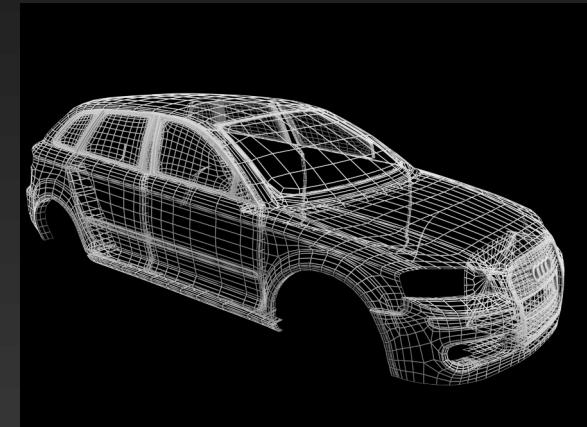
# Objemová reprezentace (= 3D raster)

- Základní jednotka
  - voxel = volumetric pixel
- Velký objem dat
  - 1024x1024x1024xRGBA = 4 GiB
- Speciální účely
  - magnetická rezonance
  - fyzikální simulace
    - mlha, kouř, kapaliny...
- Sada 2D obrázků nebo vlastní formáty



# Hraniční reprezentace

- 3D vektorový popis obálky (tj. hranice, slupky) objektu
  - Polygonální popis
    - vertexy (tj. body)
    - hrany
    - plošky + normálové vektory
  - Křivky (NURBS, spline, ...)
  - CSG – Constructive Solid Geometry
  - Implicitní plochy



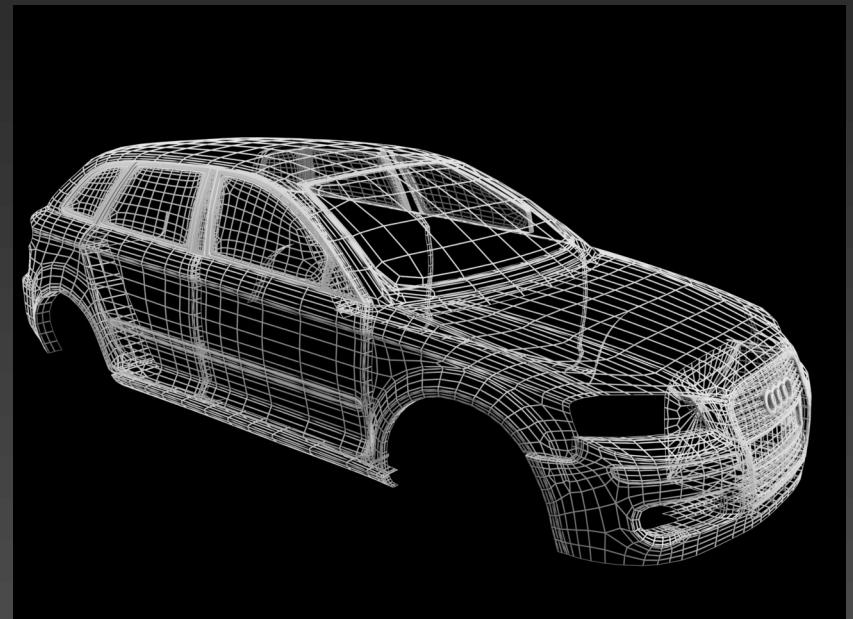
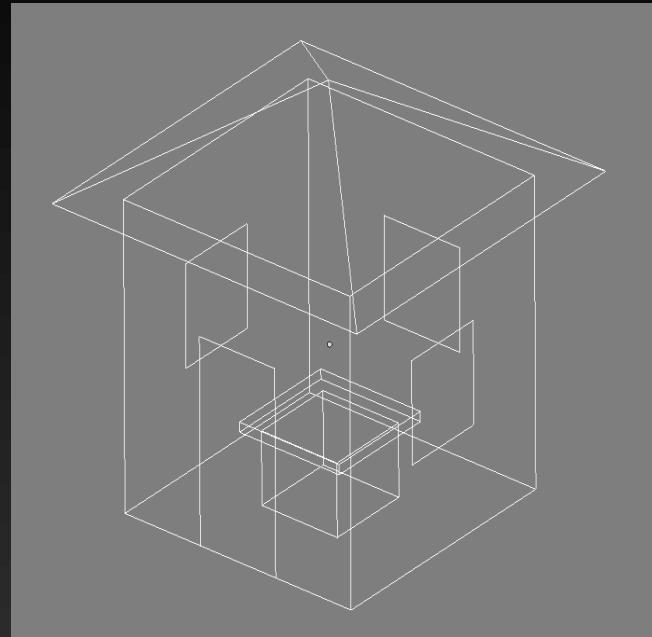
# Hraniční reprezentace pomocí bodů (vertexů)

- Pouze vertexy
- 3D laserové skenování
  - bod má pozici, barvu, normálu, ...
- Velké objemy dat
  - menší než objemová reprezentace (popisujeme jen obálku, ne celý objem)



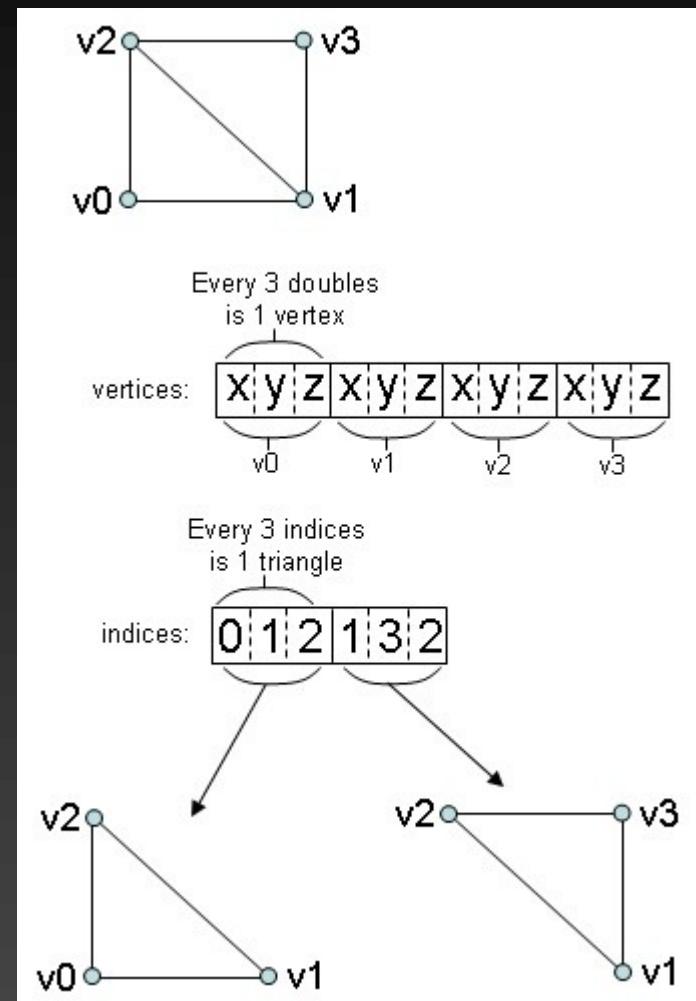
# Hraniční reprezentace pomocí hran

- Pouze vertexy a jejich spojení - hrany
- Jednoduché na vykreslení
  - není třeba řešit viditelnost a vyplnění ploch
- ... zároveň obtížné
  - ... pokud máme mnoho hran
  - všechno je vždy vidět, nic nelze vynechat



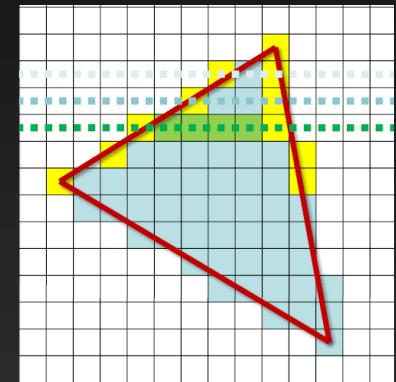
# Hraniční reprezentace pomocí plošek

- Topologie – obsahuje informace o ploškách
- Trojúhelníková síť
  - pole trojic indexů do dalšího pole s vertexy
- Polygonální síť
  - pole n-tic indexů různé délky
  - šetří paměť, komplikuje vykreslení



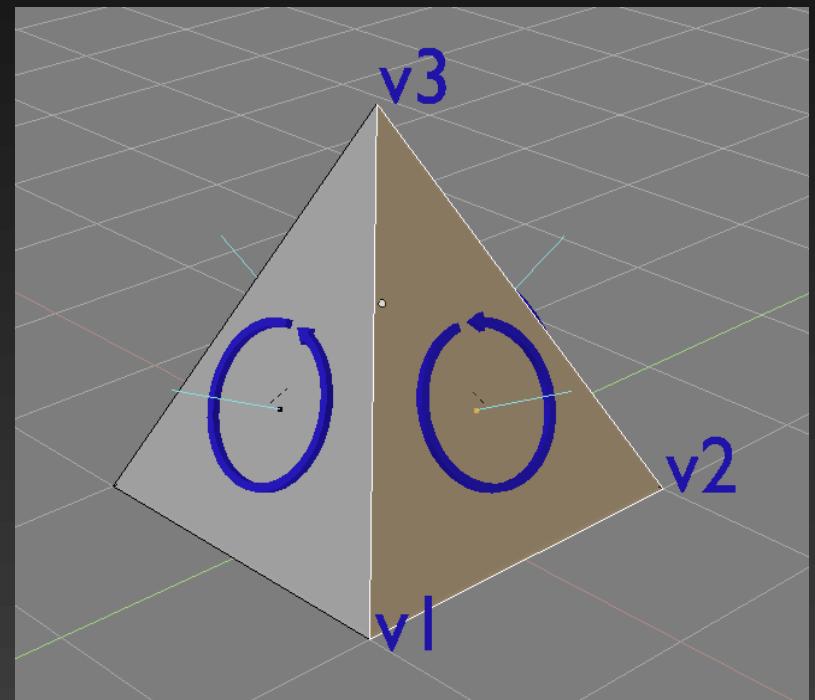
# Trojúhelníková síť

- Trojúhelník
  - všechny body vždy v rovině (definuje rovinu)
  - vždy konvexní
  - jednoduché na rasterizaci
    - HW akcelerace
- Trojúhelníková síť
  - geometrie ... kde jsou vertexy?
  - topologie ... jak jsou spojené?



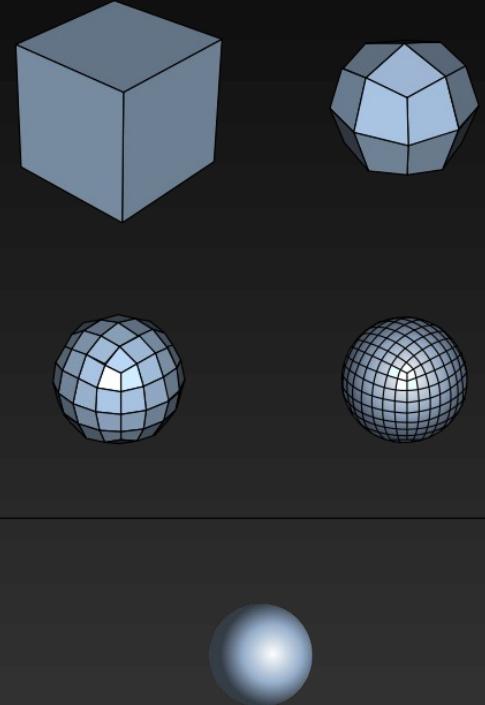
# Plošky

- Přední strana určena směrem normály
  - dáno pořadím zadávání vertexů
  - CCW – Counter Clock Wise
    - obvyklé
    - pravidlo pravé ruky
  - CW – Clock Wise
    - možné, ale nepoužívá se

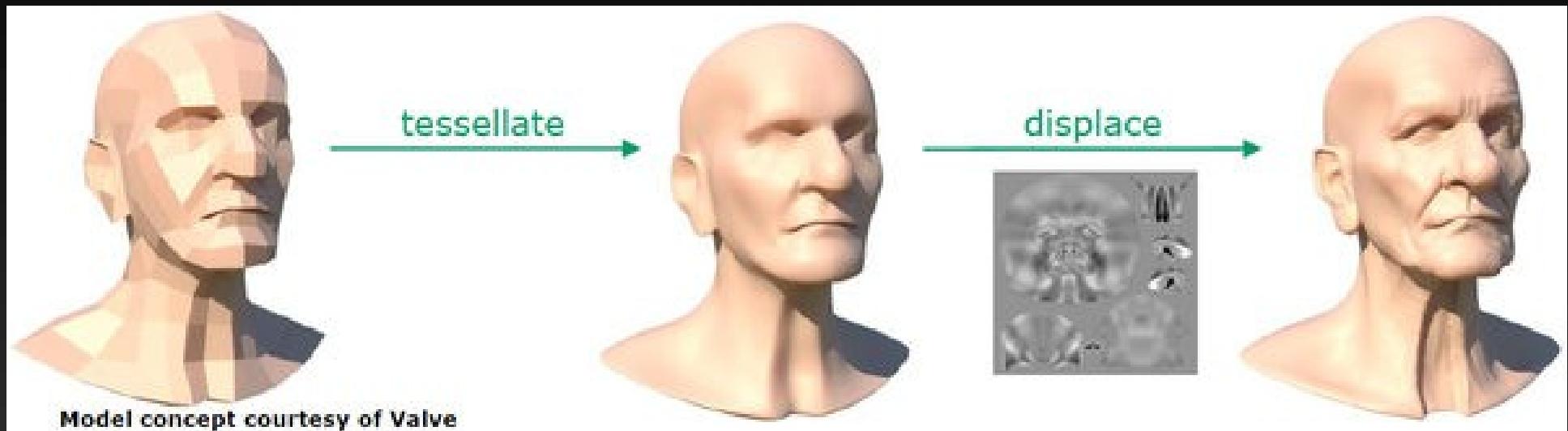


# Dělení plošek - teselace

- Několik způsobů
  - Aproximace vs. interpolace
- Dělení nad vertexy
  - Doo-Sabin
- Dělení nad ploškami
  - Catmull-Clark
    - OpenGL + DirectX 11 teselace
  - pracuje s čtyřúhelníkovou sítí
  - náročnější než Doo-Sabin
  - garantuje C1 spojitost (první derivace)



# Typické použití



60  
triangles

600  
triangles

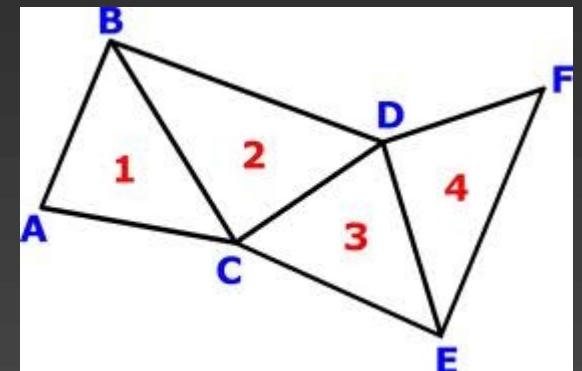
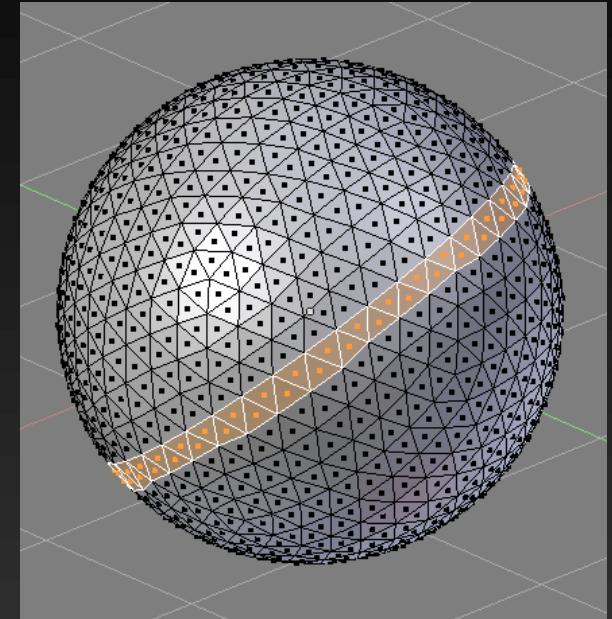
6000  
triangles

60000  
triangles

Diminishing returns - 15 years ago even doubling the amount of triangles resulted in a much better mesh. Now, multiplying the amount by 10 hardly does.

# Optimalizace trojúhelníkové sítě

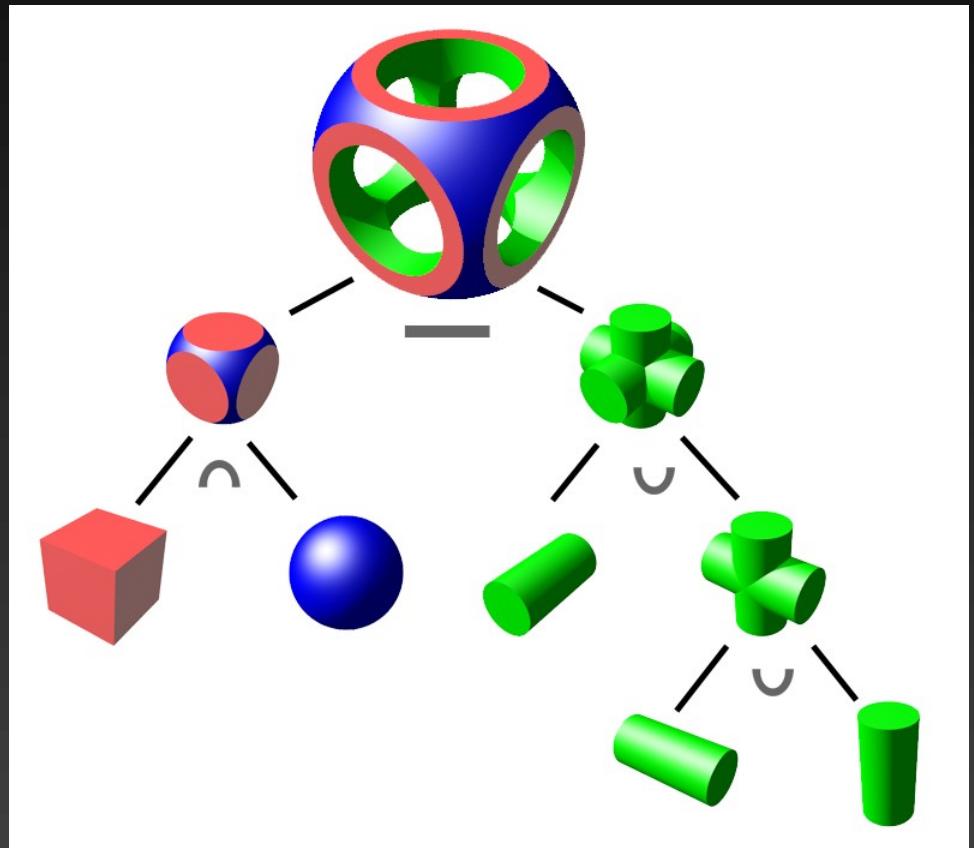
- Pás trojúhelníků (vějíř)
  - přenáší se jen nezbytné informace
  - GPU má HW implementaci → rychlé
  - OpenGL, Direct3D
    - `GL_TRIANGLE_STRIP`
    - `GL_TRIANGLE_FAN`



# CSG

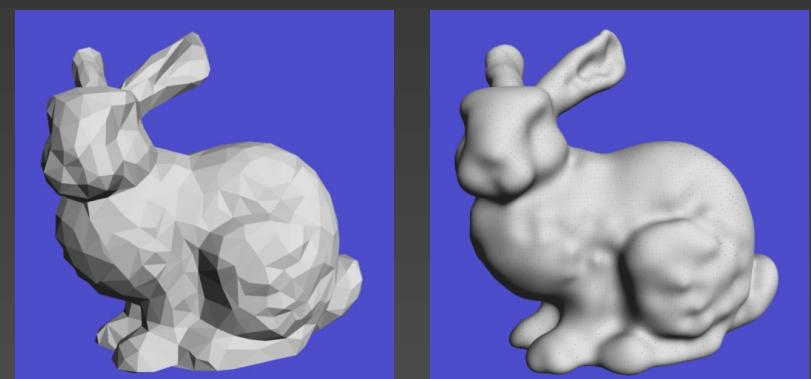
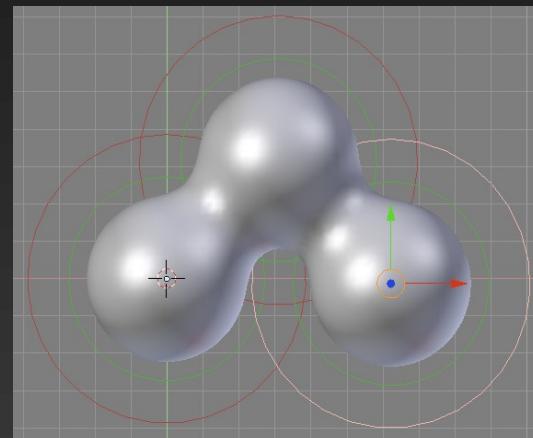
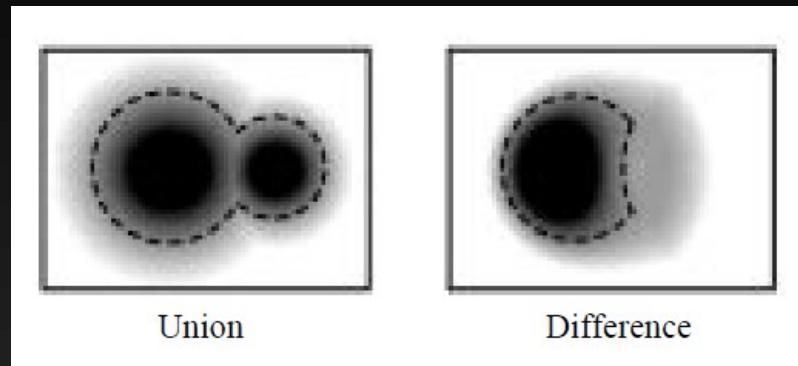
## Konstruktivní geometrie

- Výsledek vytvořen pomocí primitiv a booleovských operací
  - krychle, koule, válec, kužel, ...
- Před zobrazením obvykle převedeno na trojúhelníkovou síť
- CAD apikace, fyzika (detekce kolizí)



# Implicitní plochy

- Obálka vytvořena z primitiv a booleovských operací jako zobrazení pole
  - bod → koule
  - úsečka → cca válec
  - čtverec → cca kvádr
  - ...
- Vytváří izoplochu
  - ... a převede na trojúhelníky



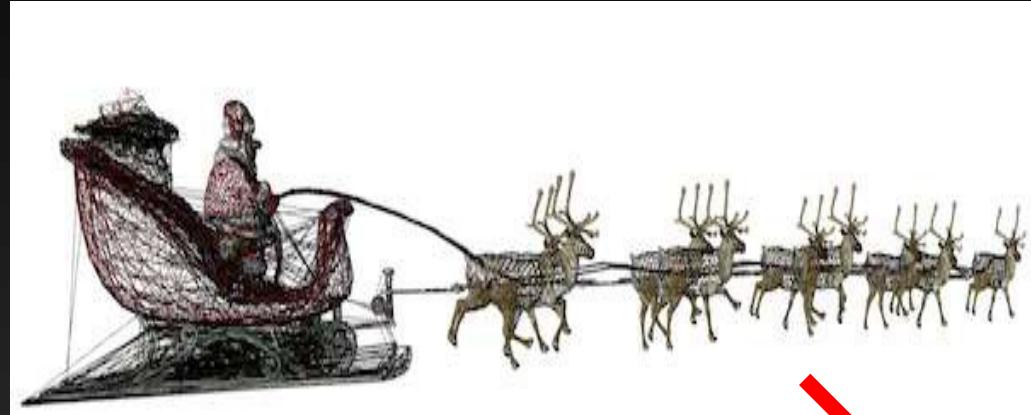
# 3D obrazové formáty

# Ukládání 3D

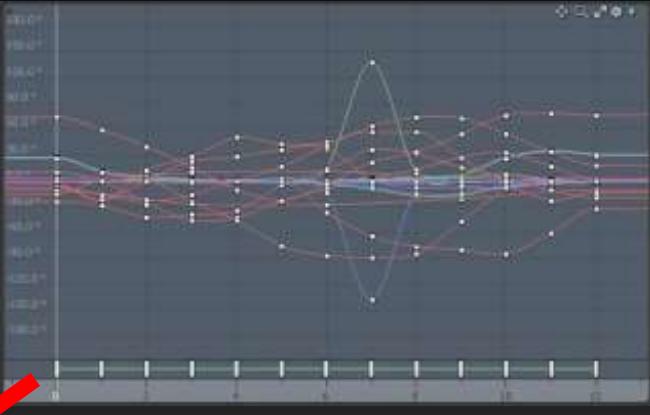
- 3D rastr
  - sada 2D rastrových obrázků
- Hraniční reprezentace
  - popis vertexů, hran a trojúhelníkové sítě
    - Přílišná variabilita dat
    - nedostatek rozumných standardů



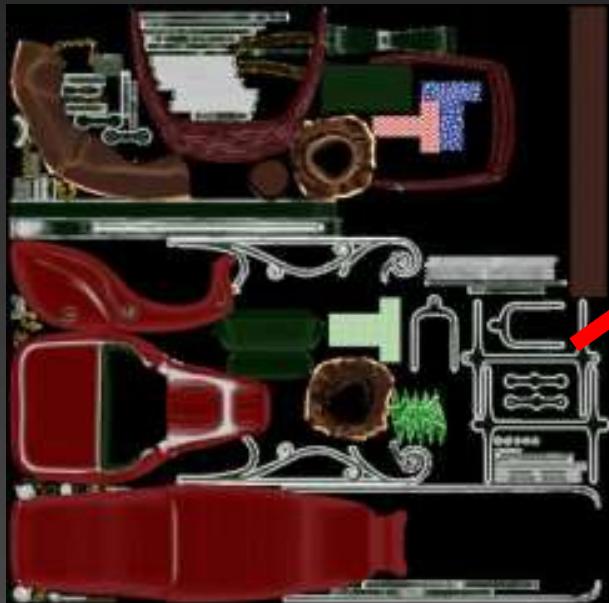
# Co potřebujeme pro 3D model (asset)?



Hierarchie scény a geometrie



Animace



Materiály a textury

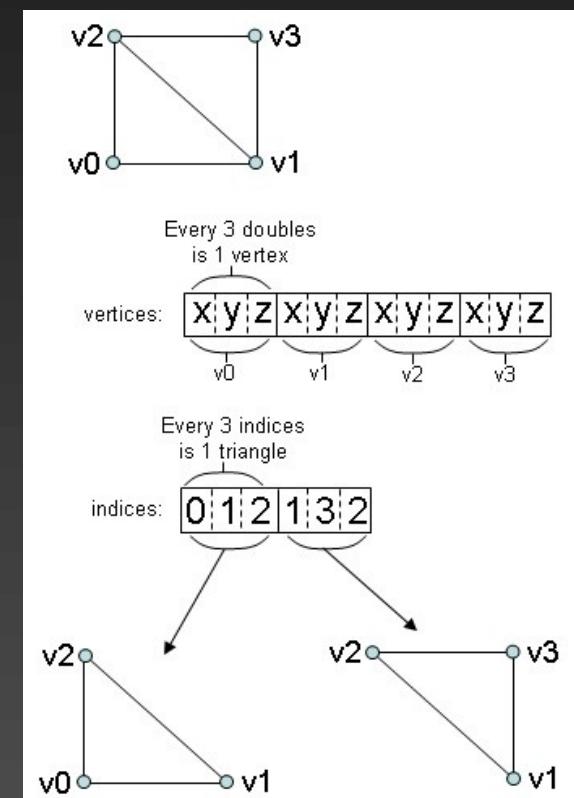
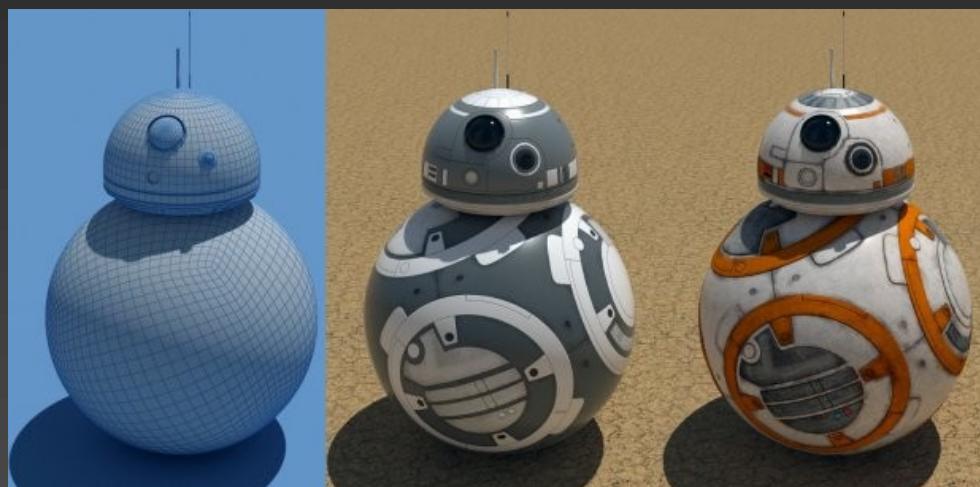


Výsledný asset ve scéně

a další...

# Wavefront Object .OBJ StereoLithography .STL

- Jednoduchý textový formát, běžný, jen základní vlastnosti
- Geometrie
  - vertexy, normály, texturovací koordináty
  - CCW trojúhelníky pomocí indexovaných vertexů
- Materiály i textury uloženy externě
  - .MTL = Material Template Library
  - .TGA = Targa rastrový formát pro textury



# Příklad Wavefront OBJ

## model.OBJ

```
# comment...

# object name
o cube01

# vertex list
v -0.500000 -0.500000 0.500000
v 0.500000 -0.500000 0.500000
v -0.500000 0.500000 0.500000
v 0.500000 0.500000 0.500000
v -0.500000 0.500000 -0.500000
v 0.500000 0.500000 -0.500000
v -0.500000 -0.500000 -0.500000
v 0.500000 -0.500000 -0.500000

# texture coordinate list
vt 0.000000 0.000000
vt 1.000000 0.000000
vt 0.000000 1.000000
vt 1.000000 1.000000

# normal vector list
vn 0.000000 0.000000 1.000000
vn 0.000000 1.000000 0.000000
vn 0.000000 0.000000 -1.000000
vn 0.000000 -1.000000 0.000000
vn 1.000000 0.000000 0.000000
vn -1.000000 0.000000 0.000000

# faces (v/vt/vn)
f 1/1/1 2/2/1 3/3/1
f 3/3/1 2/2/1 4/4/1
f 3/1/2 4/2/2 5/3/2
f 5/3/2 4/2/2 6/4/2
f 5/4/3 6/3/3 7/2/3
f 7/2/3 6/3/3 8/1/3
f 7/1/4 8/2/4 1/3/4
f 1/3/4 8/2/4 2/4/4
f 2/1/5 8/2/5 4/3/5
f 4/3/5 8/2/5 6/4/5
f 7/1/6 1/2/6 5/3/6
f 5/3/6 1/2/6 3/4/6
```

## model.MTL

```
# define material
newmtl FrontColor_snowwhite

# ambient color
Ka 0.000000 0.000000 0.000000
# diffuse color
Kd 1.000000 1.000000 1.000000
# specular color
Ks 0.330000 0.330000 0.330000
# specular reflectivity
Ns 10.000

# the ambient texture
map_Ka texture.tga

# the diffuse texture
map_Kd texture.tga

# specular map (texture)
map_Ks specular.tga
```

# VRML, X3D

- Virtual Reality Modeling Language
  - zastarálý, ale normovaný formát
  - vhodné pro import-export
- Popis 3D s aktivními i pasivními objekty
  - dynamika pomocí JavaScriptu
- ISO standard
- Textový formát
- Pro web vyžaduje plugin, nepoužívané
- VRML nahrazeno X3D (lepší integrace s XML)

# FilmBoX

## .FBX



- Proprietární (Autodesk – 3D Studio MAX)
  - poměrně rozšířené, ale všeobecná snaha nahradit
  - SDK zdarma
    - silně restriktivní licence
  - bez dokumentace, časté změny formátu, záměrné znepřehlednění (obfuscation)
  - binární i textová verze

# COLLADA .DAE



- COLLABorative Design Activity
- Soudobý **kvalitní** formát pro **editaci a trvalé uložení**
- Založeno na XML + komprese
- Spravováno Khronos skupinou – otevřený formát
  - stejně jako OpenGL, OpenCL apod.
- Široká podpora vlastností
  - popis 3D scény, textury, shadery, animace
  - fyzikální vlastnosti (Bullet Physics, PhysX...)
- Široká podpora software
  - Maya, 3DS Max, LightWave 3D, Cinema 4D, Softimage, SketchUp, Blender, Unreal Engine, Google Earth ...
- Maximální kvalita, ale **nevhodné pro online použití**

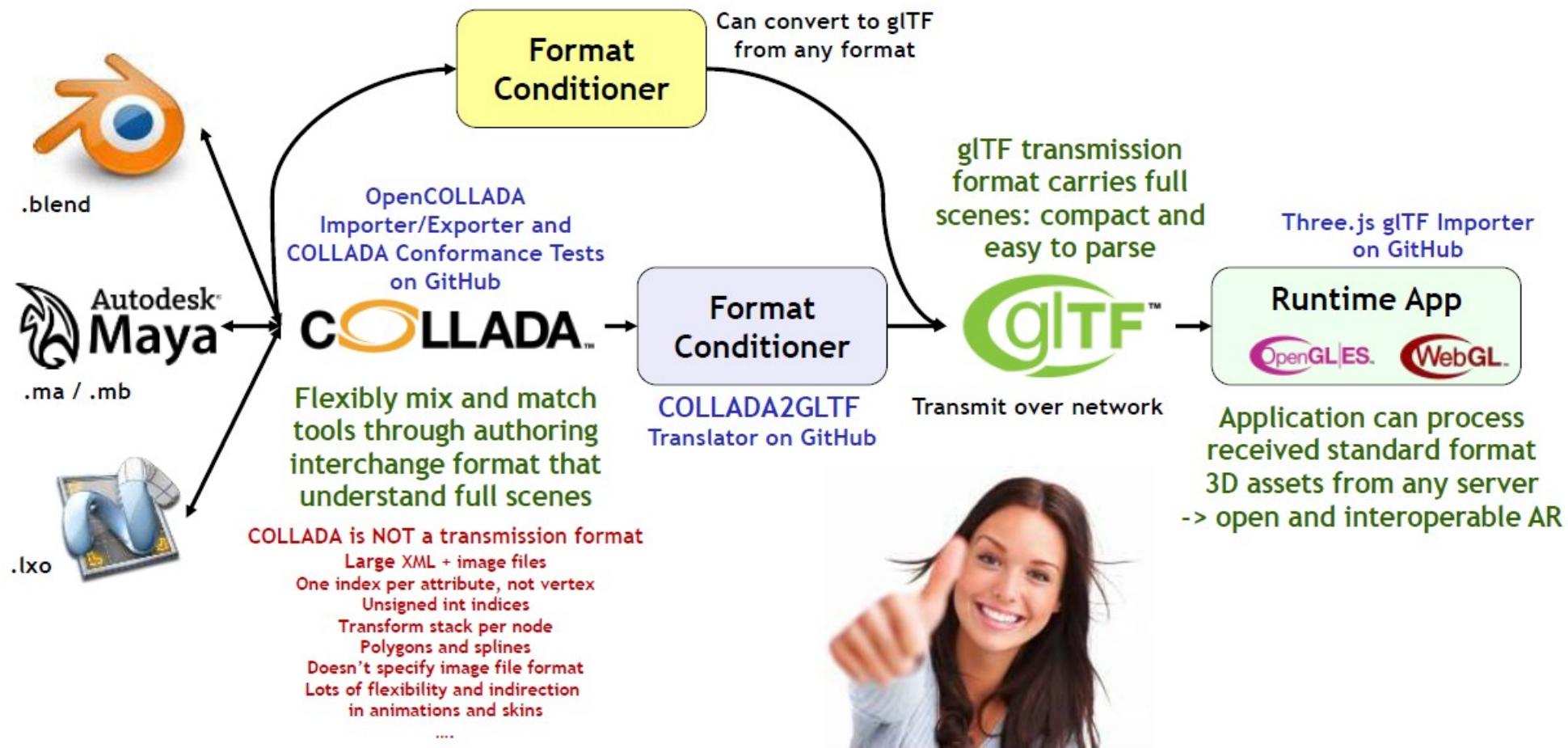
# OpenGL Transmission Format glTF



- Bohaté možnosti – jako Collada
  - ale optimalizován pro rychlé načtení a okamžité použití
- Aktuální formát pro **přenos a online použití**
  - online 3D aplikace, 3D na webu, herní engine, ...
- Spravováno Khronos skupinou
  - otevřený standard stejně jako Collada, OpenGL, apod.

Audio	Video	Images	3D	glTF :)
MP3	H.264	JPEG	?	
napster.	YouTube™	facebook	!	

# 3D Model Creation and Deployment Standards!



# WebGL, WebGPU (APIs, not formats)

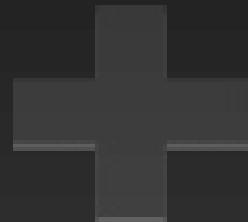
- Managed by Khronos group
- Based on
  - WebGL → OpenGL ES 2.0 (PlayStation 3)
  - WebGPU → Vulkan + Metal + D3D12
- JavaScript based
- Standard for 3D on web
  - part of HTML 5
  - no need for plugins
  - HW acceleration
- Support in all major browsers
  - WebGL = now
  - WebGPU = emerging

# Příklady modelů

<https://free3d.com/3d-models/>

# Programování 3D grafiky běžící v reálném čase

# 3D API



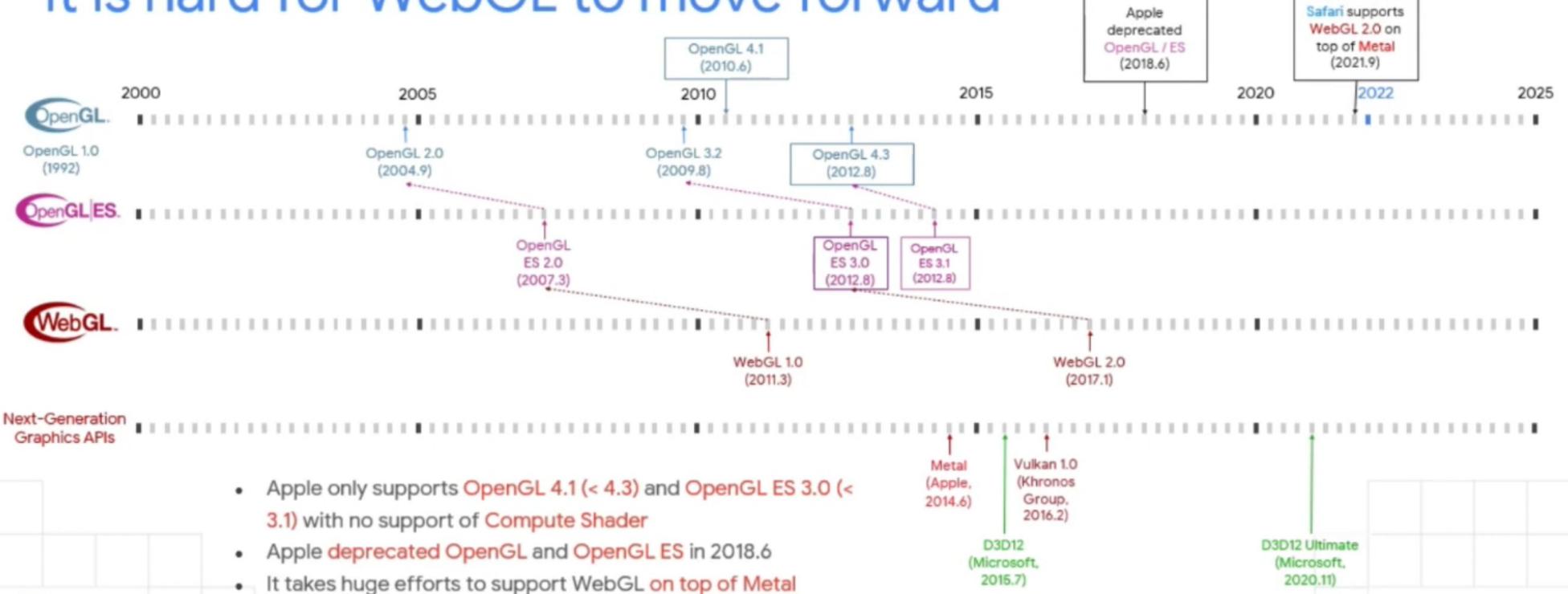
Microsoft®  
DirectX11



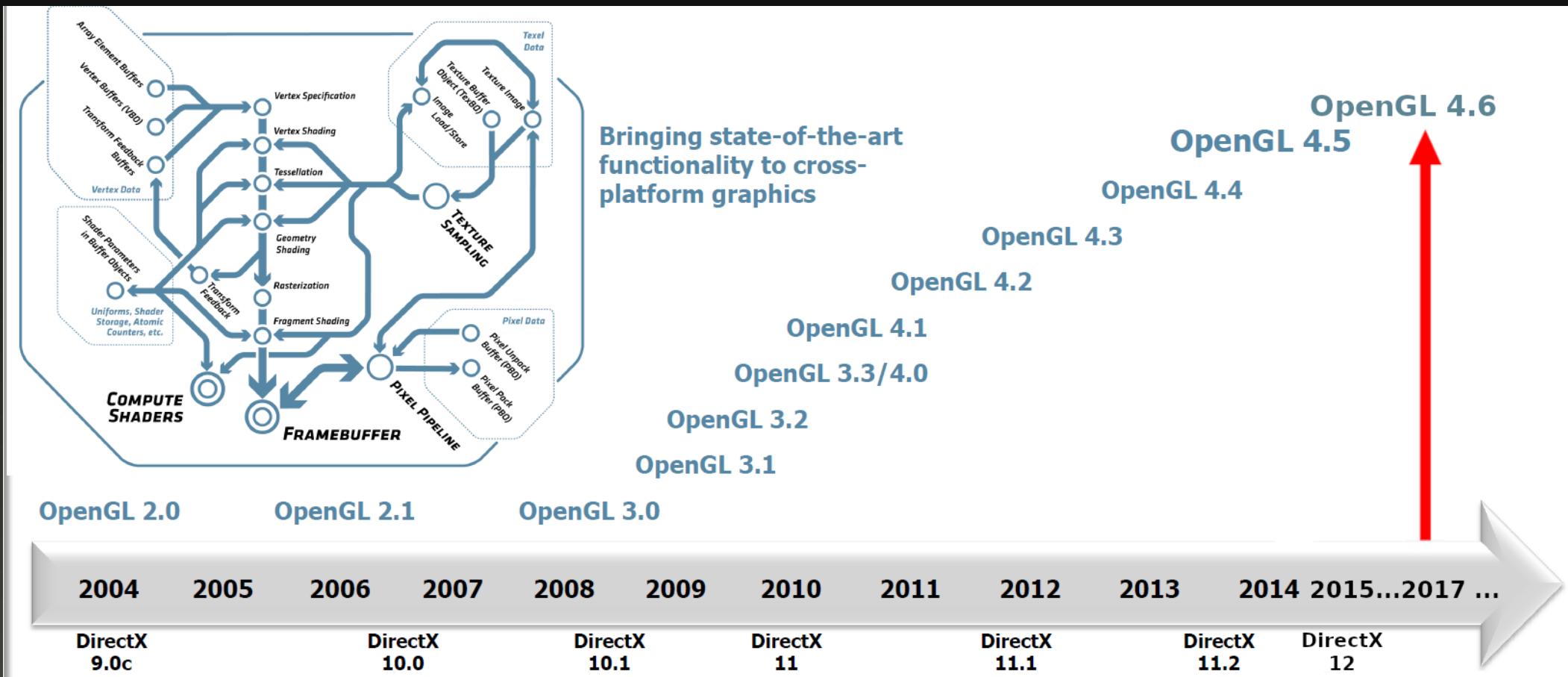
# Vývoj 3D API

## WebGL Issues

It is hard for WebGL to move forward



# Postupný vývoj OpenGL



# OpenGL



- **Open Graphics Library**, knihovna pro tvorbu 2D a 3D grafiky
- Multiplatformní
  - nezávislá na OS, HW, jazyce
    - Win, Linux, OSX, mobily, ...
    - C++, Java, .NET, Python, ...
- Vznik v Silicon Graphics počátkem 90. let
- Používané verze
  - (1.0 … 1.3), 1.4, (1.5 … 2.0), 2.1, (3.0, 3.1), 3.2, (4.0 … 4.5), 4.6

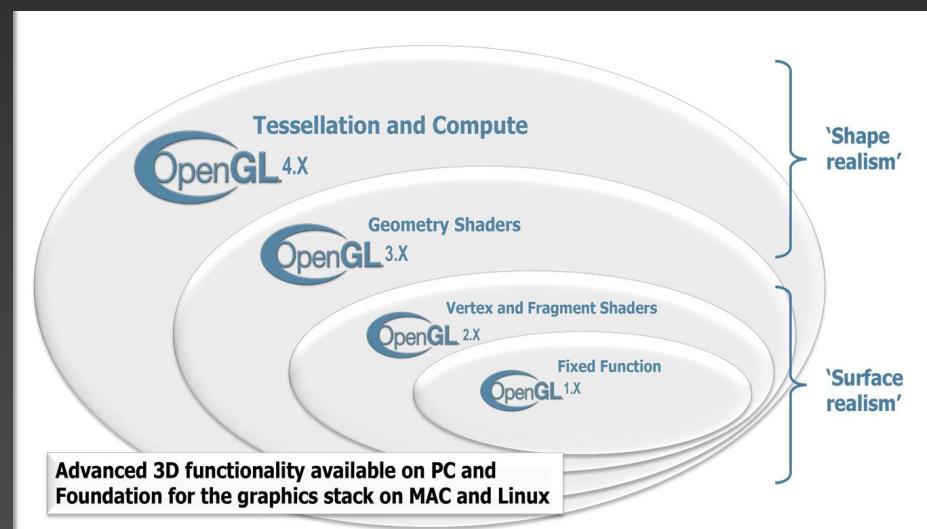


# OpenGL ekosystém



# Důležité verze OpenGL

- 1.x ... 1.4 – fixed pipeline
  - 1.2 – imaging subset
- 2.x – GLSL – OpenGL Shading Language
  - programovatelné shadery, obdobné C
- 3.x
  - 3.0 – první použití pojmu „deprecated“
    - v podstatě vše 1.x
  - 3.2 – rozdělení na „core“ a „compatibility“
- 4.x – moderní techniky
  - 4.0 – teselace
  - 4.3 – obecné výpočty,  
OpenGL ES 3.0 kompatibilita





# Vlastnosti

- Na různých platformách přibližně stejný výsledek
  - nemusí být bitově shodný
  - ale scéna „vypadá stejně“
  - pro použití loga OpenGL je nutné ověření kvality implementace
- Pouze vykreslování
  - nepodporuje vstup z klávesnice, myši
  - nepodporuje práci s okny
    - platformě závislý kód
- Procedurální vykreslování
  - voláním funkcí („**drawcall**“) tvoříme scénu
  - skládání primitiv (obvykle trojúhelníky)
- Retained mód
  - data scény uložena **předem** v GPU
  - vykreslena najednou, nejlépe jediným drawcall → **rychlejší**
- Vlastní datové typy
  - přenositelnost mezi HW a platformami

# Vnitřní stavový automat

- Stavový automat
  - nastavíme parametr
    - uloží se do vnitřního stavu
  - provádíme příkazy
    - volba stále aktivní,  
nemusíme opakovat
  - nastavíme jinou hodnotu...
- C++ aplikace (GL 1.1)

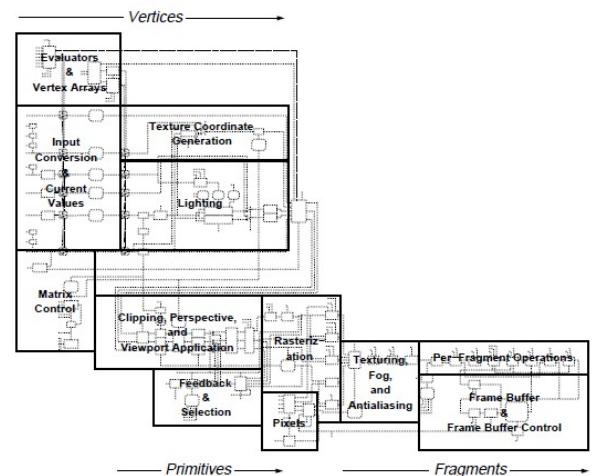
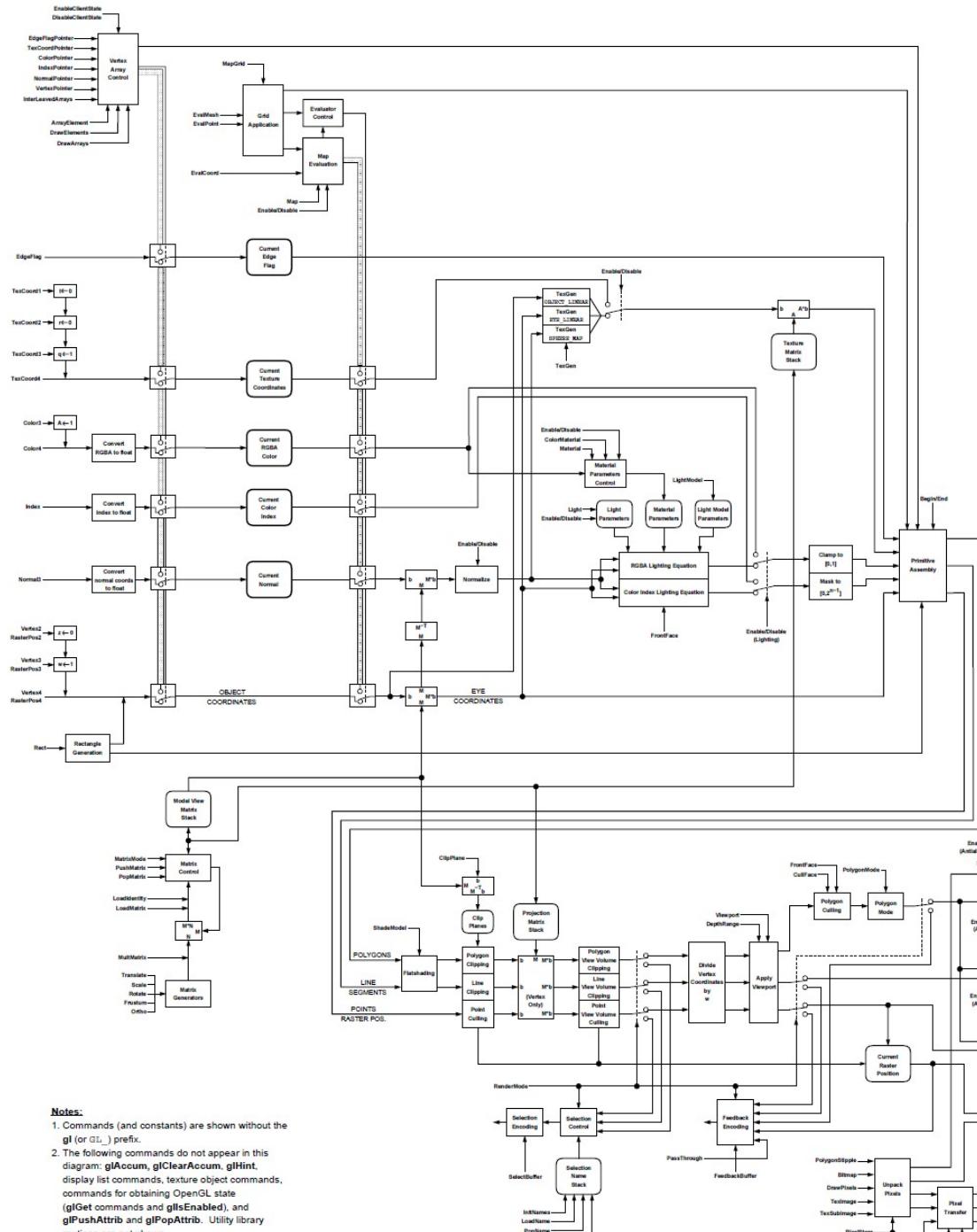
```
glColor4f(R, G, B, A);
glBegin(GL_TRIANGLES);
glVertex3f(x, y, z);
glVertex3f(x2, y2, z2);
glVertex3f(x3, y3, z3);
//...more vertices...
glEnd();
```

```
class GL_context {
public:
    void glColor4f(GLfloat r, GLfloat g, GLfloat b, GLfloat a) { current_color = glm::vec4(r,g,b,a); }
    void glBegin(GLenum prim) { current_primitive = prim; }
    void glEnd(void) { current_primitive = GL_NONE; }

    void glVertex4f(GLfloat x, GLfloat y, GLfloat z, GLfloat w) {
        if (current_primitive != GL_NONE) {
            if (current_primitive == GL_TRIANGLES) {
                if (cnt < 3)
                    vert_buff[cnt++] = glm::vec4(x, y, z, w);
                else {
                    send_command_to_gpu(this->current_primitive, this->vert_buff, this->current_color, ...);
                }
            }
        }
    }
private:
    glm::vec4 current_color;
    GLenum current_primitive;
    glm::vec4 vert_buff[4];
    int cnt = 0;
};
```

# The OpenGL Machine

The OpenGL® graphics system diagram, Version 1.1. Copyright © 1996 Silicon Graphics, Inc. All rights reserved.



Key to OpenGL Operations

- Notes:**
- Commands (and constants) are shown without the `gl` (or `GL_`) prefix.
  - The following commands do not appear in this diagram: `glAccum`, `glClearAccum`, `glHint`, `display list` commands, texture object commands, commands for obtaining OpenGL state (`glGet` commands and `glIsEnabled`), and `glPushAttrib` and `glPopAttrib`. Utility library routines are not shown.
  - After their execution, `glDrawArrays` and `glDrawElements` leave affected current values indeterminate.
  - This diagram is schematic; it may not directly correspond to any actual OpenGL implementation.

# Rozšíření pro OpenGL (standardní)

- GLU (OpenGL Utility Library)
  - povinná součást každé OpenGL implementace
  - vytváření matic pohledů
  - teselace, evaluátory NURBS
  - generování UV souřadnic
  - pokročilejší primitiva (koule, válec, apod.)
- dnes se obvykle nepoužívá, nahrazena modernějšími knihovnami
  - GLM

# Rozšíření pro OpenGL (samostatná)

- C/C++
  - GLFW
    - nezávislé na platformě
    - základní interakce s okny, myší atd.
    - jen jednoduché aplikace nebo fullscreen
  - GLEW – zpřístupňuje pokročilá GL volání
- C#
  - OpenTK, GLMSharp, ...
- SDL (Simple DirectMedia Layer)
  - podpora pro multimédia, periférie, vlákna, apod.
- GLX (UNIX/Linux), WGL (Windows), CGL (Mac)
  - spolupráce s OS (okna, události, myš, klávesnice...)
- irrKlang - audio
- GLUI, GLEE, OpenSceneGraph, ...
- GLUT (OpenGL Utility Toolkit)
  - od Khronosu, zastaralé, nepoužívat

