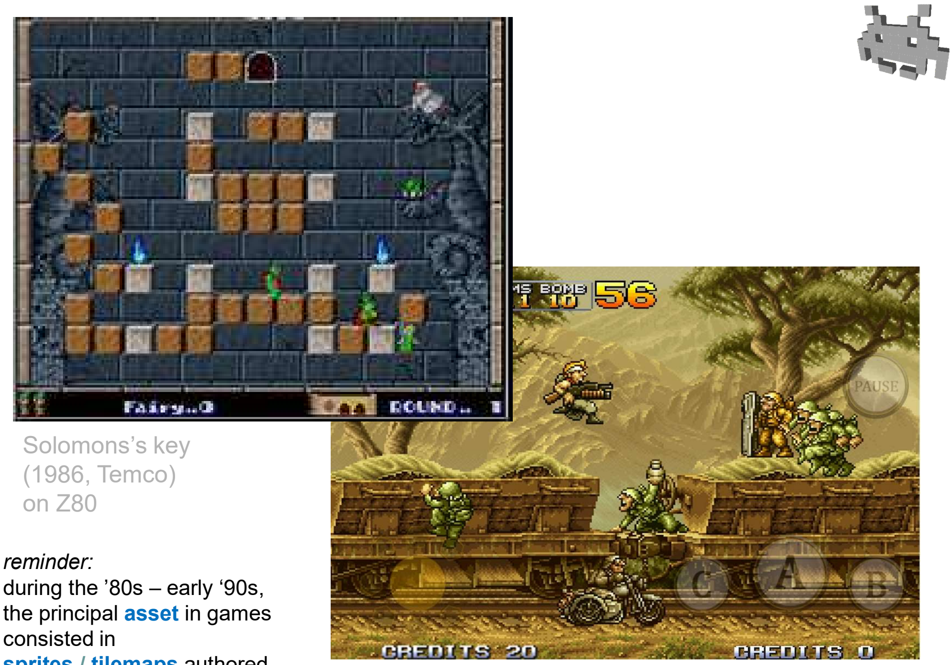
3D video games

Models for Games

Marco Tarini



Solomons's key
(1986, Temco)
on Z80

reminder:
during the '80s – early '90s, the principal **asset** in games consisted in **sprites** / **tilemaps** authored by **pixel artists** ...

Metal Slug (1996, Nazca Copr), on Neo Geo (SNK)

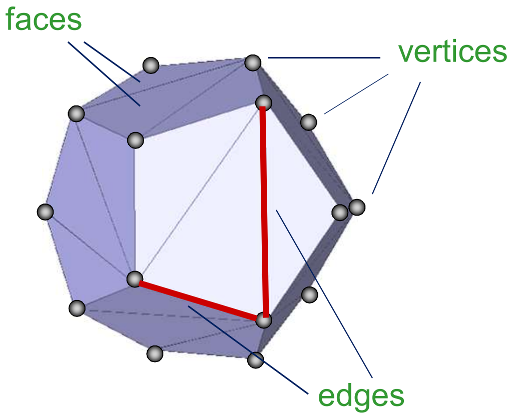
Triangle Meshes

The universal 3D models of games

- Data structure for modeling 3D objects
  - GPU friendly
  - Resolution = number of faces
  - (Potentially) Adaptive resolution
- Piecewise linear surface
  - a bunch of surface samples “vertices”
    connected by a set of triangular “faces”
    attached side to side by “edges”

Triangle Mesh
(or simplicial mesh)

- A set of adjacent triangles



(Polygonal mesh) Mesh: data structure

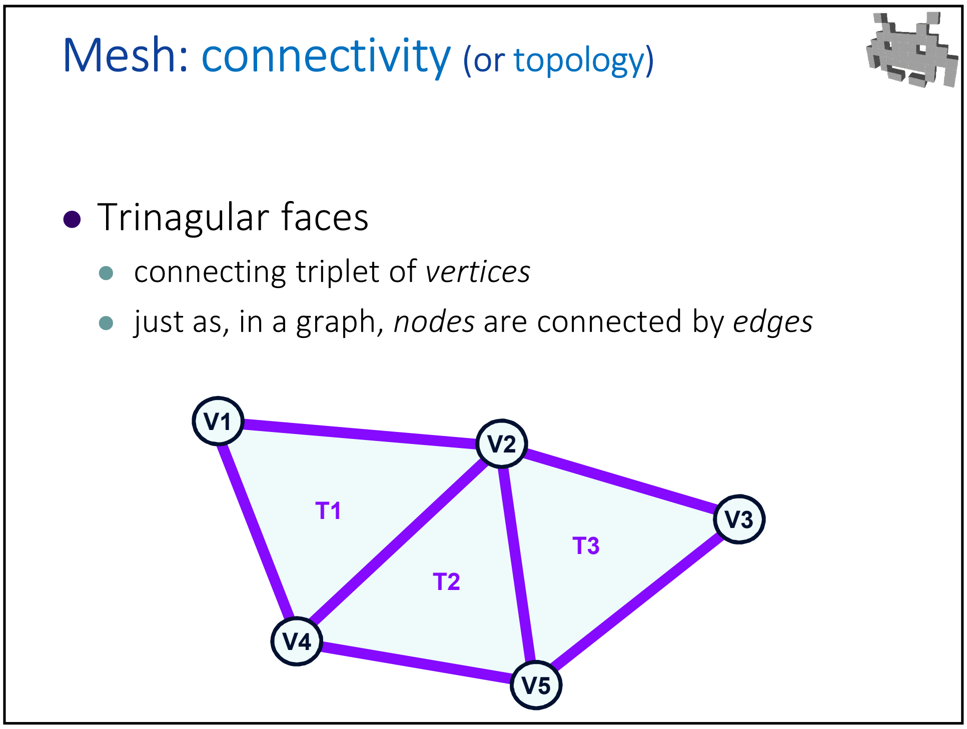
Made of

- geometry
  - The vertices, each with pos (x,y,z)
  - It’s a sampling of the surface
- connectivity or topology
  - Faces connecting the vertices
    - Triangle mesh: faces are triangles (what the GPU is designed to render!)
    - (pure) quad mesh: faces are quadrilateral
    - Quad dominant mesh: most faces are quadrilateral
    - Polygonal mesh: faces are polygons (general case)
- attributes
  - Ex.: color, material, normal, UV, …

Mesh: geometry

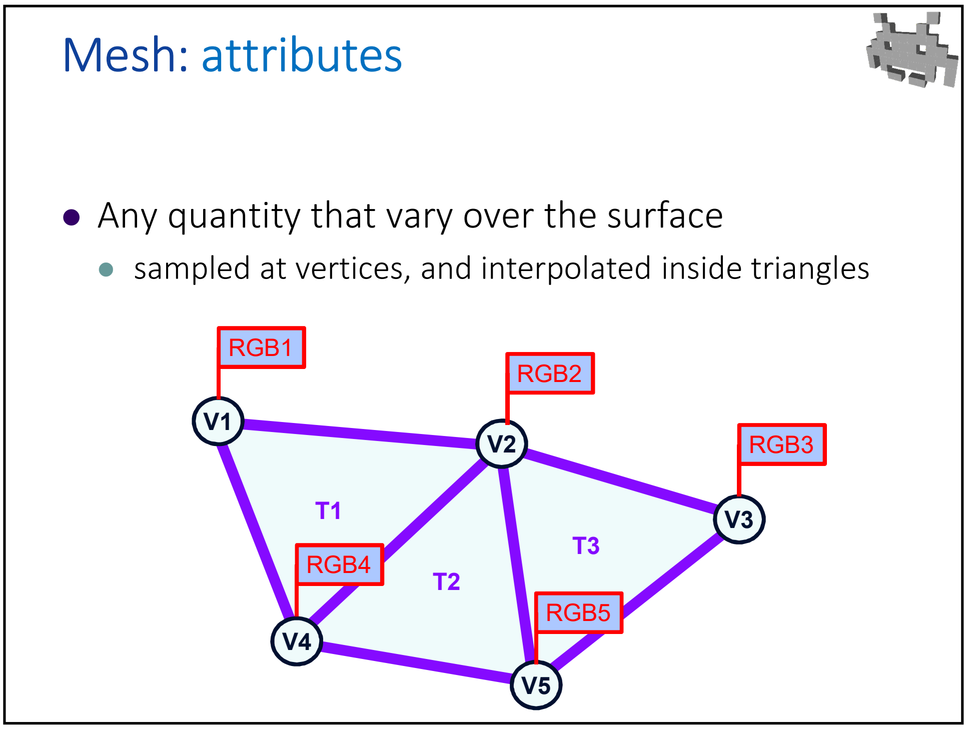
- Vertex position set
  - A position vector (x,y,z) for every vertex
  - Coordinates by def in Object space!

V1
V2
V3
V4
V5



Mesh: connectivity (or topology)

- Trinagular faces
  - connecting triplet of *vertices*
  - just as, in a graph, *nodes* are connected by *edges*



Mesh: attributes

- Any quantity that vary over the surface
  - sampled at vertices, and interpolated inside triangles

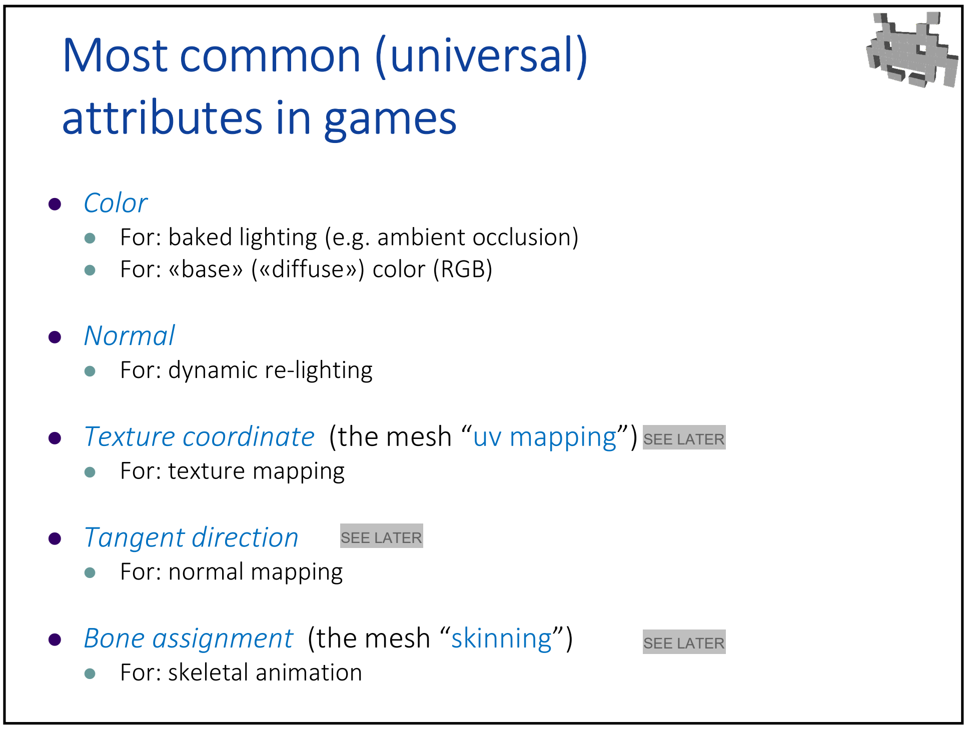
Mesh: attributes

- Properties varying on the surface
  - Vectors or scalars
- Stored for each vertex
  - (at least in games)
- Interpolated within the faces
  - Linear interpolation
- Note: by construction C0 continuous on adjacent faces
  - And in general C1 discontinuous on adjacent faces

- Common attributes in games:
  - *Color*
    - For: baked lighting (ambient occlusion)
    - For: base color (RGB)
  - *Normal*
    - For: dynamic re-lighting
  - *Texture coordinate*
    (the mesh "uv mapping")
    - For: texture mapping LATER
  - *Tangent direction*
    - For: normal mapping LATER
  - *Bone assignment*
    (the mesh "skinning")
    - For: skeletal animation LATER

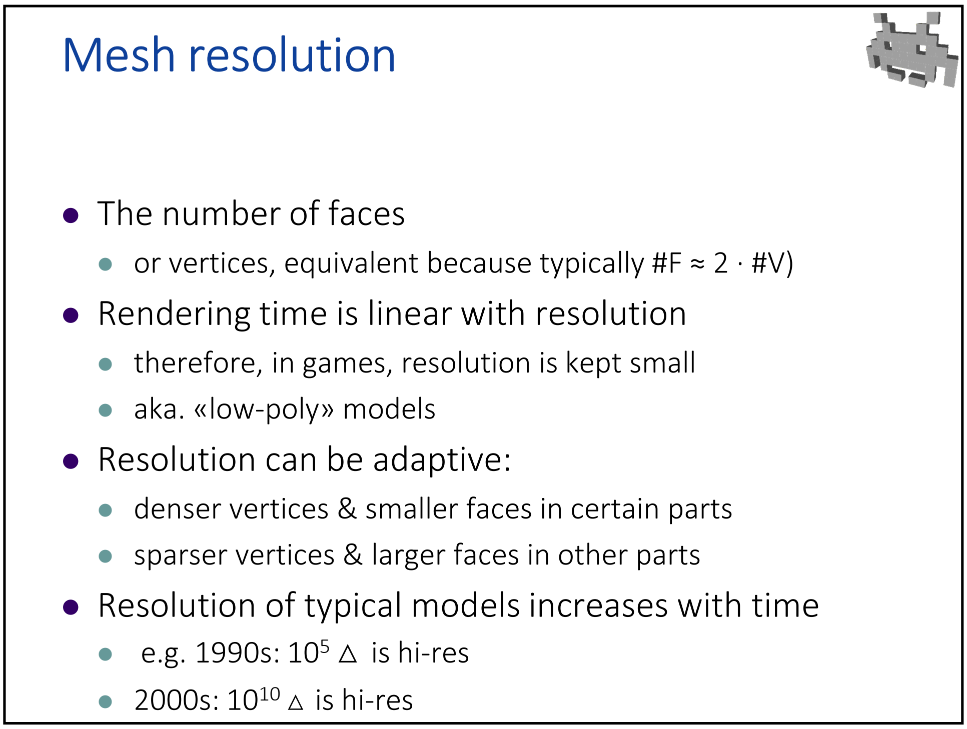
Mesh: attributes

- Properties varying on the surface
  - Vectors or scalars
- Stored for each vertex
  - (at least in games)
- Interpolated within the faces
  - Linear interpolation, with barycentric coords
- Note: *by construction*
  C0 continuous even across faces
  - and in general C1 discontinuous across faces



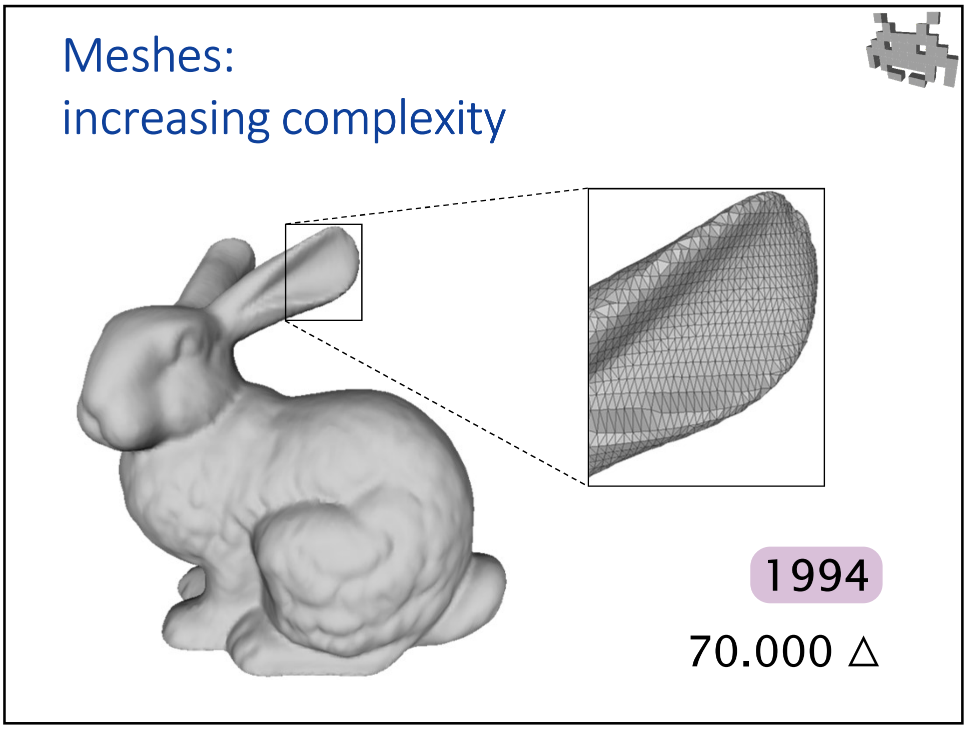
Most common (universal) attributes in games

- *Color*
  - For: baked lighting (e.g. ambient occlusion)
  - For: «base» («diffuse») color (RGB)
- *Normal*
  - For: dynamic re-lighting
- *Texture coordinate* (the mesh "uv mapping") SEE LATER
  - For: texture mapping
- *Tangent direction* SEE LATER
  - For: normal mapping
- *Bone assignment* (the mesh "skinning") SEE LATER
  - For: skeletal animation



Mesh resolution

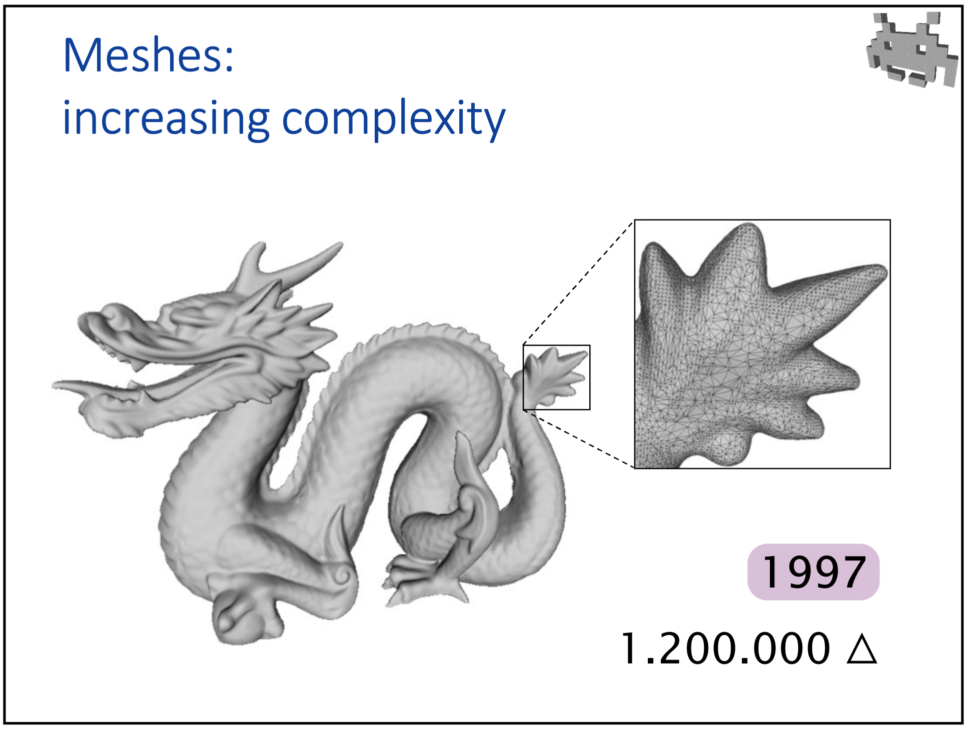
- The number of faces
  - or vertices, equivalent because typically $\#F \approx 2 \cdot \#V$)
- Rendering time is linear with resolution
  - therefore, in games, resolution is kept small
  - aka. «low-poly» models
- Resolution can be adaptive:
  - denser vertices & smaller faces in certain parts
  - sparser vertices & larger faces in other parts
- Resolution of typical models increases with time
  - e.g. 1990s: 10^5 △ is hi-res
  - 2000s: 10^{10} △ is hi-res



Meshes: increasing complexity

1994

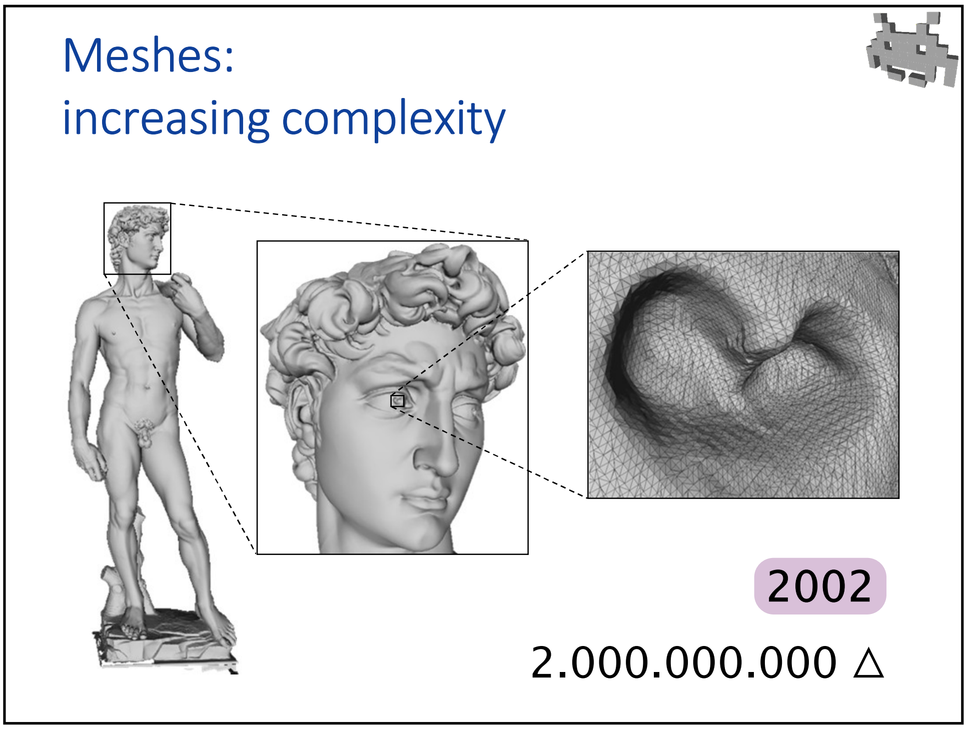
70.000 △



Meshes: increasing complexity

1997

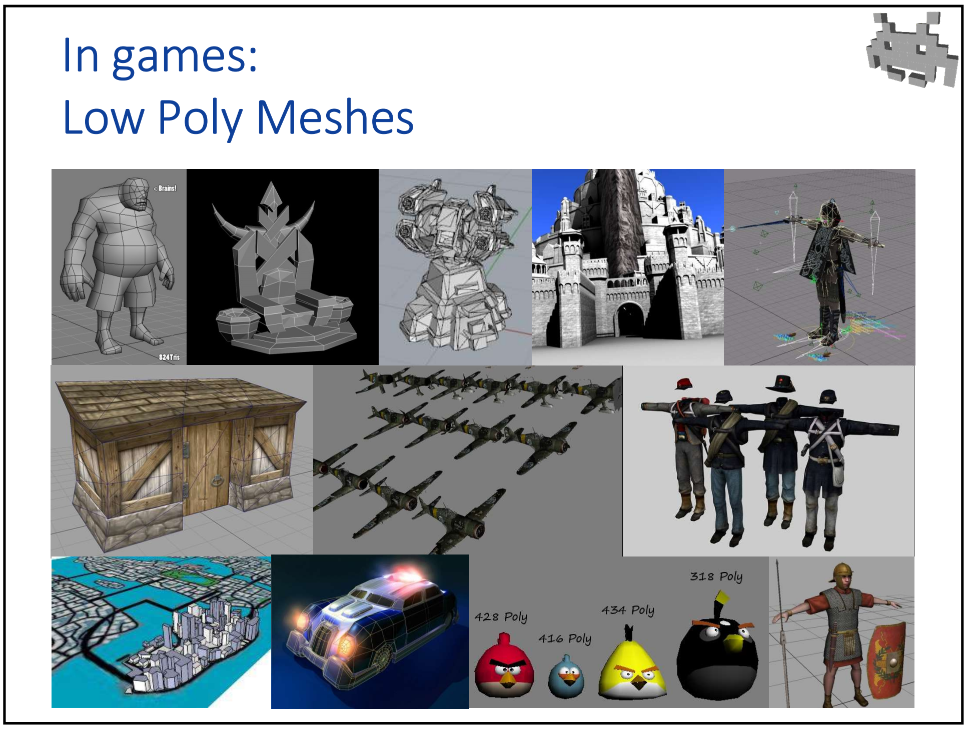
1.200.000 △



Meshes:
increasing complexity

2002

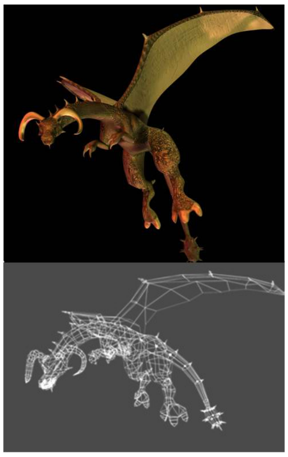
2.000.000.000 △



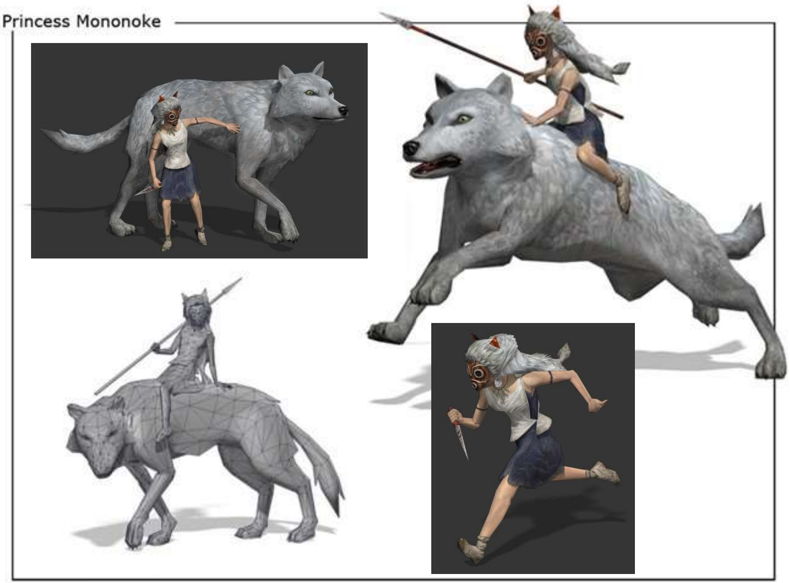
In games:
Low Poly Meshes

But… in games

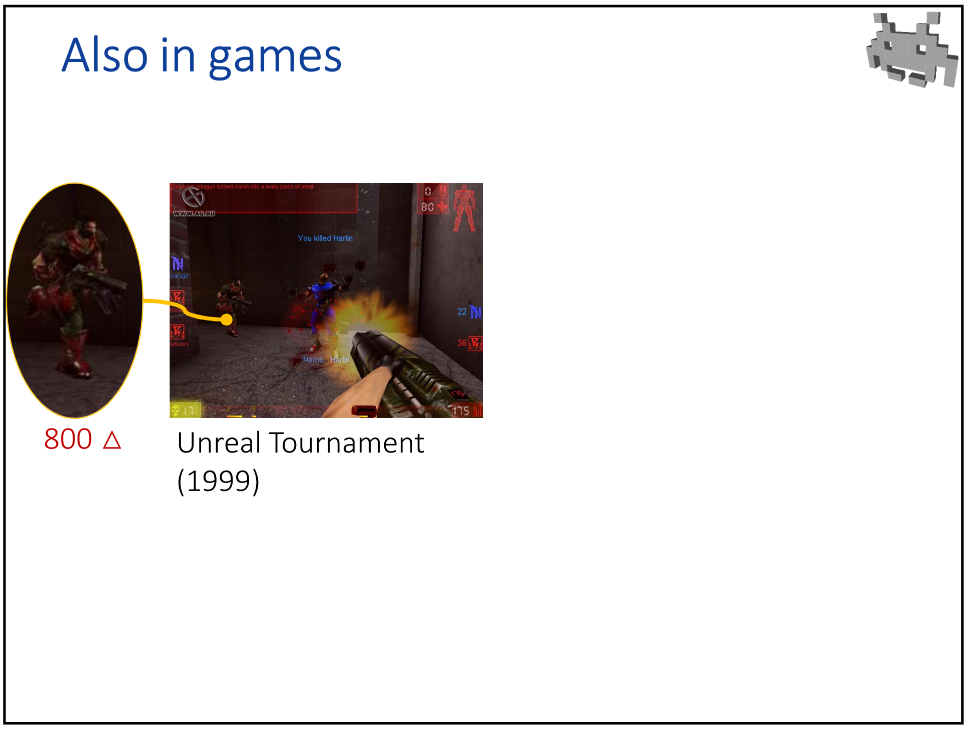
- LOW POLY MODELING!



Low-poly modeling



by Phillip Heckinger (3D modeller)



Also in games

800 △

Unreal Tournament
(1999)

Also in games

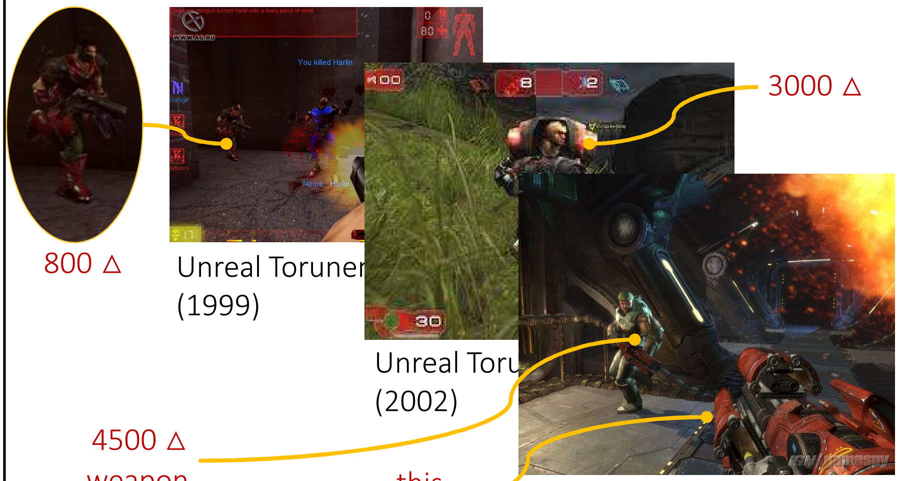
800 △

Unreal Tourna
(1999)

3000 △

Unreal Tournament 2K3
(2002)

Also in games



800 △

Unreal Toruner
(1999)

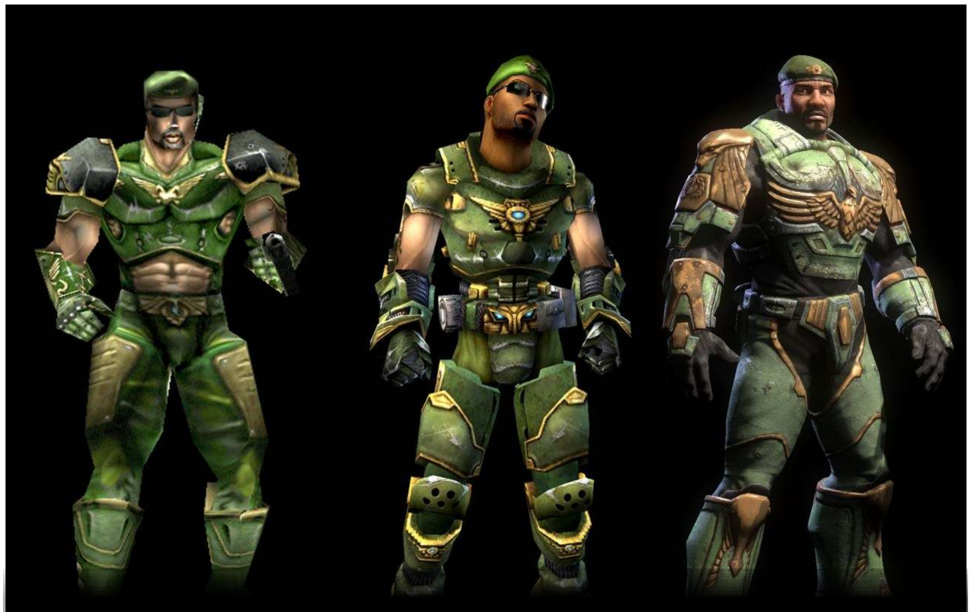
3000 △

Unreal Toru
(2002)

4500 △
weapon

this
12000 △

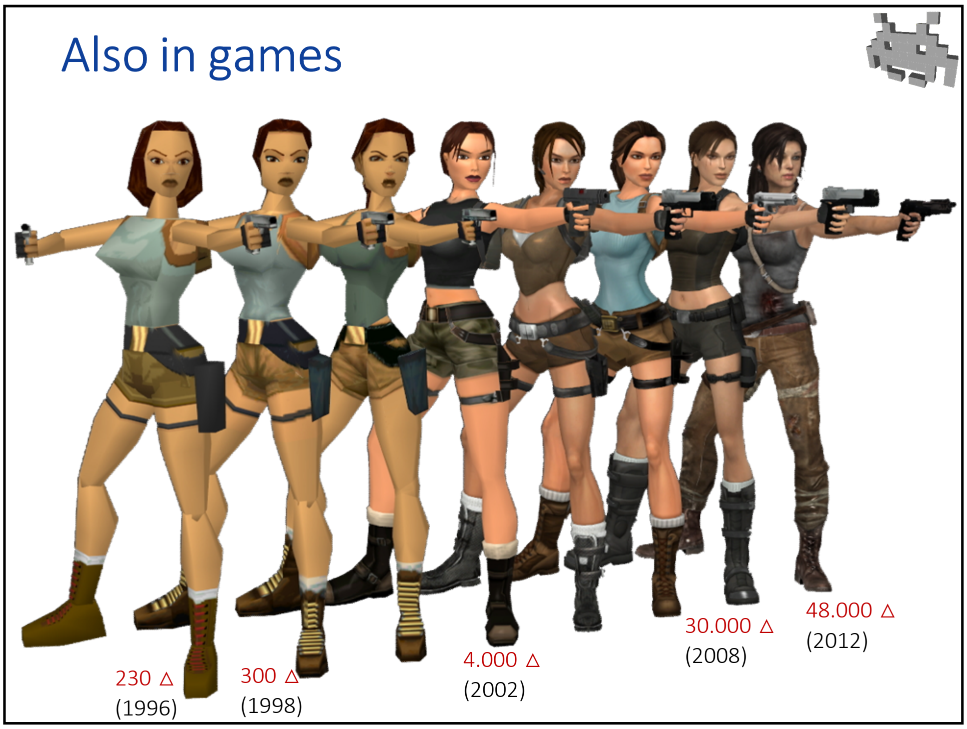
Unreal Tournament 3
(2007)



800 △
(1999)

3000 △
(2002)

15000 △
(2006)



Also in games

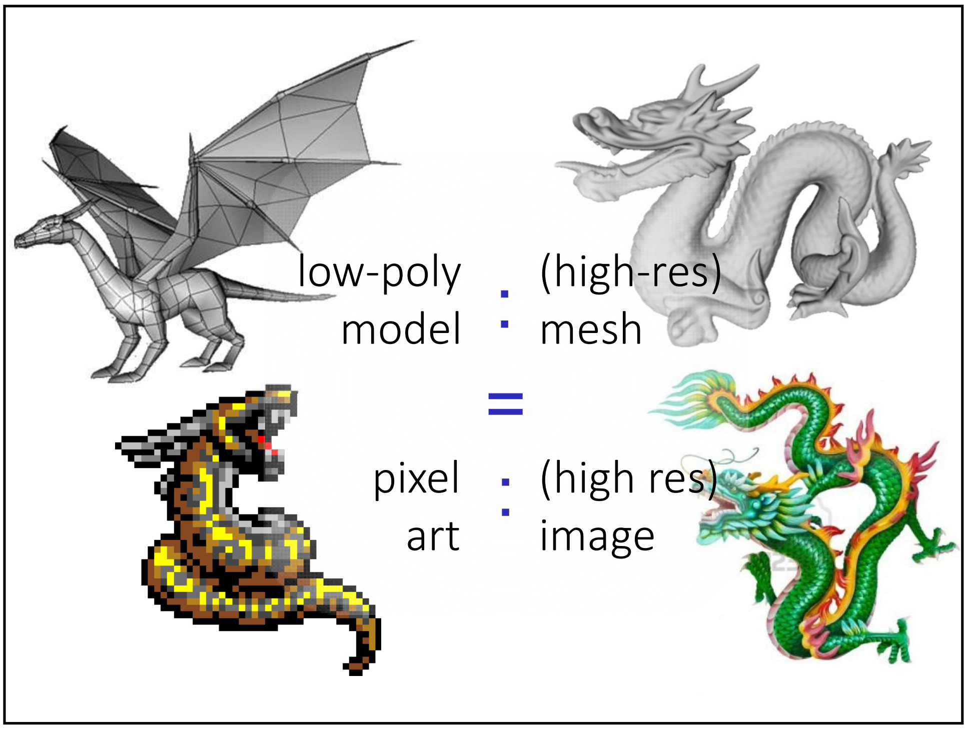
230 △
(1996)

300 △
(1998)

4.000 △
(2002)

30.000 △
(2008)

48.000 △
(2012)



low-poly model : (high-res) mesh

=

pixel art : (high res) image

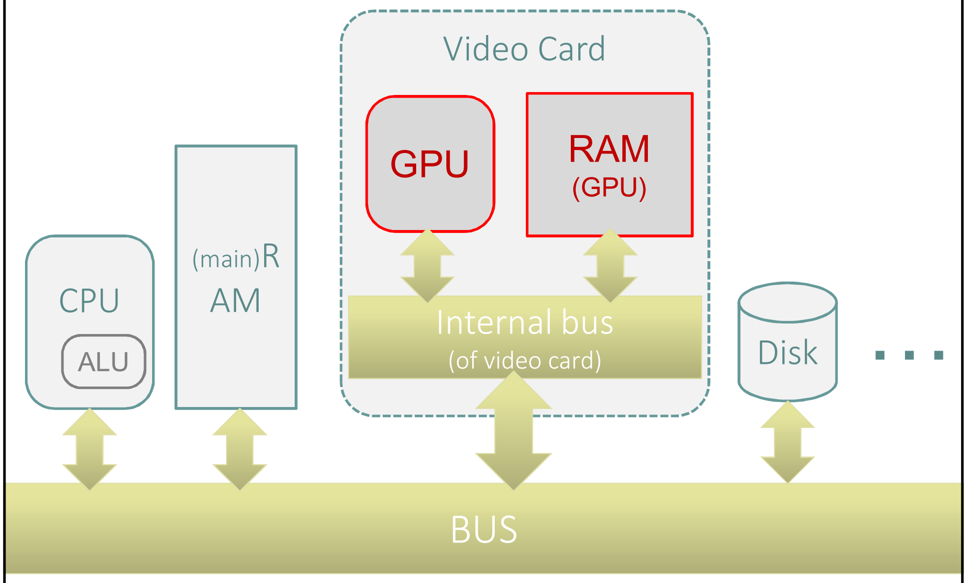
Rendering of a Mesh in a nutshell

- Load…
  - store all data on GPU RAM
    - Geometry + Attributes
    - Connectivity
    - Textures
    - Shaders
    - Parameters / Settings
- …and Fire!
  - send the command: “*do it*” !

THE MESH

THE “MATERIAL”

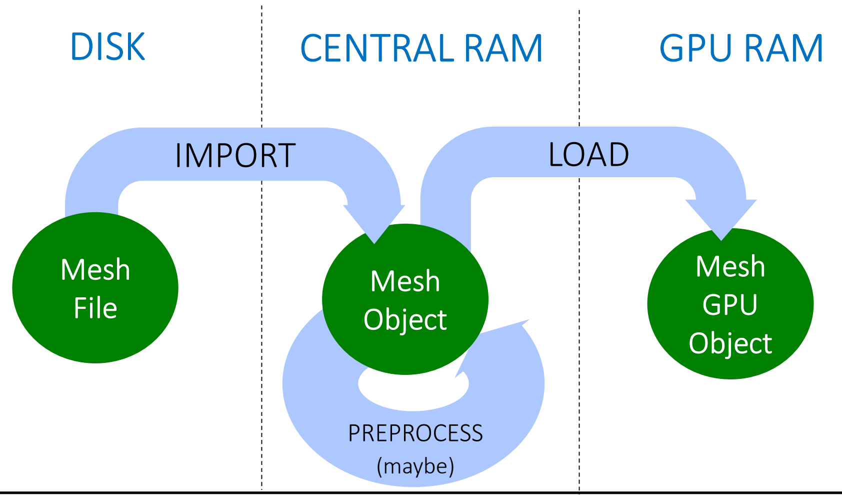
Simplified schema of: “PC + Video Card”

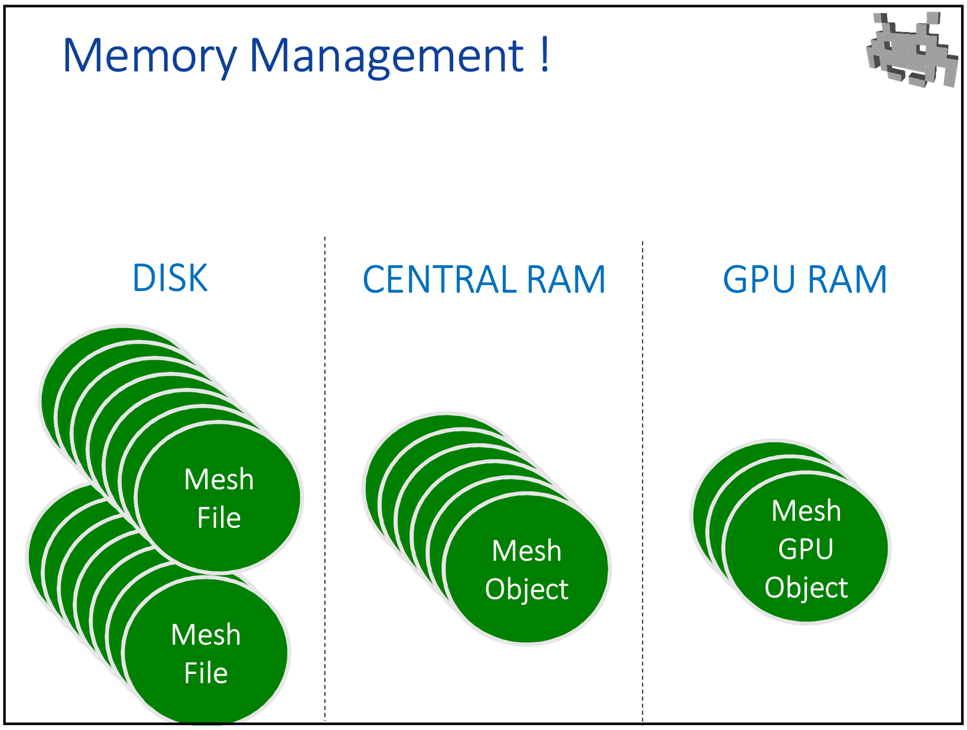


Tasks of the Game Engine for Meshes

- Import (from disk)
- Simple Pre-processing
  - e.g.: Compute Normals (if needed, i.e. rarely)
  - e.g.: Compute Tangent Dirs
  - e.g.: Bake Lighting (sometimes)
- Render
  - (graphic engine)
  - GPU based
  - + animate (more about this later)

Life of a Mesh in a Game Engine





Memory Management !

DISK

CENTRAL RAM

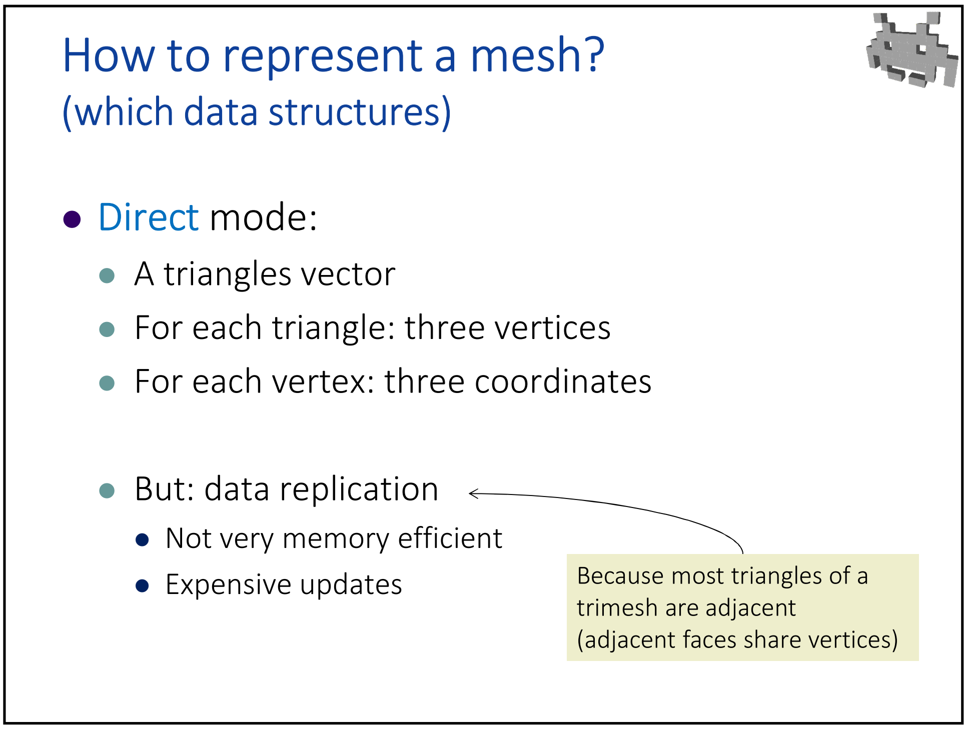
GPU RAM

Mesh File

Mesh File

Mesh Object

Mesh GPU Object



How to represent a mesh?
(which data structures)

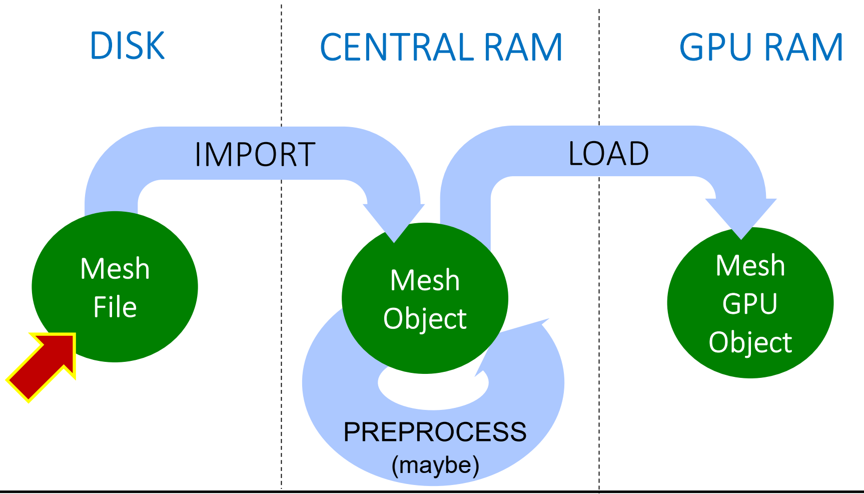
- Direct mode:
  - A triangles vector
  - For each triangle: three vertices
  - For each vertex: three coordinates
  - But: data replication
    - Not very memory efficient
    - Expensive updates

Because most triangles of a trimesh are adjacent
(adjacent faces share vertices)

How to represent a mesh?
(which data structures)

- Indexed mode:
  - Geometry: vertices array
    - For each vertex: position and attributes
  - Attributes:
    - On vertices
      - (ex.: members of class "Vertex")
  - Connectivity: (sometimes: "topology")
    - Triangles array
    - For each triangle:
      - triplet of indices (referring to a vertex)

Life of a Mesh
in a Game Engine



DISK

CENTRAL RAM

GPU RAM

IMPORT

LOAD

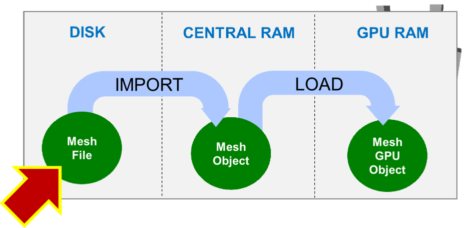
Mesh File

Mesh Object

Mesh GPU Object

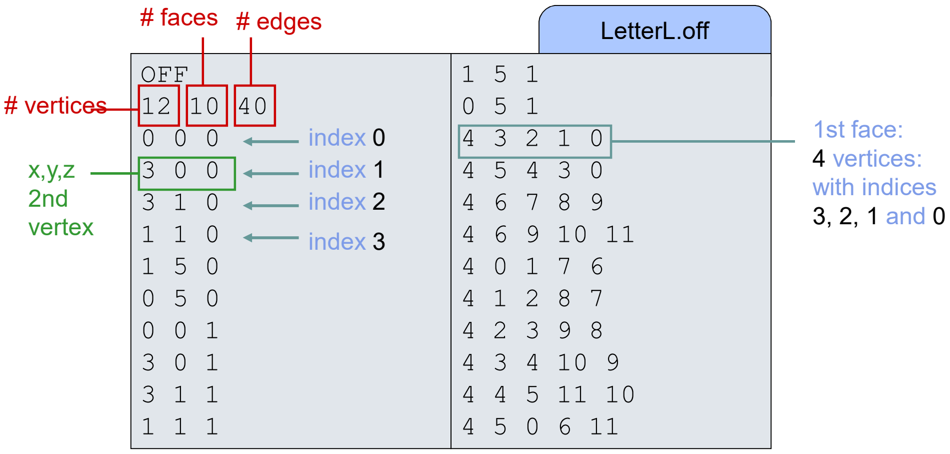
PREPROCESS
(maybe)

Mesh File (as asset)



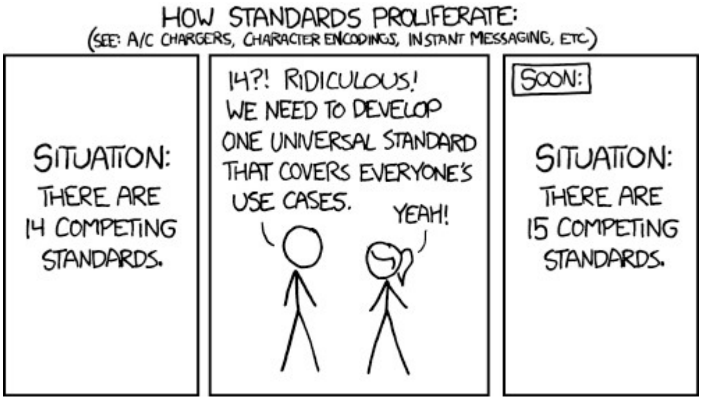
- A file of a given format sitting on the disk
- Choices for the game engine:
  - which formats(s) to import?
    - proprietary, standard…
  - storing which attribute?
- Issues:
  - storage cost
  - loading time

Example of file format for indexed meshes: OFF format



```
OFF
12 10 40
0 0 0
3 0 0
3 1 0
1 1 0
1 5 0
0 5 0
0 0 1
3 0 1
3 1 1
1 1 1
1 5 1
0 5 1
4 3 2 1 0
4 5 4 3 0
4 6 7 8 9
4 6 9 10 11
4 0 1 7 6
4 1 2 8 7
4 2 3 9 8
4 3 4 10 9
4 4 5 11 10
4 5 0 6 11
```

File formats for meshes …

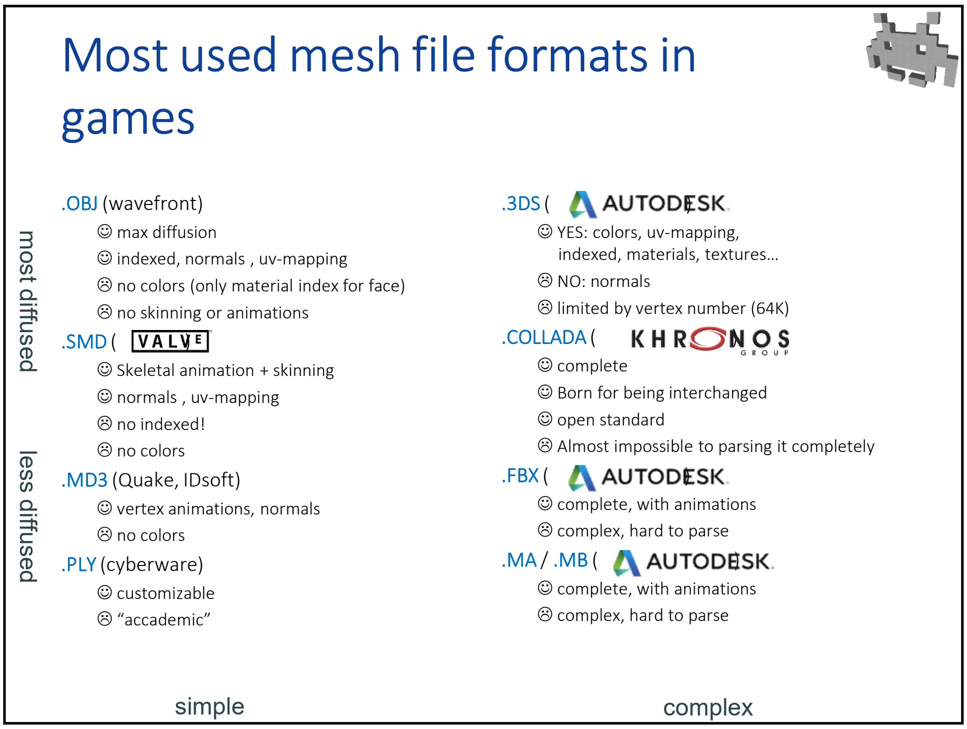

HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

(from **xkcd.com**)

File formats for meshes

(a Babel tower!)

- 3DS - 3D Studio Max file format
- OBJ - Another file format for 3D objects
- MA, MB - Maya file formats
- 3DX - Rinoceros file format
- BLEND - Blender file format
- DAE - COLLADA file format (Khornos)
- FBX - Autodesk interchange file format
- X - Direct X object
- SMD - good for animations (by Valve)
- MD3 - quake 3 vertex animations
- DEM - Digital Elevation Models
- DXF - (exchange format, Autodesk's AutoCAD)
- FIG - Used by REND386/AVRIL
- FLT - MulitGen Inc.'s OpenFlight format
- HDF - Hierarchical Data Format
- IGES - Initial Graphics Exchange Specification
- IV - Open Inventor File Format Info
- LWO, LWB & LWS - Lightwave 3D file formats
- MAZ - Used by Division's dVS/dVISE
- MGF - Materials and Geometry Format
- MSDL - Manchester Scene Description Language
- 3DML - by Flatland inc.
- C4D – Cinema 4D file format
- SLDPTR - SolidWork "part"
- WINGS - Wings3D object
- NFF - Used by Sense8's WorldToolKit
- SKP - Google sketch up
- KMZ - Google Earth model
- OFF - A general 3D mesh Object File Format
- OOGL - Object Oriented Graphics Library
- PLG - Used by REND386/AVRIL
- POV - “persistence of vision” ray-tracer
- QD3D - Apple's QuickDraw 3D Metafile format
- TDDD - for Imagine & Turbo Silver ray-tracers
- NFF & ENFF - (Extended) Neutral File Format
- VIZ - Used by Division's dVS/dVISE
- VRML, VRML97 - Virtual Reality Modeling Language (RIP)
- X3D - tentato successore di VRML
- PLY - introdotto by Cyberware – tipic. dati range scan
- DICOM - Dalla casa omonima – tipic. dati CAT scan
- Renderman - per l'omonimo visualizzatore
- RWX - RenderWare Object
- Z3D - ZModeler File format
- etc, etc, etc…



Most used mesh file formats in games

most diffused

less diffused

.OBJ (wavefront)
- ☺ max diffusion
- ☺ indexed, normals , uv-mapping
- ☹ no colors (only material index for face)
- ☹ no skinning or animations

.SMD (VALVE
- ☺ Skeletal animation + skinning
- ☺ normals , uv-mapping
- ☹ no indexed!
- ☹ no colors

.MD3 (Quake, IDsoft)
- ☺ vertex animations, normals
- ☹ no colors

.PLY (cyberware)
- ☺ customizable
- ☹ "accademic"

.3DS (AUTODESK
- ☺ YES: colors, uv-mapping, indexed, materials, textures…
- ☹ NO: normals
- ☹ limited by vertex number (64K)

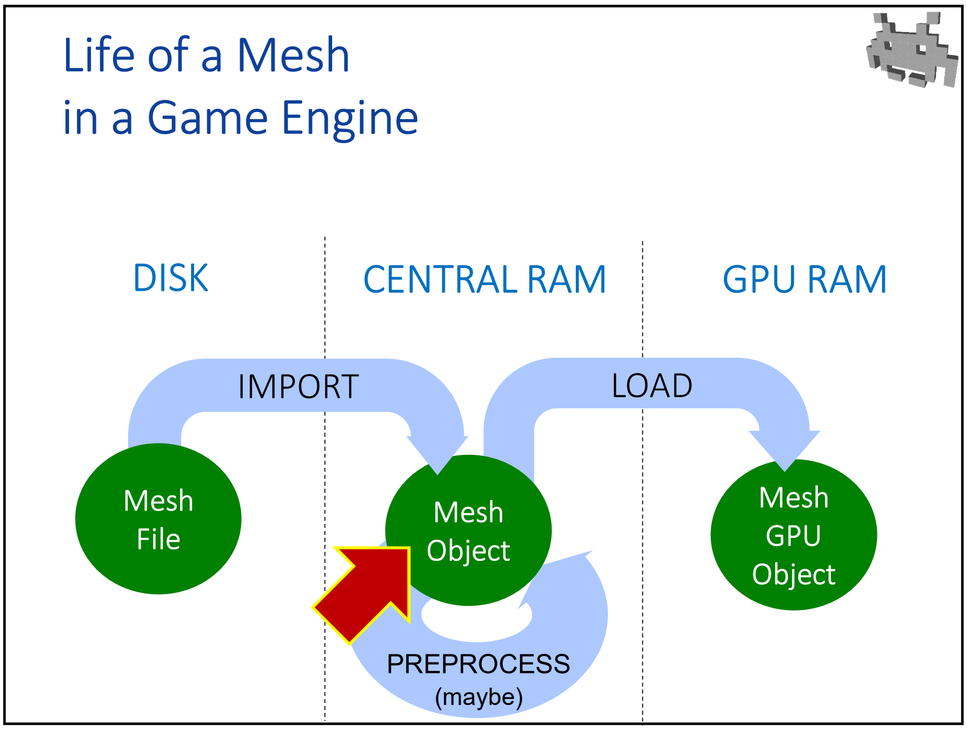
.COLLADA (KHRONOS GROUP
- ☺ complete
- ☺ Born for being interchanged
- ☺ open standard
- ☹ Almost impossible to parsing it completely

.FBX (AUTODESK
- ☺ complete, with animations
- ☹ complex, hard to parse

.MA / .MB (AUTODESK
- ☺ complete, with animations
- ☹ complex, hard to parse

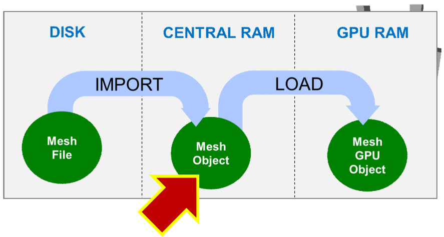
simple

complex



Life of a Mesh in a Game Engine

Mesh Object (in RAM)



- A (C++ / Javascript / etc) structure in main RAM
- Choices for the game engine:
  - which attribute to store?
  - storage formats… (floats, bytes, double…)
  - which preprocessing to offer (typically at load time)

How to represent a mesh? (which data structures)

- Indexed mode in C++ :

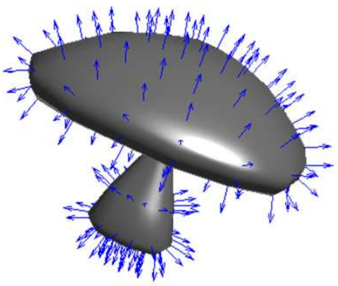
```
class Vertex {
  vec3 pos;
  rgb color;   /* attribute 1 */
  vec3 normal; /* attribute 2 */
};

class Face{
   int vertexIndex[3];
};

class Mesh{
  vector<Vertex> verts;  /* geom + attr */
  vector<Face> faces;    /* connectivity */
};
```

Most common attribute: the normal

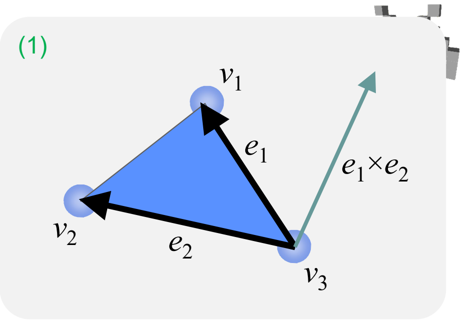
- Unit direction vector
- Represents the surface orientation
- Used for lighting
- Sometimes computed automatically from geometry...
- But the artist decides which edges are *soft* and which are *hard*



Computing normals from geometry

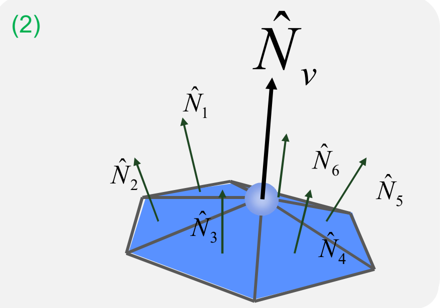
(1) normals for face

(2) normals for vertex



$$N = \hat{N}_1 + \hat{N}_2 + \ldots + \hat{N}_n$$

$$\hat{N} = \frac{N}{|N|}$$

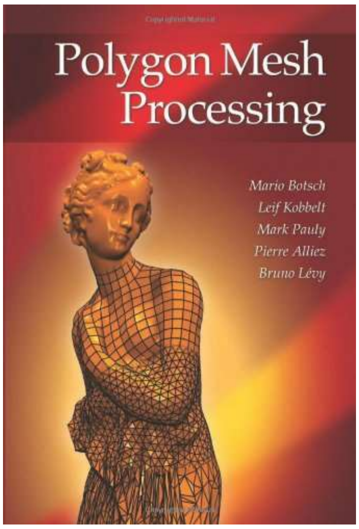


Mesh processing: part of Geometry Processing

- See also GID course

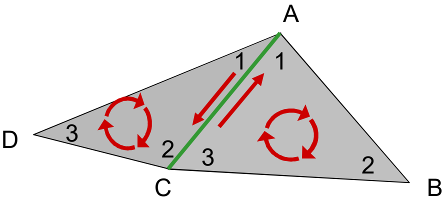
Mesh processing aka Geometry Processing

- A good manual for mesh processing programming:

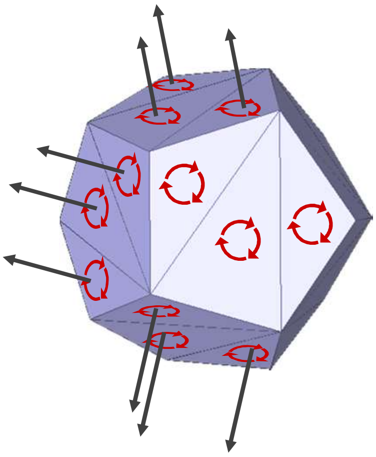


Compute normals from geometry

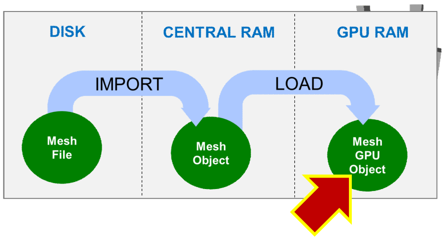
- Note:
  the face orientations
  must be coherent



Opposite direction, coherent edge



Mesh GPU Object (on Graphic Card)



DISK

CENTRAL RAM

GPU RAM

IMPORT

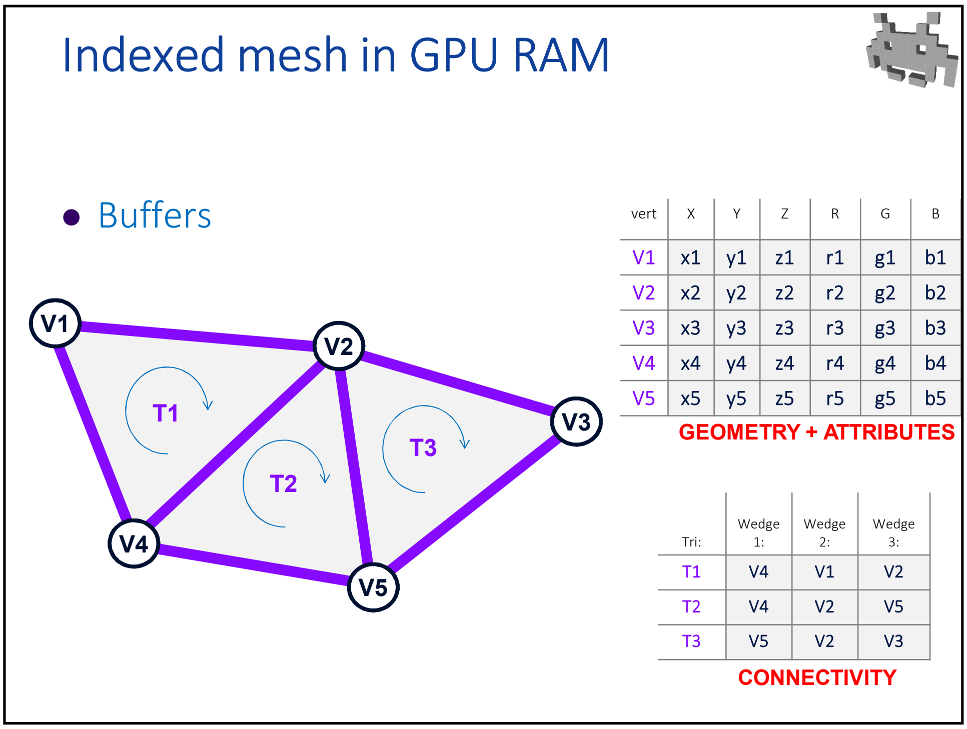
LOAD

Mesh File

Mesh Object

Mesh GPU Object

- VBO / Vertex Arrays / etc
  - buffers storing “tables” for geometry, connectivity, etc.
- Sitting in GPU RAM
  - *The most precious one !*
- Ready to render!
- Choices for Game Engine:
  - which GPU mechanism
  - storage formats
  - balance storage cost / precision / computation



Indexed mesh in GPU RAM

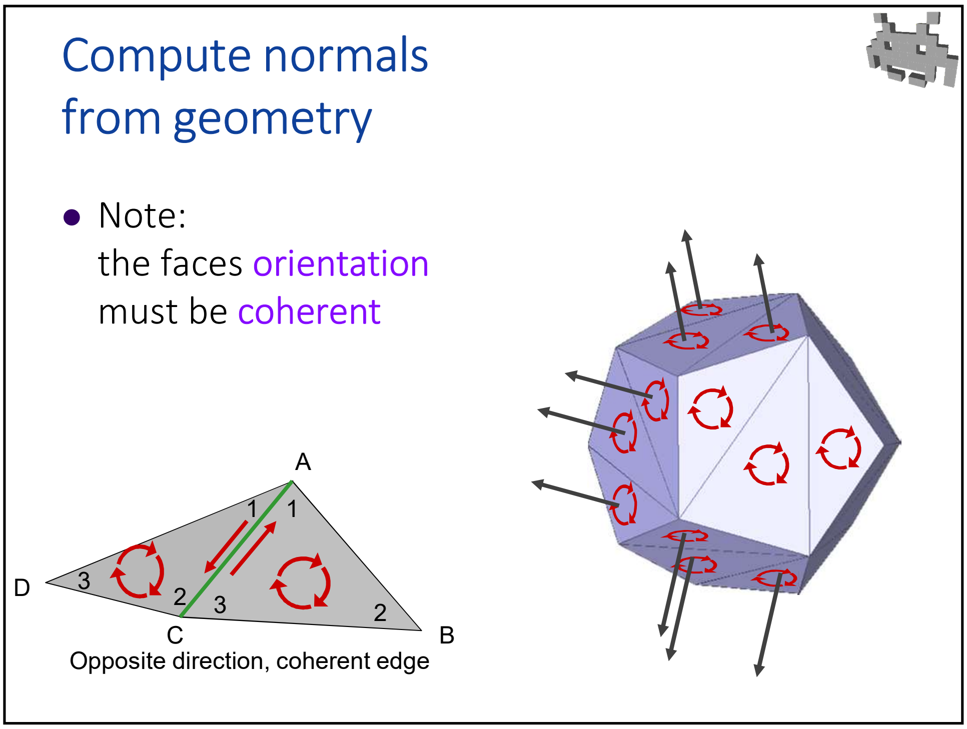
- Buffers

vert	X	Y	Z	R	G	B
V1	x1	y1	z1	r1	g1	b1
V2	x2	y2	z2	r2	g2	b2
V3	x3	y3	z3	r3	g3	b3
V4	x4	y4	z4	r4	g4	b4
V5	x5	y5	z5	r5	g5	b5

GEOMETRY + ATTRIBUTES

Tri:	Wedge 1:	Wedge 2:	Wedge 3:
T1	V4	V1	V2
T2	V4	V2	V5
T3	V5	V2	V3

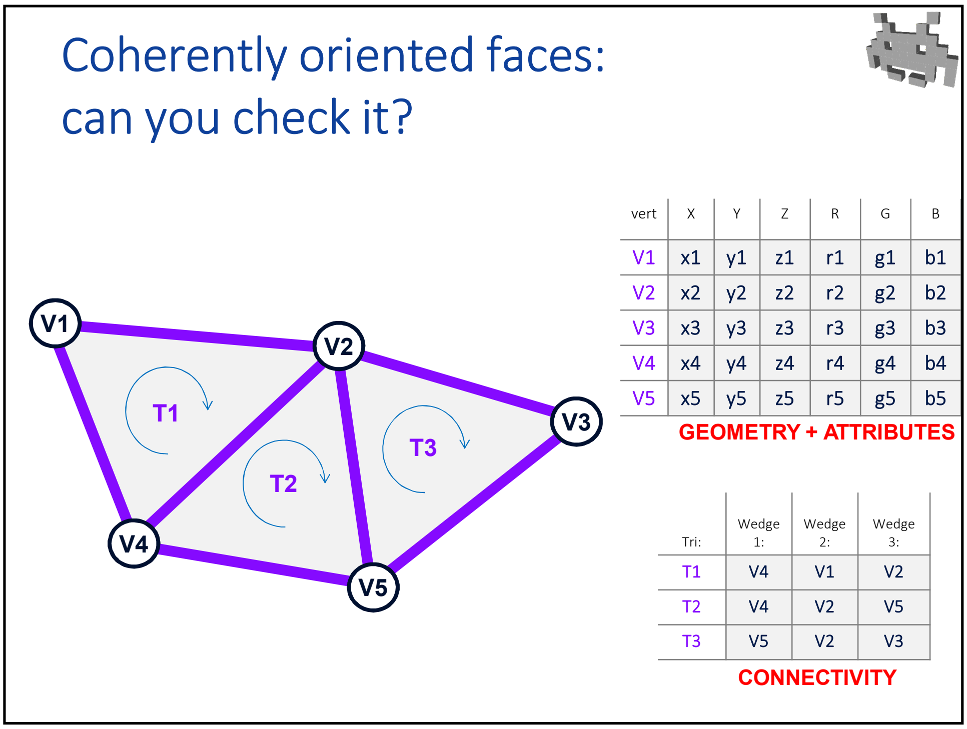
CONNECTIVITY



Compute normals from geometry

- Note:
  the faces orientation must be coherent

Opposite direction, coherent edge



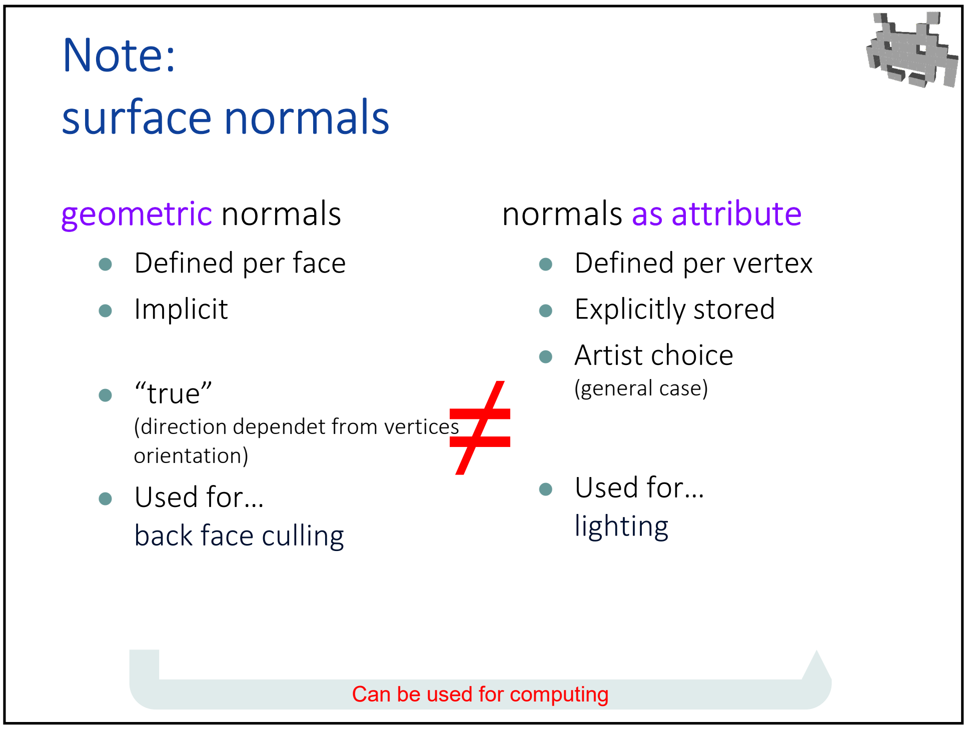
Coherently oriented faces: can you check it?

vert	X	Y	Z	R	G	B
V1	x1	y1	z1	r1	g1	b1
V2	x2	y2	z2	r2	g2	b2
V3	x3	y3	z3	r3	g3	b3
V4	x4	y4	z4	r4	g4	b4
V5	x5	y5	z5	r5	g5	b5

GEOMETRY + ATTRIBUTES

Tri:	Wedge 1:	Wedge 2:	Wedge 3:
T1	V4	V1	V2
T2	V4	V2	V5
T3	V5	V2	V3

CONNECTIVITY



Note: surface normals

geometric normals

- Defined per face
- Implicit
- "true"
  (direction dependet from vertices orientation)
- Used for…
  back face culling

≠

normals as attribute

- Defined per vertex
- Explicitly stored
- Artist choice
  (general case)
- Used for…
  lighting

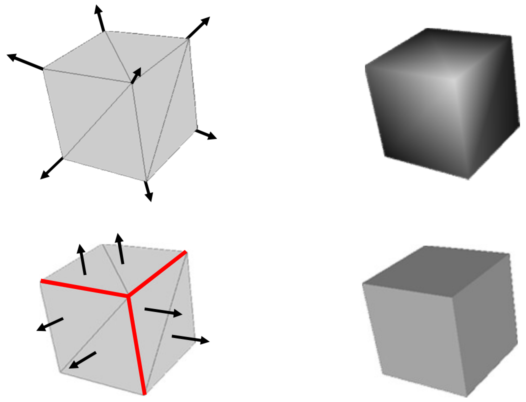
Can be used for computing

Crease edges
(aka “hard edges”)

- Edges of discontinuity of the normals.

No Creases:
(all edges “soft”)

With Creases:
(red edges “hard”)

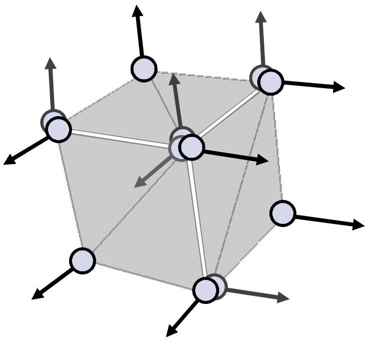


- How to obtain a discontinuity (C_0) in the attributes?

answer:

Vertex seams

- Vertex seam = two coincident vertices in *xyz*
  - (different attributes assigned to each copy)



Ex.: vertex seams
for implementing hard edges

Vertex seams

- Needed for every attribute discontinuity
- Data replication… a necessary evil

vert	X	Y	Z	R	G	B
V1	x1	y1	z1	r1	g1	b1
V2	x2	y2	z2	r2	g2	b2
V3	x3	y3	z3	r3	g3	b3
V4	x4	y4	z4	r4	g4	b4
V5	x5	y5	z5	r5	g5	b5

GEOMETRY + ATTRIBUTES

Tri:	Wedge 1:	Wedge 2:	Wedge 3:
T1	V4	V1	V2
T2	V4	V2	V5
T3	V5	V2	V3

CONNECTIVITY

Mesh processing aka Geometry Processing

Libraries:

- VCG-Lib *(CNR,* 🇮🇹
  - Vision and Computer Graphic Lib
- OpenMesh *(RWTH, de* 🇩🇪
  - + open flipper
- CGAL *(INRIA,* 🇫🇷
  - Computational Geometry Algorithms Library

(all: C++, open-source.)

Common attributes: color

- Useful for:
  - Cheaply add variations to models
  - Bake global lighting
    (e.g. per-vertex ambient occlusion)
  - Dynamic recoloring of meshes
  - …and much more

Common attributes: texture coords

- Text coords dictate how a texture image must cover the mesh
  (see later)
- Set of per-vertex texture coords =
  the "UV-map" of the mesh
- Typically,
  they require discontinuities (*Texture seams*)
  (more so than other attributes)

Common attributes: recap

- Position (mesh *"geometry"*)
- Normal
- Color
- Texture Coords (mesh *"UV-mapping"*)
- Tangent Dirs

for tangent space normal mapping (see texturing, later)

Common attributes: recap

- Position (mesh *"geometry"*)
- Normal
- Color
- Texture Coords (mesh *"UV-mapping"*)
- Tangent Dirs
- Bone assignments (mesh *"skinning"*)

We'll see this during lecture on animation

Common attributes: recap

- Position
- Normal
- Color
- Texture Coords
- Tangent Dirs
- Bone assignments

Tri:	W1:	W2:	W3:
T1			
T2			
T3			

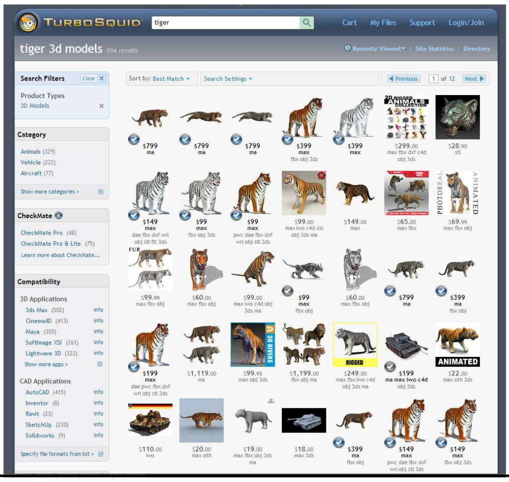
CONNECTIVITY

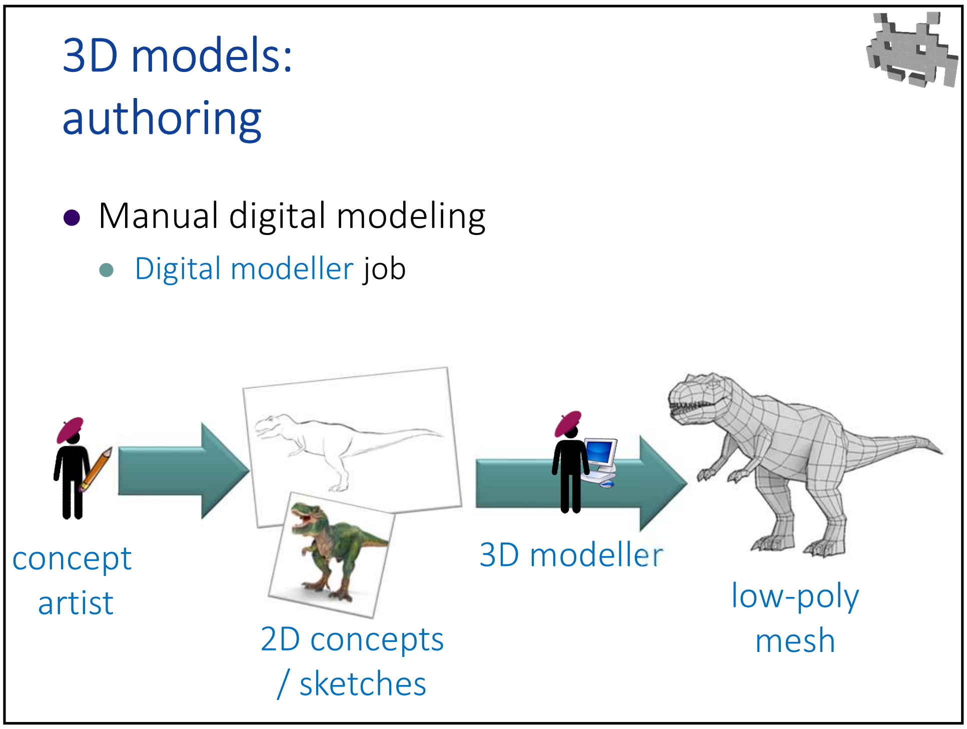
vert	X	Y	Z	Nx	Ny	Nz	R	G	B	A	U	V	Tx	Ty	Tz	Bx	By	Bz
V1																		
V2																		
V3																		
V4																		
V5																		

GEOMETRY + ATTRIBUTES

3D models:
how to obtain them?

- Like any asset, often just bought / off-sourced





3D models: authoring

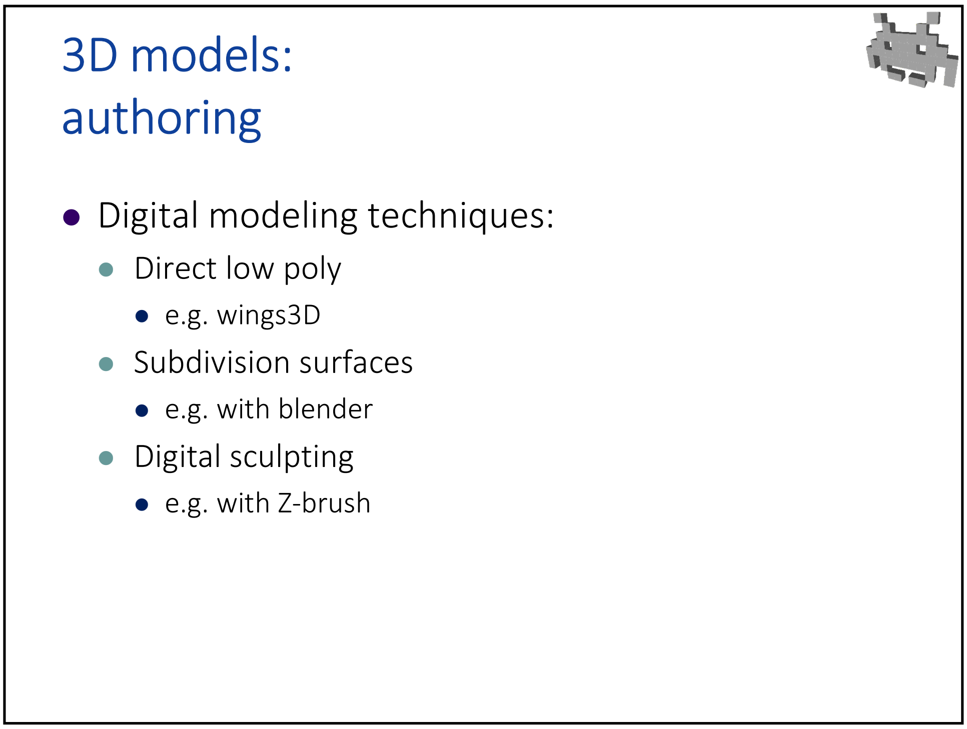
- Manual digital modeling
  - Digital modeller job

concept artist

2D concepts / sketches

3D modeller

low-poly mesh



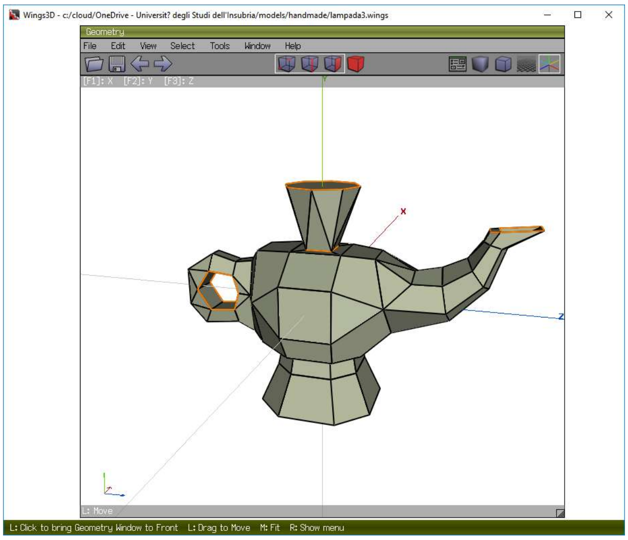
3D models: authoring

- Digital modeling techniques:
  - Direct low poly
    - e.g. wings3D
  - Subdivision surfaces
    - e.g. with blender
  - Digital sculpting
    - e.g. with Z-brush

Mesh editing: generic applications

- 3D Studio Max (autodesk), Maya (autodesk), Cinema4D (maxon) Lightweight 3D (NewTek), Modo (The Foundry), ...
  - generic, powerful, complete
- Blender
  - idem, but open-source and freeware (like: Gimp VS. Adobe Photoshop for 2D images)
- MeshLab
  - open-source, big collection of geometry processing algorithms ...
- AutoCAD (autodesk), SolidWorks (SolidThinking)
  - for CAD
- ZBrush (pixologic), + Sculptris, Mudbox (autodesk)
  - cirtual sculpting metaphore, specialized on manual editing of hi-freq details, bumpmapping, normalmaps...
- Wings3D
  - open-source, small, specialized in low-poly editing, subdivision surfaces
- [Rhinoceros]
  - parametric surfaces (NURBS)
- FragMotion
  - specialized on animated meshes
- + a lot of tools for specific contexts
  - (editing of human models, of architectural interiors, environments, or specific editors for game-engines, etc...)

Low-poly modelling (demo)

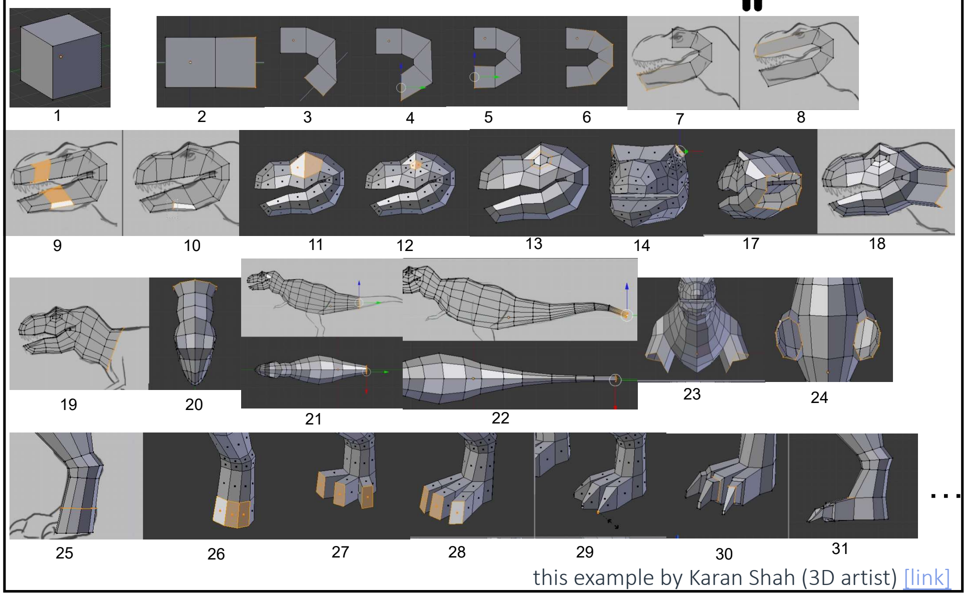


Wings3D - c:/cloud/OneDrive - Universit? degli Studi dell'Insubria/models/handmade/lampada3.wings

L: Click to bring Geometry Window to Front L: Drag to Move M: Fit R: Show menu

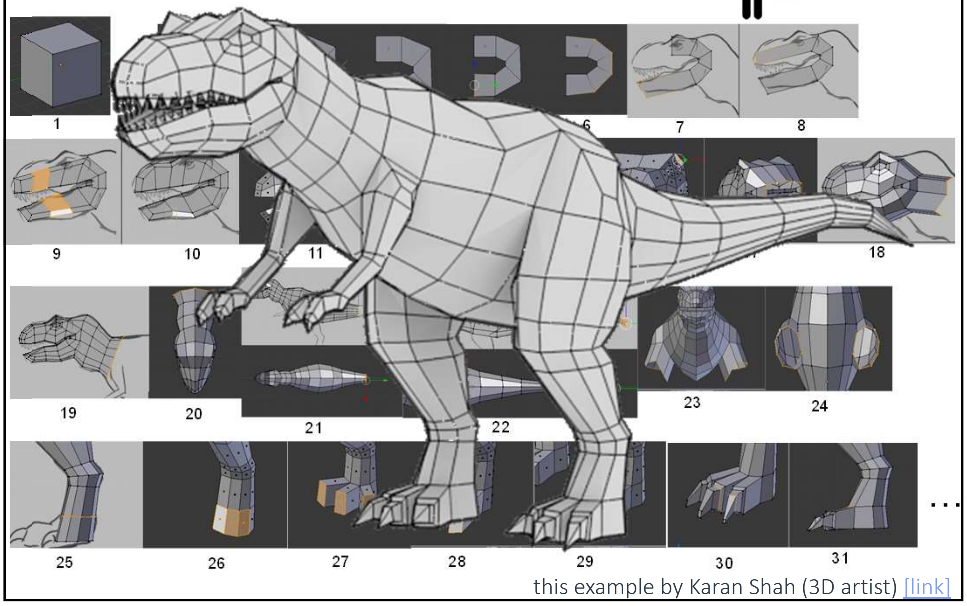
Note: Often during creation, the meshes are polygonal instead of triangle ones. But is simple to decompose any polygon of n>3 edges to (n-2) triangles.
(e.g. just before exporting the asset, or by the game engine, during the import)

Low-poly modelling



this example by Karan Shah (3D artist) [link]

Low-poly modelling



this example by Karan Shah (3D artist) [link]