Game Engines
Master Game Dev
Verona, 2014-2015

Assets: 3D Mesh

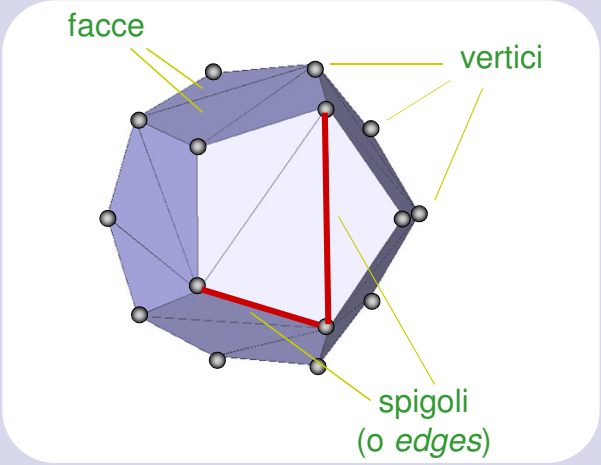
Marco Tarini

I Modelli 3D più comuni: Mesh poligonali

- Di triangoli, o mista (quadrilateri + triangoli)
- Struttura dati per modellare oggetti 3D
  - GPU friendly
  - Risoluzione (potenzialmente) adattiva
  - “Complessità” = numero facce

Mesh triangolare (o mesh simpliciale)

- Un insieme di triangoli adiacenti



Mesh di triangoli

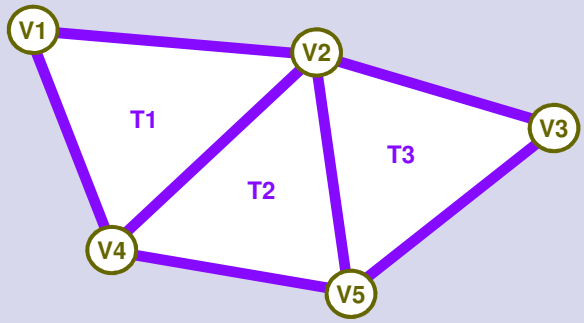
- discretizzazione lineare a tratti di una superfice continua (un “2 manifold”) immersa in R3
- Componenti:
  - **geometria**
    - i vertici, ciascuno con pos (x,y,z)
    - un campionamento della superficie!
  - **connettività** (a volte: “topologia”)
    - come sono connessi i vertici
    - (es.: in una tri-mesh, i triangoli)
  - **attributi**
    - es: colore, materiali, normali, UV, …

Mesh: geometria

- Insieme di posizioni dei vertici
  - Un vettore posizione (x,y,z) per ogni vertice
  - (Spazio Oggetto)

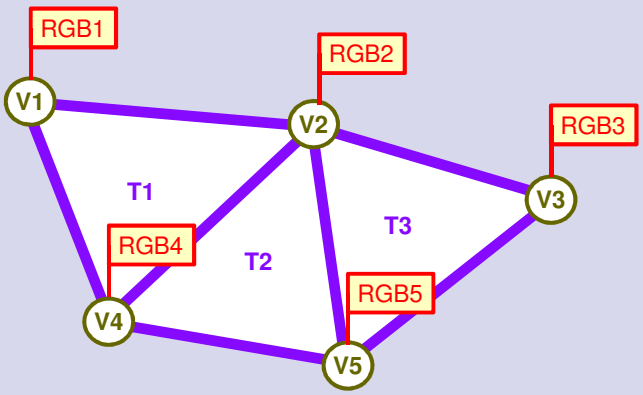
Mesh: connettività (o topologia)

- Triangoli (o quads, o edges…)
  - che connettono fra loro i vertici
  - Come nodi connessi da archi, in un grafo



Mesh: attributi

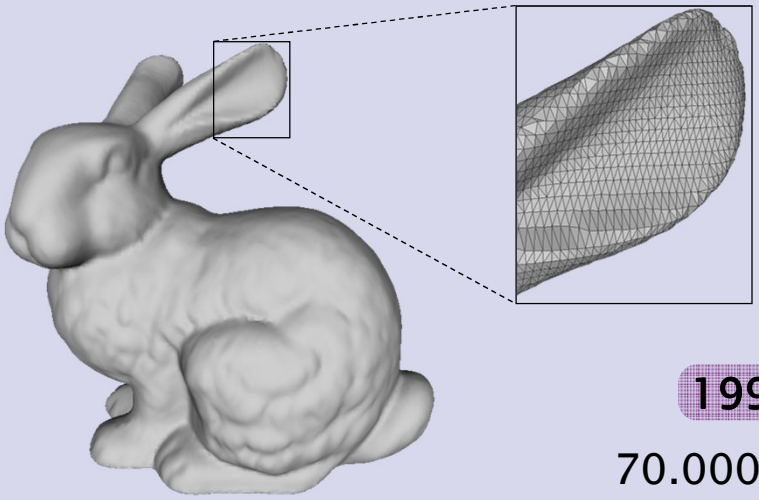
- Quantità che variano sulla superficie
  - Campionati per vertice, interpolati nei poly



Mesh: attributi

- Proprietà che variano sulla superficie
- Memorizzati per vertice
  - (almeno, nei games)
- Attributi più diffusi nei games:
  - Normale
    - per: re-lighting dinamico
  - Colore
    - per: baked lighting (ambient occlusion)
    - per: aggingere varietà (RGB)
  - Coordinate tessitura (“uv mapping”)
    - per: texture mapping
  - Direzioni tangenti
    - per: bump mapping
  - Bone assignment (“rigging” o “skinning”)
    - per: animazioni skeletali

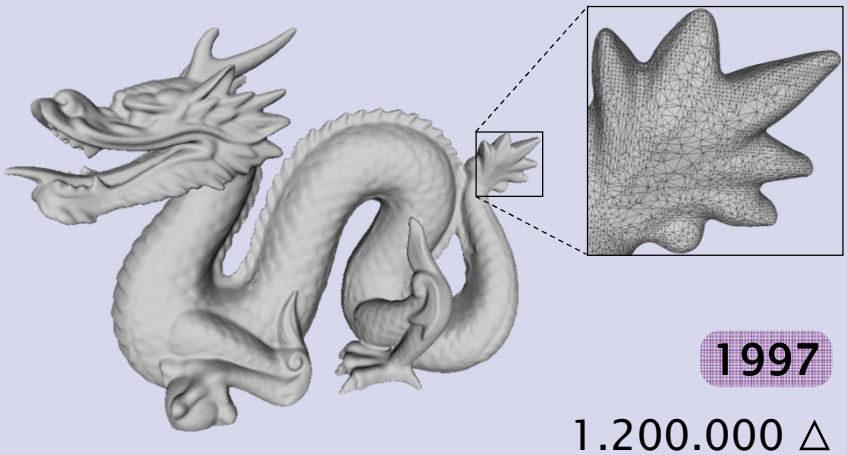
Meshes: complessità crescente



1994

70.000 △

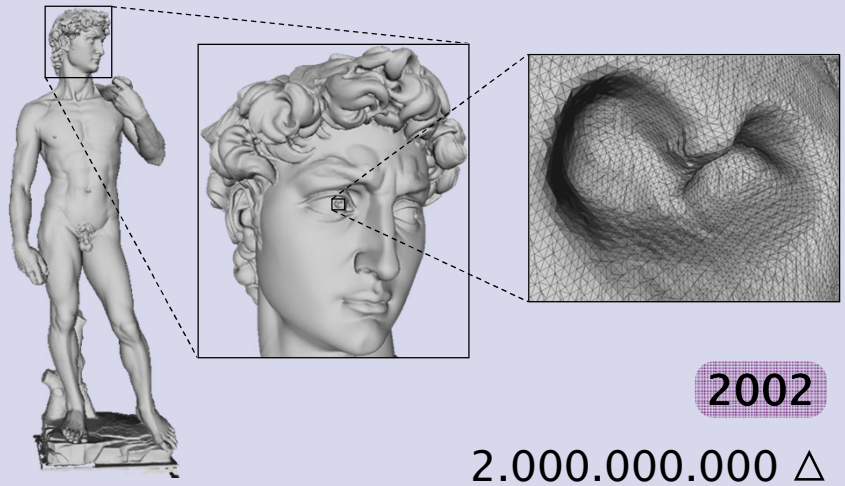
Meshes: complessità crescente



1997

1.200.000 △

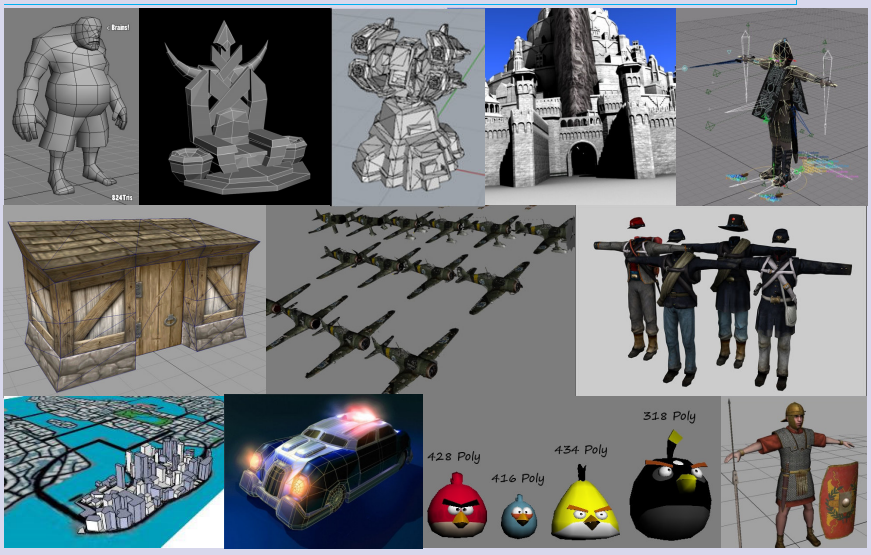
Meshes: complessità crescente



2002

2.000.000.000 △

Low Poly Meshes



Come rappresento una mesh?
(quali strutture dati)

- Modo **indexed** in C++ :

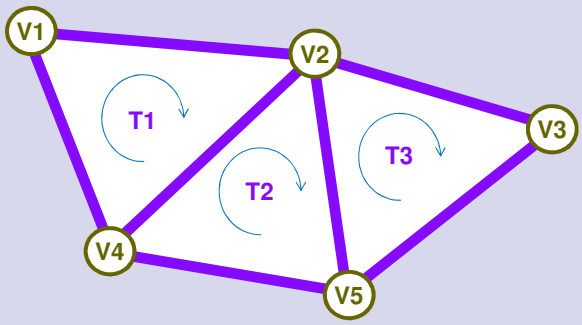
```
class Vertex {
  vec3 pos;
  rgb color;   /* attribute 1 */
  vec3 normal; /* attribute 2 */
};

class Face{
   int vertexIndex[3];
};

class Mesh{
  vector<Vertex> vert; /* geom + attr */
  vector<Face> tris;   /* connettivita' */
};
```

Come rappresento una mesh?
(quali strutture dati)

- **indexed mesh**



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

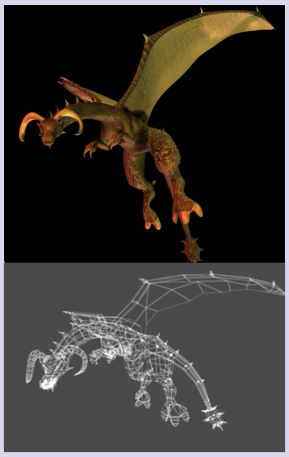
GEOMETRIA + ATTRIBUTI

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

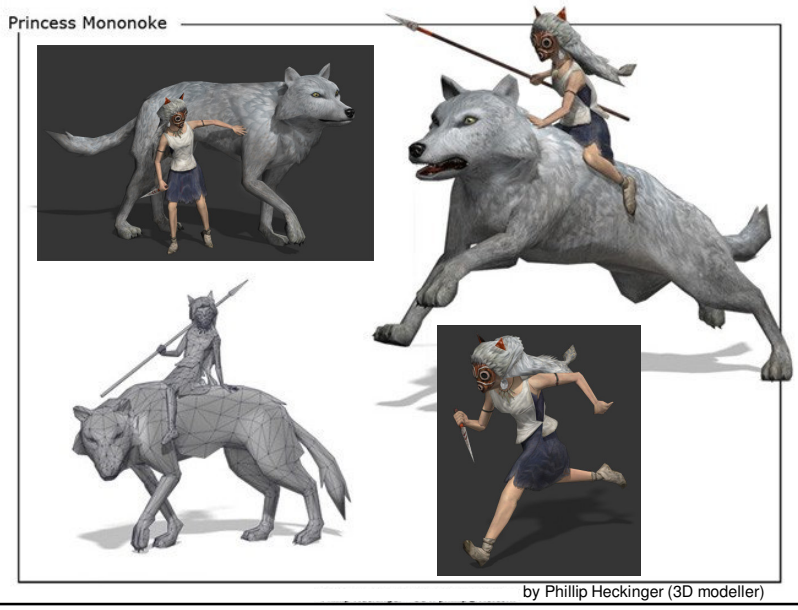
CONNETTIVITA'

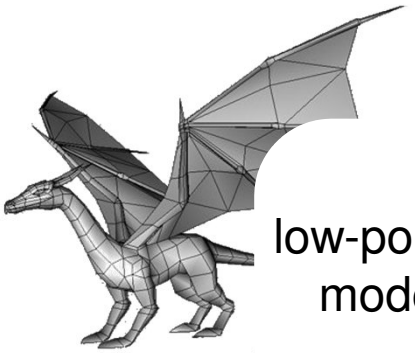
Ma… in ambiente games

- LOW POLY MODELLING!

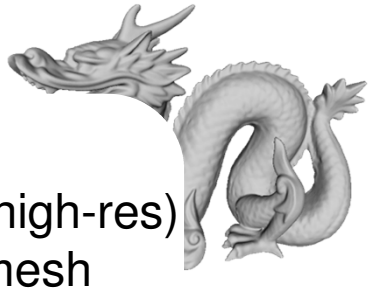


Low-poly modelling

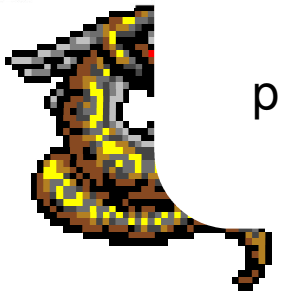




low-poly model : (high-res) mesh



=



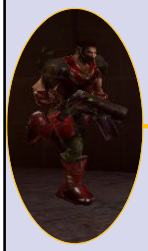
pixel art : (high res) image

Solomons's key (1986, Temco) on Z80

reminder:
per tutti gli '80,
il principale **asset** dei games
è consistito da
sprites / **tilemaps** in **pixel art** ...

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

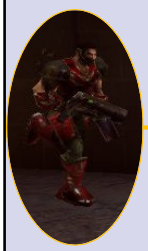
Anche nei games



800 △

Unreal Torunement
(1999)

Anche nei games



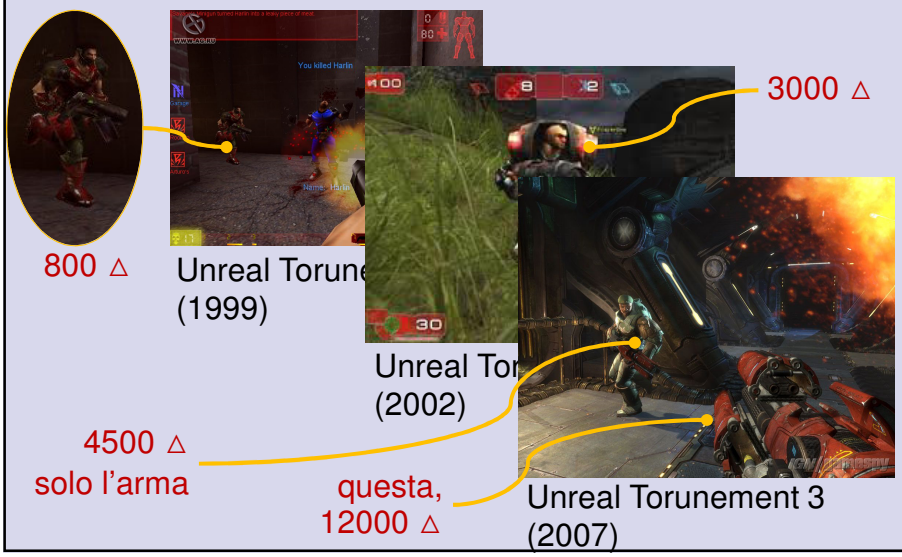
800 △

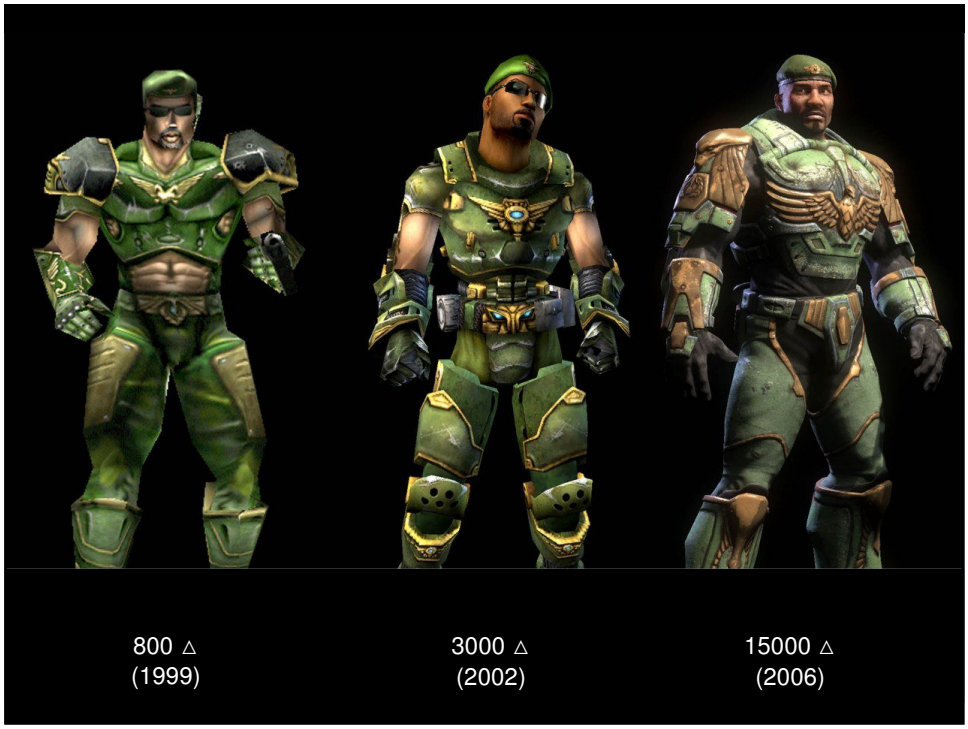
Unreal Torune
(1999)

3000 △

Unreal Torunement 2K3
(2002)

Anche nei games





Anche nei games

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**
  - GPU based
  - (+ animate)

Rendering of a Mesh in a nutshell

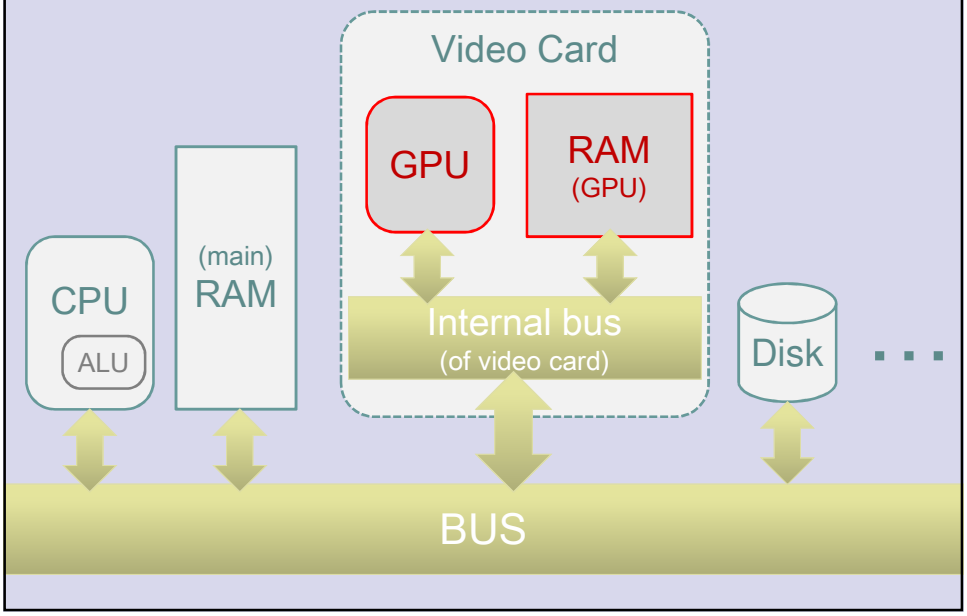
- Load…
  - store all data on **GPU RAM**
    - Geometry + Attributes
    - Connectivity
    - Textures
    - Shaders

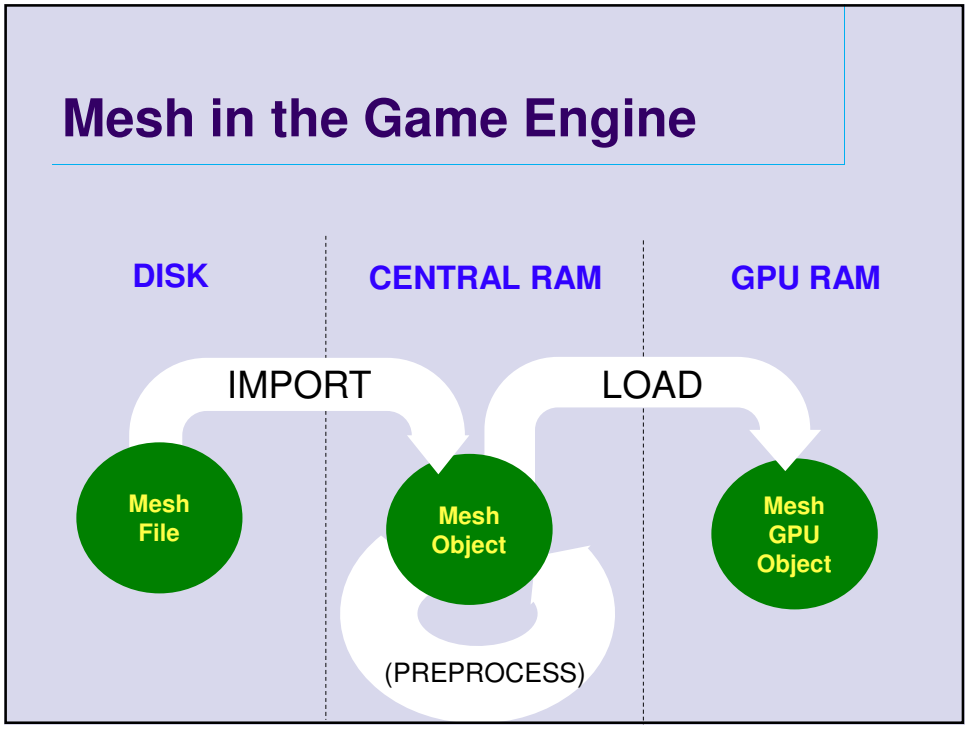
THE MESH

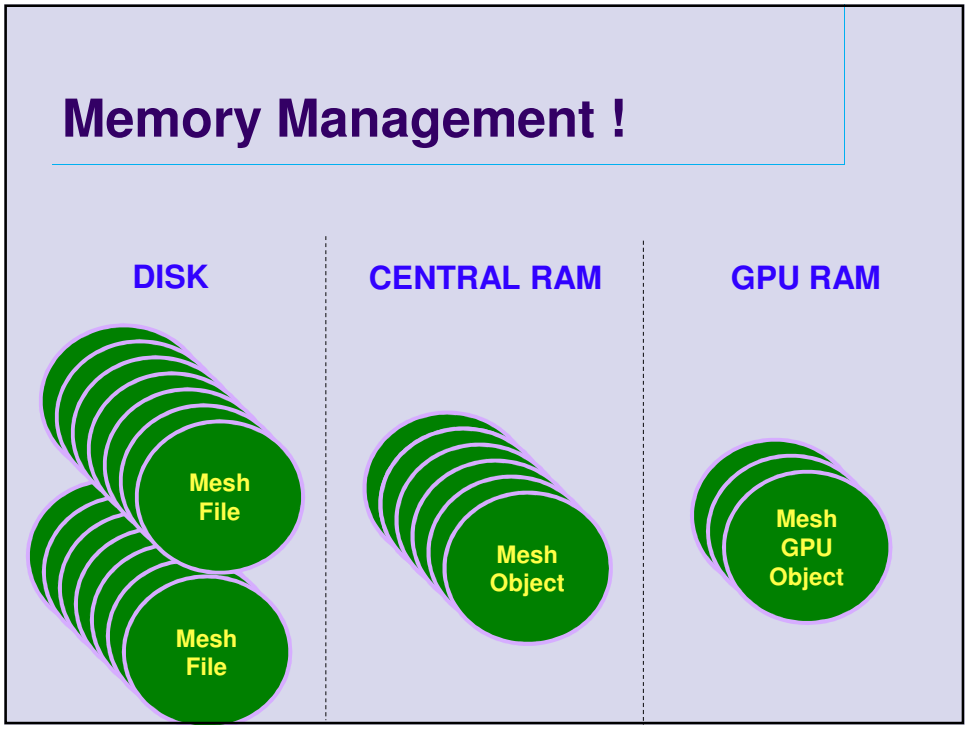
THE MATERIAL

- …and Fire!
  - send the command: *do it* !

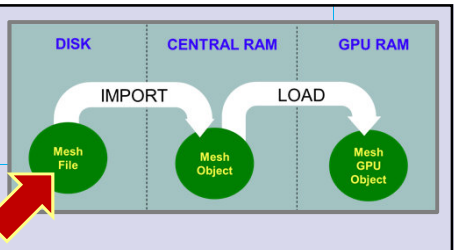
Simplified schema of: “PC + Video Card”




Mesh in the Game Engine
DISK
CENTRAL RAM
GPU RAM
IMPORT
LOAD
Mesh File
Mesh Object
Mesh GPU Object
(PREPROCESS)

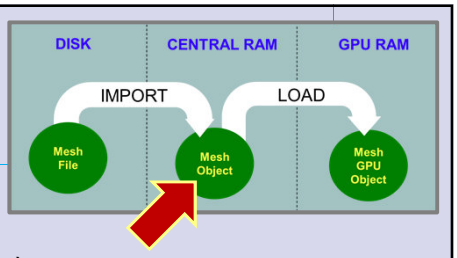

Memory Management !
DISK
CENTRAL RAM
GPU RAM
Mesh File
Mesh File
Mesh Object
Mesh GPU Object

Mesh File



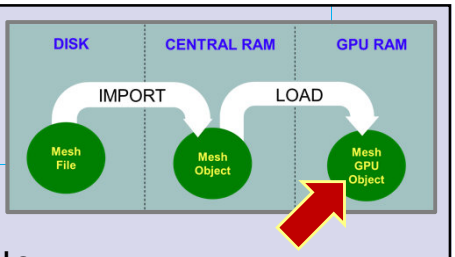
- A file of a given format sitting on the disk
- Choices for the game engine:
  - which formats(s) to import?
    - proprietary, standard…
  - storing which attribute?
- Issues:
  - storage cost
  - loading time

Mesh Object



- 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)

Mesh GPU Object



- VBO / Vertex Arrays / 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 / computatuion

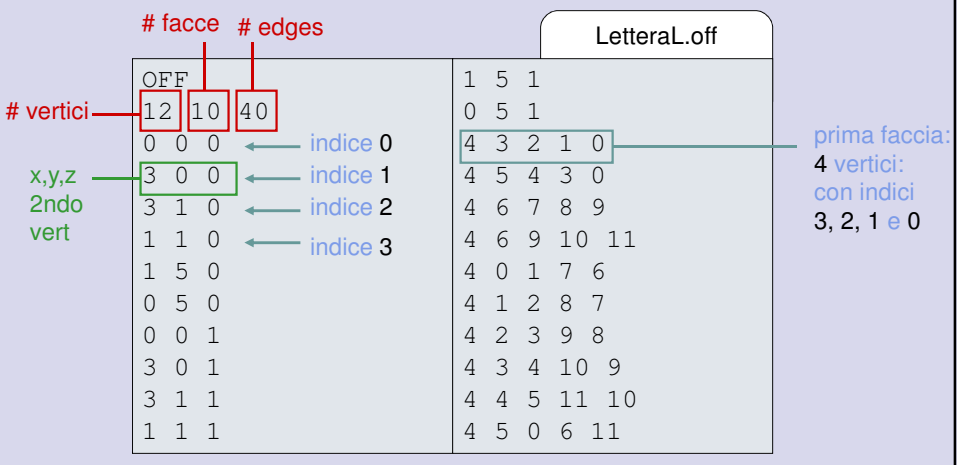
Come rappresento una mesh? (quali strutture dati)

- Una tri-mesh è un insieme di triangoli adiacenti
- Modo **diretto**:
  - un vettore di triangoli
  - e per ogni triangolo tre vertici
  - e per ogni vertice tre coordinate
  - Ma: replicazione dati
    - poco efficiente in spazio
    - oneroso fare updates

Come rappresento una mesh?
(quali strutture dati)

- Modo **indexed**:
  - Geometria: array di vertici
    - in ogni vertice, posizione e attributi
  - Attributi:
    - coi vertici
      - (e.g. campi della classe “vertice”)
  - Connettività: (a volte anche: “topologia”)
    - Array di triangoli
    - Per ogni triangolo:
      - tripletta di **indici** a vertice

Esempio di mesh indexed:
guardiamo dentro un file in formato OFF



Come rappresento una mesh?
(quali strutture dati)

- Modo **indexed** in C++ :

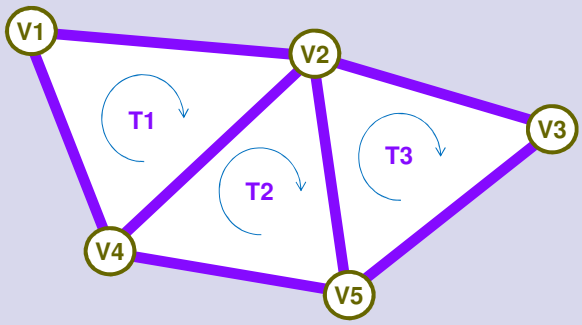
```
class Vertex {
  vec3 pos;
  rgb color;   /* attribute 1 */
  vec3 normal; /* attribute 2 */
};

class Face{
   int vertexIndex[3];
};

class Mesh{
  vector<Vertex> vert; /* geom + attr */
  vector<Face> tris;   /* connettivita' */
};
```

Come rappresento una mesh?
(quali strutture dati)

- **indexed mesh**



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

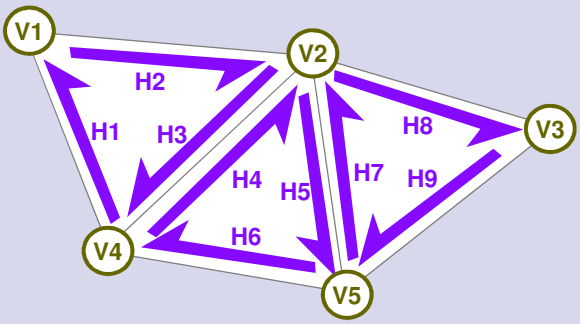
GEOMETRIA + ATTRIBUTI

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

CONNETTIVITA'

Come rappresento una mesh? (quali strutture dati)

- connettività con **half-edges**



Half edge:	Da:	A:	Next:	Oppo-site:
H1	V4	V1	H2	--
H2	V1	V2	H3	--
H3	V2	V4	H1	H4
H4	V4	V2	H5	H3
H5	V2	V5	H6	H7
H6	V5	V4	H4	--
H7	V5	V2	H8	H5
H8	V2	V3	H9	--
H9	V3	V5	H7	--

CONNETTIVITA'

Strutture per connettività a confronto

INDEXED

- HW friendly
  - (directly maps to: vertex arrays, VBO)
- Navigazione… complicata
  - richiede strutture dati ulteriori ("di adiacenza")
- Ok per: mesh "pure" (di soli tri, o, al max, di soli quad)
- Compatta: 3 indici x tri
- *> adatta per rendering*

mesh come asset

HALF-EDGES

- Rendering… complicato
- Navigazione semplice
  - es: x trovare tutti i vertici nella "stella 1" di un vertice…
- Ok *anche* per: polygonal mesh (poligoni misti a piacere)
- Prolissa: 12 indici per tri
- *> adatta a mesh processing*

mesh nei tools, maybe?

Mesh processing aka Geometry Processing

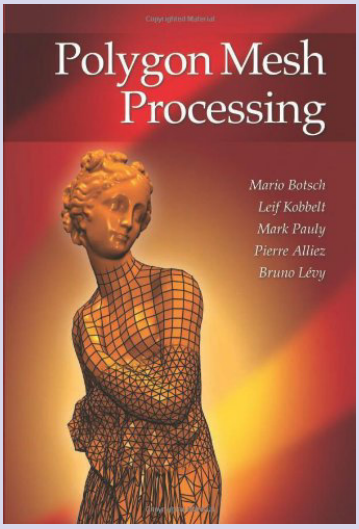
Librerie:

- VCG-Lib *(CNR,)*
  - Vision and Computer Graphic Lib
- OpenMesh *(RWTH,)*
  - + open flipper
- CGAL *(INRIA,)*
  - Computational Geometry Algorithms Library

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

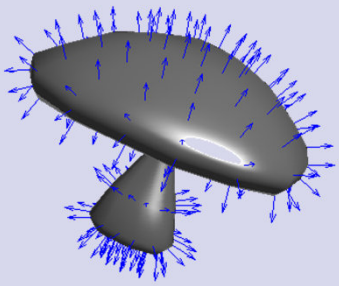
Mesh processing aka Geometry Processing

- Un buon manuale x programmare mesh processing:



Attributi comuni: normale

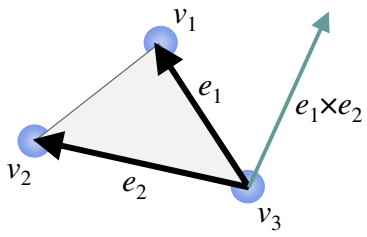
- Vettore direzione unitario
- Orientamento della superficie
- Usato per il lighting
- A volte, calcolate automaticamente dalla geometria
  …
- Ma l'artista decide quali edges sono *soft* e quali *hard*



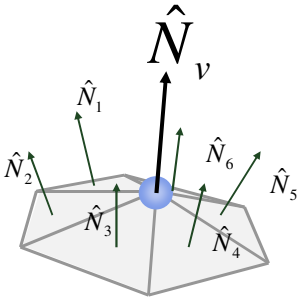
Calcolo normali dalla geometria

- Geometria
  =(1)=>
  normali x faccia
  =(2)=>
  normali x vertice

(1)



(2)

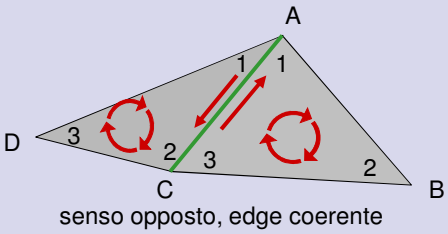


$$N = \hat{N}_1 + \hat{N}_2 + \ldots + \hat{N}_n$$

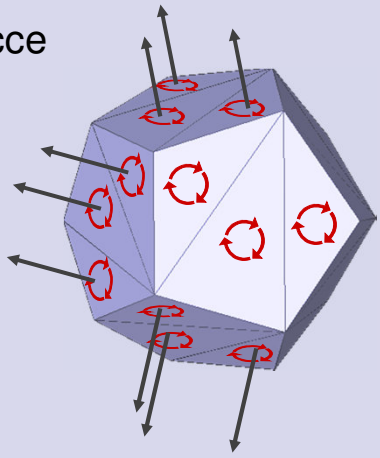
$$\hat{N} = \frac{N}{|N|}$$

Calcolo normali dalla geometria

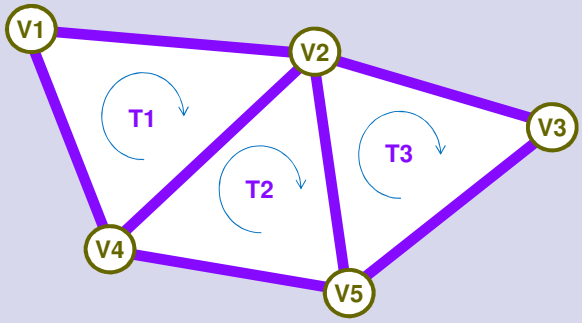
- Nota:
  l'**orientamento** delle facce deve essere **coerente**



senso opposto, edge coerente



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

GEOMETRIA + ATTRIBUTI

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

CONNETTIVITA'

Nota: normali alla superficie

normali **geometriche**

- definite per faccia
- implicite
- "vere" (verso dipende da orientamento vertici in faccia)
- usate per… back face culling

≠

normali **come attributo**

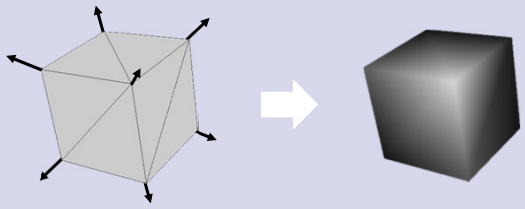
- definite per vertice
- esplicitamente memorizzate
- arbitrio dell'artista (nel caso generale)
- usate per… lighting

si *possono* usare come modo per calcolare

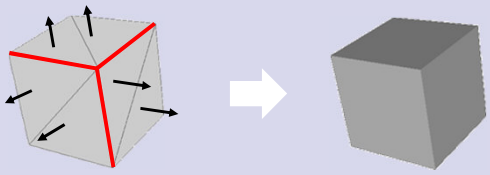
Crease edges (aka "hard edges")

- Edges di discontinuità delle normali.

No Creases: (all edges "soft")



With Creases: (red edges "hard")

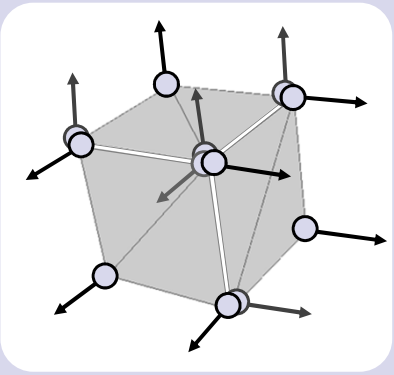


- Come si ottiene una **discontinuità** (C_0) negli attributi**?**

risposta:

Vertex seams

- Vertex seam = due vertici coincidenti in *xyz*
  - (attributi diversi assegnati ad ogni copia)



es: vertex seams
per implementare hard edges

Vertex seams

- Necessari per ogni discontinuità di attributo
- Replicazione dati… un male necessario)

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

GEOMETRIA + ATTRIBUTI

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

CONNETTIVITA'

Attributi Comuni: colore

- Utile per:
  - Differenziare modelli
  - Baked global lighting
    (per vertex ambient occlusion)
  - …e molto altro

Attributi Comuni: texture coords

- Legano la mesh ad un'immagine tessitura
  (vedremo presto)
- Tipicamente,
  necessitano *Texture seams*
  (piu' di ogni altro attributo)

Attributi Comuni: recap

- Posizione ("***geometria***" della mesh)
- Normale
- Colore
- Texture Coords ("***UV-mapping***" della mesh)
- Tangent Dirs

per tangent space
narmal mapping
(vedi texturing, dopo)

Attributi Comuni: recap

- Posizione ("***geometria***" della mesh)
- Normale
- Colore
- Texture Coords ("***UV-mapping***" della mesh)
- Tangent Dirs
- Bone assignments ("***skinning***" della mesh)

vedremo nella lezione
sull animazione

Attributi Comuni: recap

- Posizione
- Normale
- Colore
- Texture Coords
- Tangent Dirs
- Bone assignments

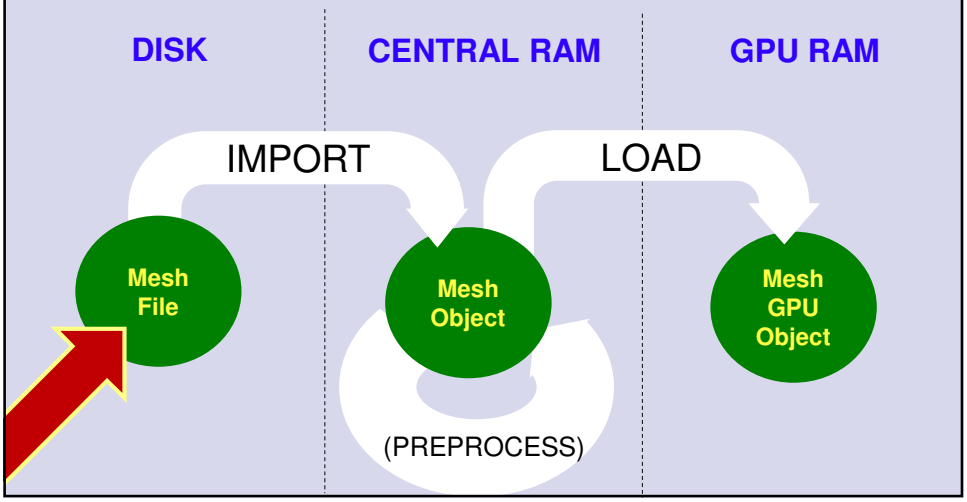
Tri:	W1:	W2:	W3:
T1			
T2			
T3			

CONNETTIVITA'

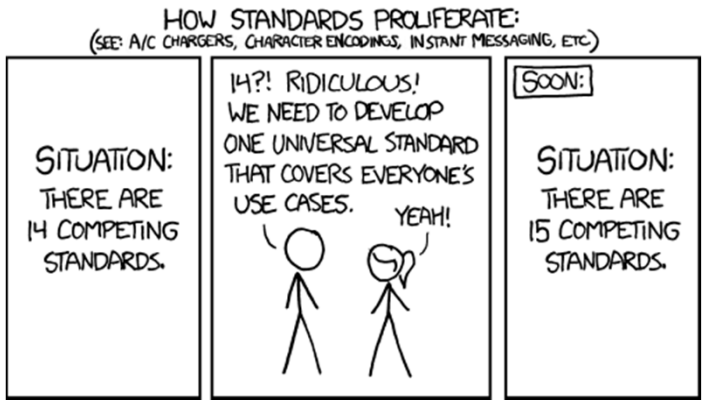
vert	X	Y	Z	Nx	Ny	Nz	R	G	B	A	U	V	Tx	Ty	Tz	Bx	By	Bz
V1																		
V2																		
V3																		
V4																		
V5																		

GEOMETRIA + ATTRIBUTI

Life of a Mesh *(and, basically, any 3D assets)* in a Game Engine



Formati files per mesh …



(da **xkcd.com**)

Formati files per mesh
(una Torre di Babele!)

- 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...

Formati file per mesh più usati nei games

più diffusi

meno diffusi

.OBJ (wavefront)
- ☺ max diffusione
- ☺ indexed, normali , uv-mapping
- ☹ no colori (solo indice materiale x faccia)
- ☹ no skinning o ani

.SMD (VALVE)
- ☺ animazioni scheletali + skinning
- ☺ normali , uv-mapping
- ☹ no indexed!
- ☹ no colori

.MD3 (Quake, IDsoft)
- ☺ vertex animations, normali
- ☹ no colori

.PLY (cyberware)
- ☺ customizzabile
- ☹ “accademico”

.3DS (AUTODESK.)
- ☺ si: colori, uv-mapping, indexed, materiali, tessiture…
- ☹ no: normali
- ☹ limite al numero di vert (64K)

.COLLADA (KHRONOS GROUP)
- ☺ completissimo
- ☺ nato apposta per essere interscambio
- ☺ open standard
- ☹ quasi impossibile da parsare completamente

.FBX (AUTODESK.)
- ☺ completo, comprese animations
- ☹ complesso, difficile da parsare

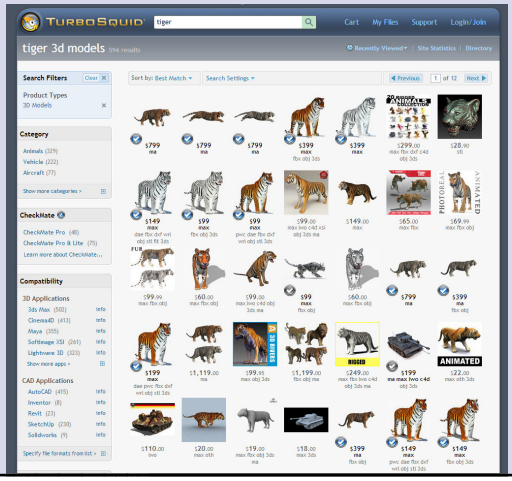
.MA / .MB (AUTODESK.)
- ☺ completo, comprese animations
- ☹ complesso, difficile da parsare

semplici

complessi

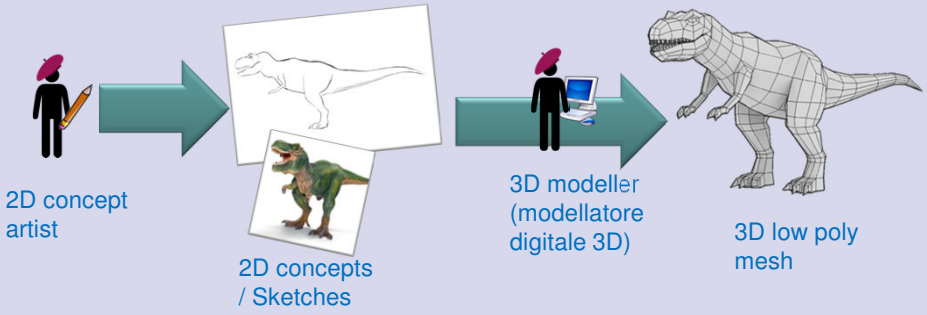
Modelli 3D: come ottenerli

- Come tutti gli asset, possible aquistarli



Modelli 3D: come ottenerli

- Modellazione digitale manuale
  - Lavoro dei modellatori digitali



Tecniche di modellazione digitale di modelli 3D

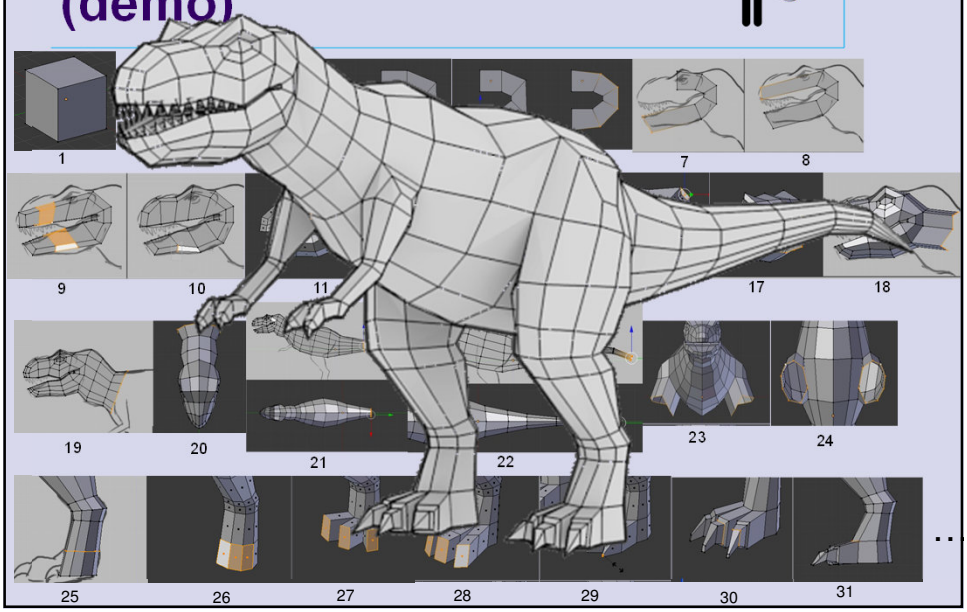
- Tecniche:
  - Low poly diretta
    - e.g. wings3D
  - Subdivision surfaces
    - e.g. con blender
  - Digital sculpting
    - e.g. con Z-brush

Mesh editing: applicativi generici

- **3D Studio Max** (autodesk) , **Maya** (autodesk) , **Cinema4D** (maxon) **Lightweight 3D** (NewTek), **Modo** (The Foundry) , …
  - generici, potenti, completi
- **Blender**
  - idem, ma open-source e freeware (simile a: Gimp VS. Adobe Photoshop per 2D images)
- **MeshLab**
  - open-source, grande collezione algoritmi di geometry processing …
- **AutoCAD** (autodesk), **SolidWorks** (SolidThinking)
  - per CAD
- **ZBrush** (pixologic), + **Sculptris** , **Mudbox** (autodesk)
  - metafora scultura virtuale, specializzato in ritocco manuale dettagli hi-freq, bumpmapping, normalmaps…
- **Wings3D**
  - open-source, piccolo, specializzato in low-poly editing, subdivision surfaces
- **[Rhinoceros]**
  - parametric surfaces (NURBS)
- **FragMotion**
  - specializzato per mesh animate
- …
- + moltissimi strumenti per contesti specifici
  - (editing di umani, di interni architetturali, di paesaggi, o editor specifici per game-engines, etc...)

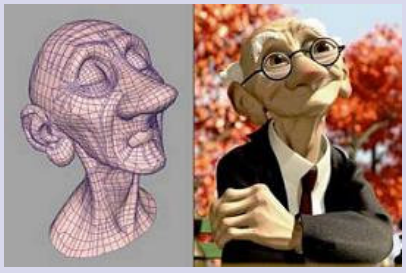
Low poly diretta (demo)

Low poly diretta (demo)



Tecniche di modellazione digitale di modelli 3D

- Subdivision surfaces
  - Raffinamento progressivo della mesh da lowest res ➔ hi res
  - Ottimo per oggetti dall'aspetto smooth, organico e "pulito"



Superfici di suddivisione

- Modo molto diffuso per costruire mesh
  - 1: fare mesh di controllo
    - a bassa risoluzione
    - "a mano"
  - 2: raffinarla automaticamente
    - iterativamente
    - (ad ogni interazione si aggiungono facce e vertici)
- molti schemi matematici differenti
  - con diverse peoprietà

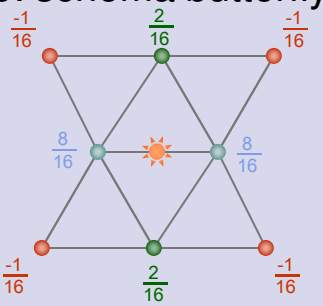
Superfici di suddivisione

- Esempio: schema butterfly (per mesh triangolari)
  - e' uno degli schemi 1=>4
    (in un passo di suddivisione, da ogni triangolo se ne ottengono 4)
    (aggiunta di un vertice per ogni edge)

- MA... quali coordinate assegnare al nuovo vertice?
  Ogni schema di suddivisone ha la sua formula. Ad esempio...

Superfici di suddivisione

- Esempio: schema butterfly



$$POS(\odot) = \frac{8}{16}(POS(\bullet) + POS(\bullet)) + \frac{2}{16}(POS(\bullet) + POS(\bullet)) + \frac{-1}{16}(POS(\bullet) + POS(\bullet) + POS(\bullet) + POS(\bullet))$$

Superfici di suddivisione

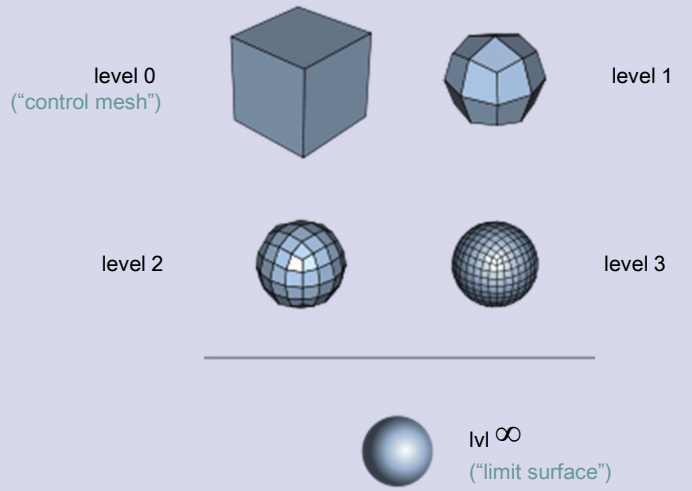
Ad ogni passo di suddivisione

- (x,y,z) dei nuovi vertici inseriti
  - formula (estrapolazione dei vicini)
- (x,y,z) dei vecchi vertici
  - si tiene la vecchia pos (schemi “interpolativi”)

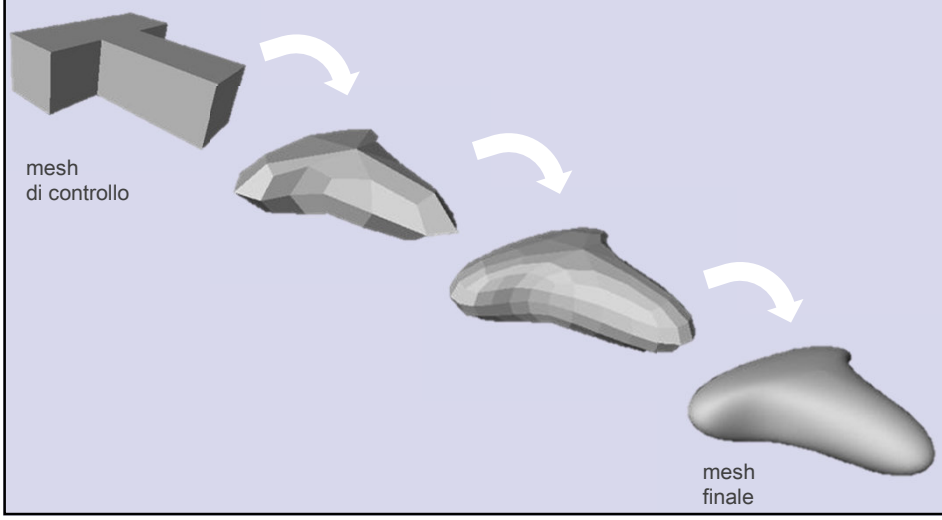
oppure

  - formula (estrapolazione) (schemi “approssimativi”)

Esempio: con schema Catmull-Clark



Superfici di suddivisione



Superfici di suddivisione

Anche iterativamente:

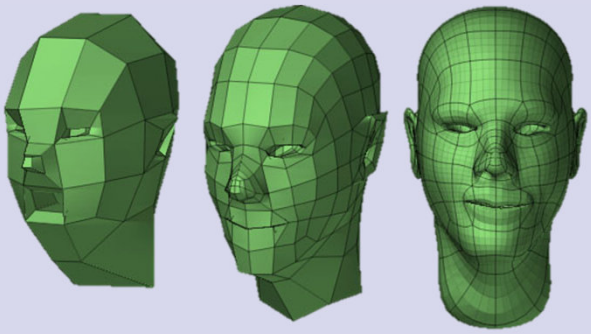
1- Modellare “control mesh”
(editing manuale)

2- Suddivisione
(un passo)

3- Ritocco!
(editing manuale)

4- Goto 2
(fino a raggiungimento risultato voluto alla risuolzione voluta)

DEMO!



Molti schemi…

- Catmull-Clark
- Doo-Sabin
- Loop
- sqrt(3)
- Butterfly
- Mid-edge
- ….

recente aumento di popolarità (GPU friendliness)

Differenze fra gli schemi di suddivisione

- interpolativi VS approssimativi
- solo triangoli, solo quads, qualunque cosa
- incremento complessità
  - (per ogni passo di suddivisione)
- proprietà della limit surface
  - (esistenza, smoothness)
- esistenza forma chiusa per la limit surface
  - (esatta o approssimata)
- …

Tecniche di modellazione digitale di modelli 3D

- Tecniche:
  - Low poly diretta
    - e.g. wings3D
  - Subdivision surfaces
    - e.g. con blender
  - Digital sculpting
    - e.g. con Z-brush

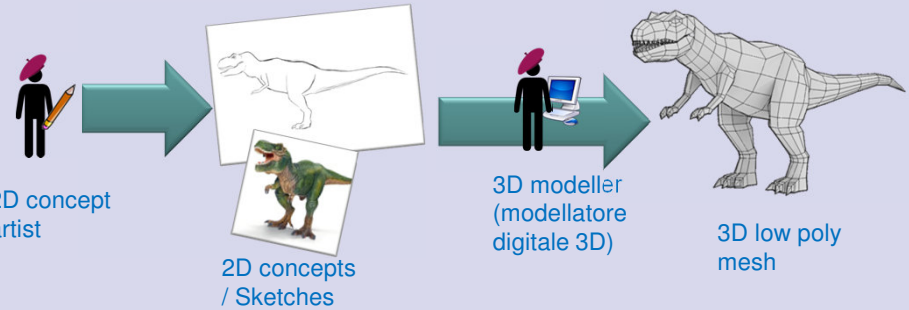
DEMO

Digital Sculpting

cisel
(scalpello)

Modelli 3D: come ottenerli

- Modellazione digitale manuale
  - Lavoro dei modellatori digitali



2D concept artist

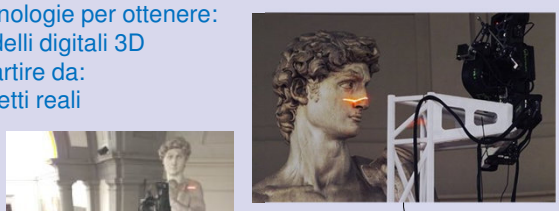
2D concepts / Sketches

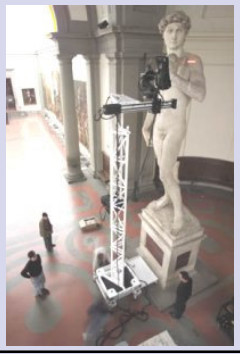
3D modeller (modellatore digitale 3D)

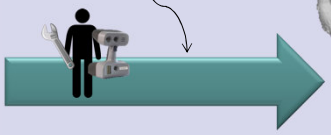
3D low poly mesh

Modelli 3D: come ottenerli

- Attraverso 3D scanning
  - Tecnologie per ottenere: modelli digitali 3D a partire da: oggetti reali



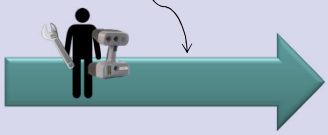




3D scanning (e.g. laser scanning)

Modelli 3D: come ottenerli

- Attraverso 3D scanning
  - Tecnologie per ottenere: modelli digitali 3D a partire da: oggetti reali



3D scanning (e.g. contact scanning)

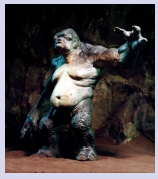
Modelli 3D: come ottenerli

- 3D scanning
  - A.k.a. *automatic 3D model acquisition*
  - Molte tecnologie diverse
    - Laser scanners
    - Time of flight
    - Structured light (kinect)
    - …
  - Caratteristiche diverse
    - Qualità risultati
      - Rumore / risoluzione
    - Automatismo
    - Invasività
      - Markers? Powder?
    - Real time? (kinect)
    - Costo
    - Dimensione massima oggetti
      - (full body scanner?)

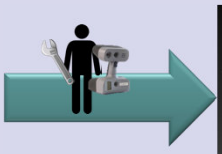
Modelli 3D: come ottenerli

- Attraverso 3D scanning

Scultore (fisico)



Modello reale

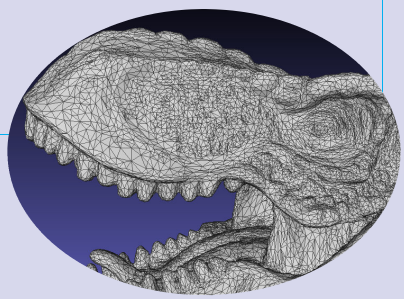


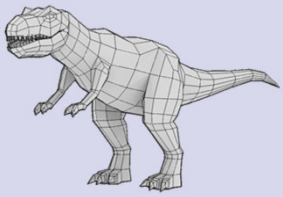
3D scanning

Hi res model

Modelli 3D: come ottenerli

PERFECT for games!
(much easier to: animate, re-edit, uvmap, …)





vs

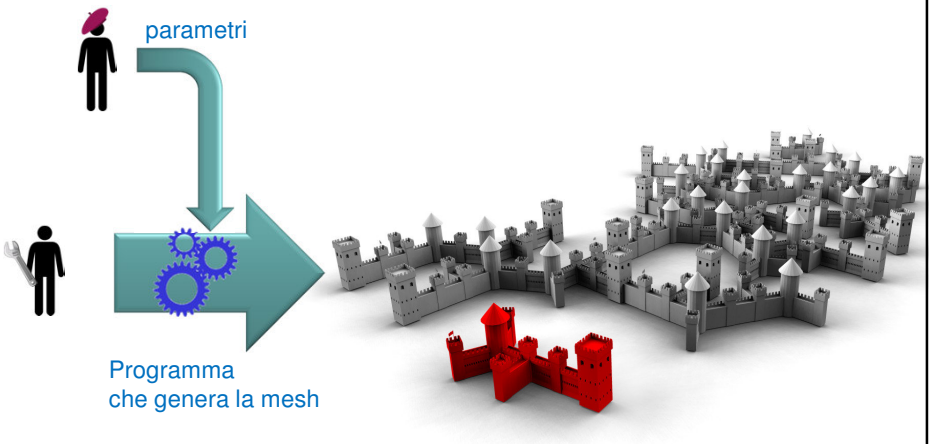
Dino, scanned by artec3d

(artistic)
manually edited
3D low poly mesh

(scanned)
hi res model
30000 tri

Modelli 3D: come ottenerli

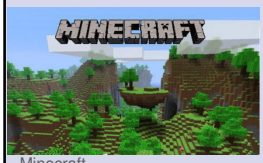
- Modellazione procedurale



Parentesi:

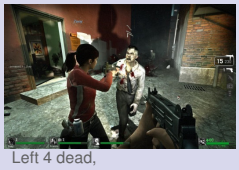
Procedural generation: ottimo per games

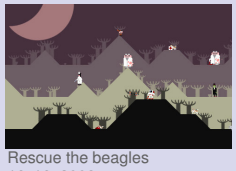
- Concetto: invece di avere un asset, avere un programma che lo crea dinamicamente
  - Modellazione procedurale
  - AI procedurali, boss procedurali…
  - Livelli procedurali
  - Terreni procedurali
  - Musica procedurale
  - Scene procedurali



Minecraft, Mojang, 2009

Elite, Acornsoft, 1984


Left 4 dead, Valve, 2008

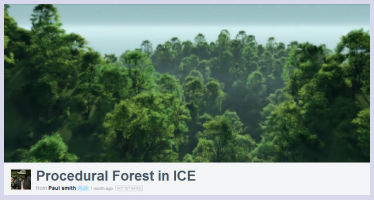

Rescue the beagles 16x16, 2008

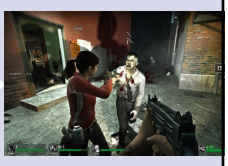
- Vantaggi: varietà, no RAM, …

Parentesi:

Procedural generation: ottimo per games

- Concetto: invece di avere un asset, avere un programma che lo crea dinamicamente
  - Modellazione procedurale
  - AI procedurali, boss procedurali…
  - Livelli procedurali
  - Terreni procedurali
  - Musica procedurale
  - Scene procedurali

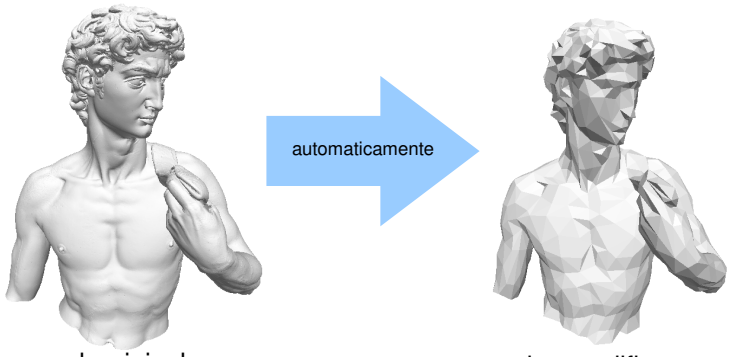



Elite, 1984

- Vantaggi: varietà, no RAM, …

Semplificazione automatica di modelli 3D

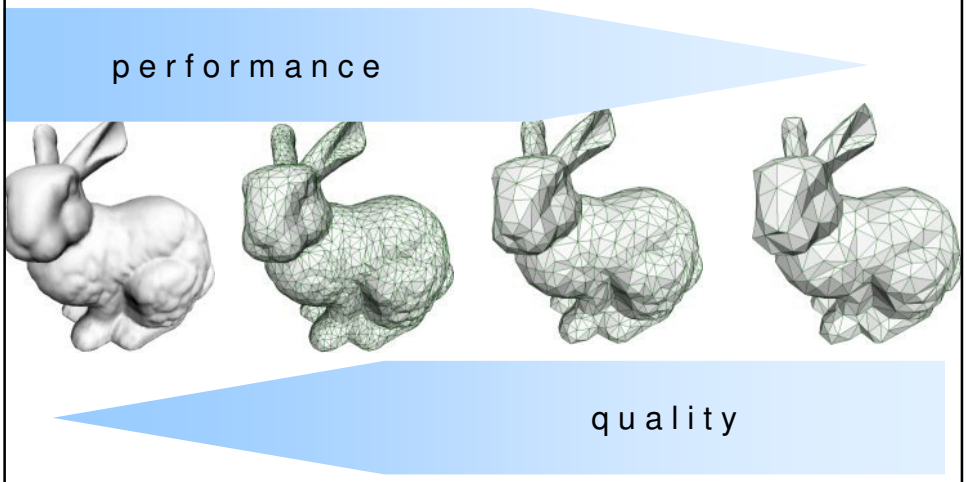
- parametri:
  - un errore massimo
  - o un numero di facce obiettivo



mesh originale
500K triangoli

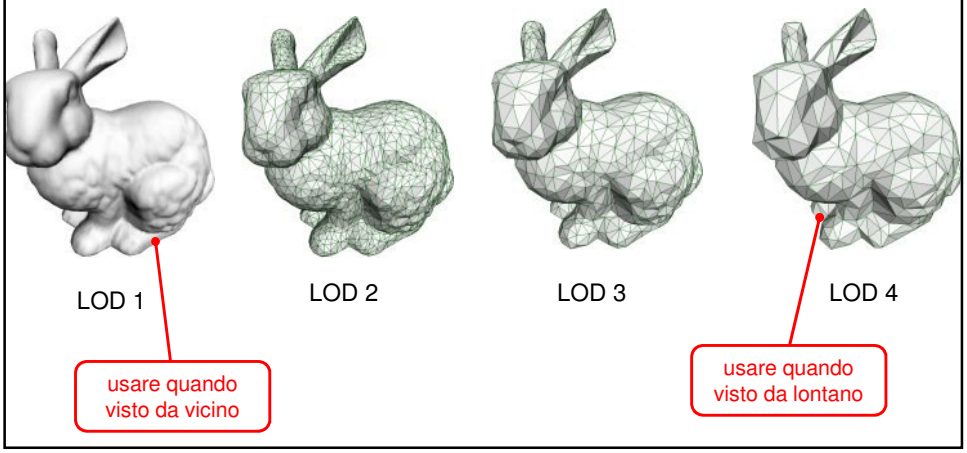
mesh semplificata
2K triangles

Semplificazione automatica di modelli 3D



Semplificazione automatica

Una piramide di Livelli di Dettaglio



LoD pyramid

Demo

Semplificazione automatica

- Molte tecniche diverse
  - Adattive oppure no
    - usare piu' triangoli dove c'e' bisogno (es non nelle zone cmq piatte)
    - oppure no
  - Errore massimo introdotto:
    - misurato e/o limitato
    - oppure no
  - Topologia:
    - mantenuta
    - oppure no
  - Streaming
    - Possibile
    - Oppure no
  - ...

Mesh: task tipici nella game industry

- Semplificazione automatica
  - LOD construction
- Light baking
  - Precomputazione Luce
  - Tipico esempio: Ambient Occlusion
- U-V mapping
  - parametrizzazione
- Texturing
  - creazione tessiture di vario tipo
- Rigging / Animation
  - linear blend skinning

Una classe di tool utili: attribute transfer

- Attribute transfer
  - Da mesh A a mesh B
  - Retargeting di:
    - Animazioni, UV-mapping, tessiture, …