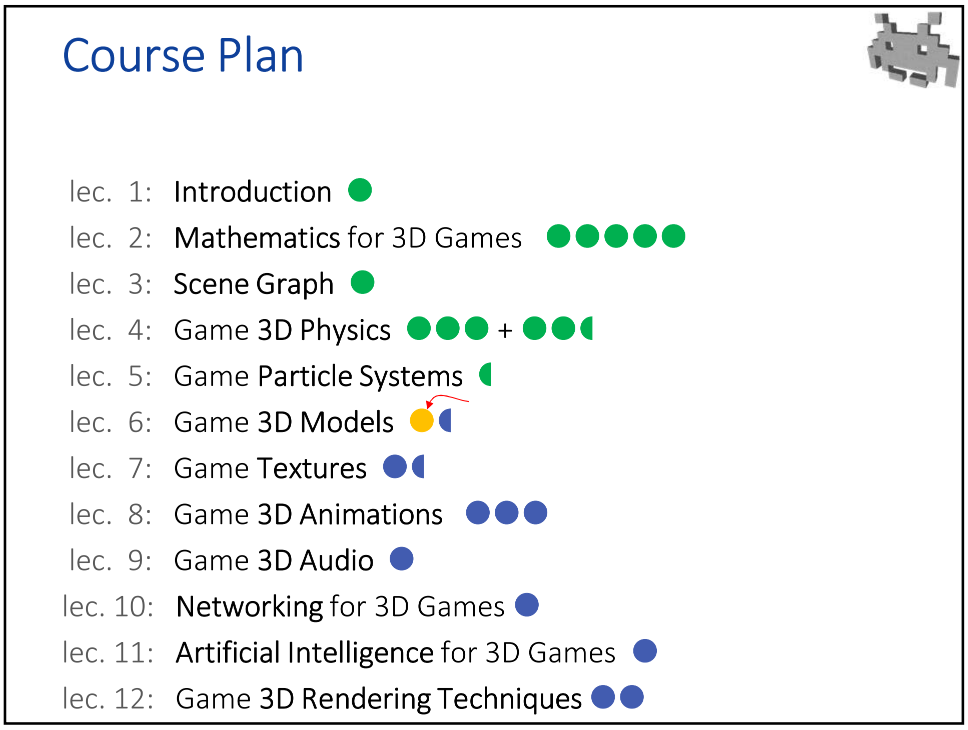
3D video games

Models for Games

Marco Tarini

1



Course Plan

2

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 ...

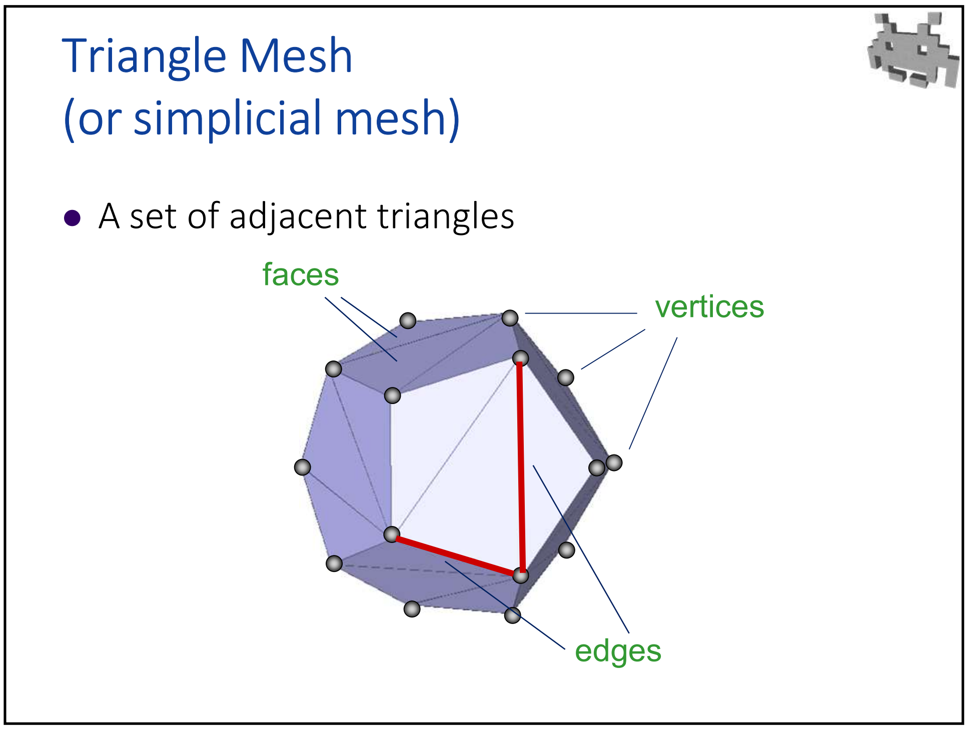
Metal Slug (1996, Nazca Copr), on Neo Geo (SNK)

3

Triangle Meshes
The visual appearance of 3D objects

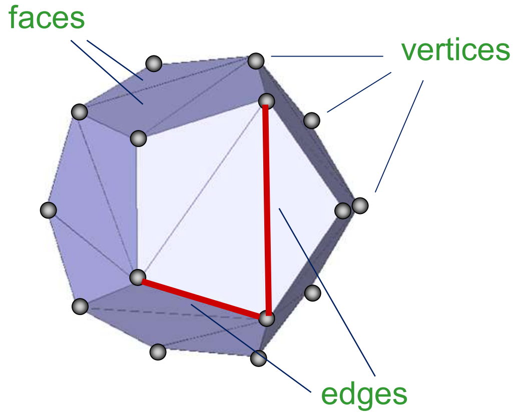
- Data structure for modelling 3D objects
  - GPU friendly
  - Resolution = number of faces
  - (Potentially) Adaptive resolution
- Used in games to represent the visual appearance of 3D objects
  - at least, the ones which can be represented by their surface
  - most solid objects (rigid or not)
- Mathematically: a piecewise linear surface
  - a bunch of surface samples "vertices"
    connected by a set of triangular "faces"
    attached side to side by "edges"

4

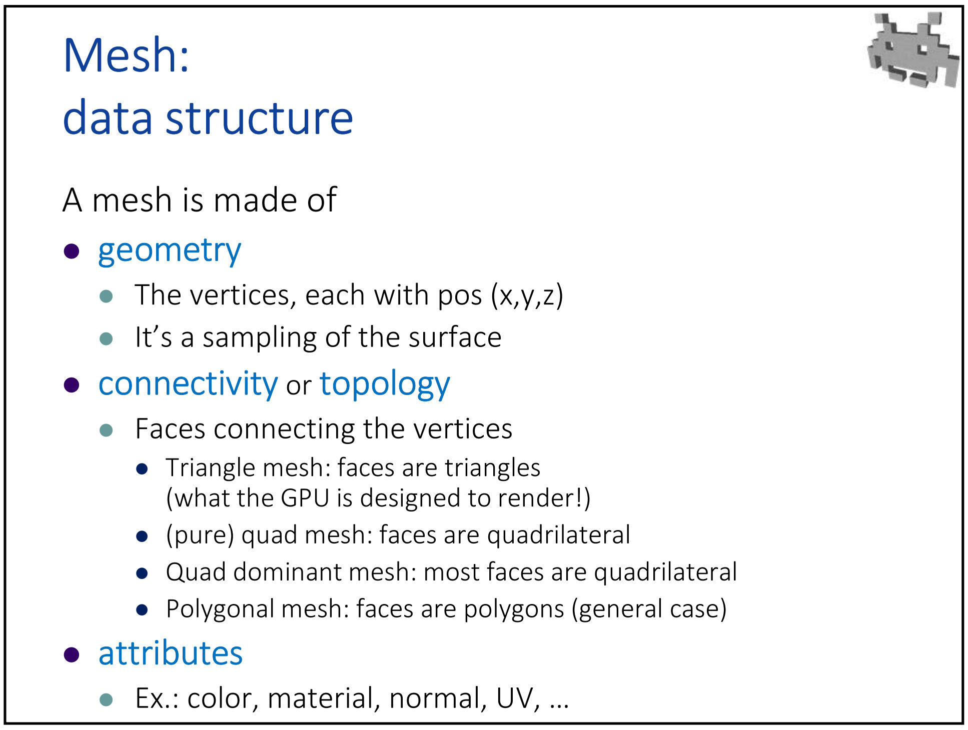


Triangle Mesh (or simplicial mesh)

- A set of adjacent triangles



5



Mesh: data structure

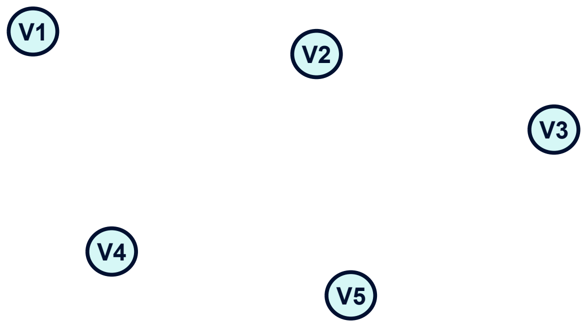
A mesh is made of

- 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, …

6

Mesh: geometry

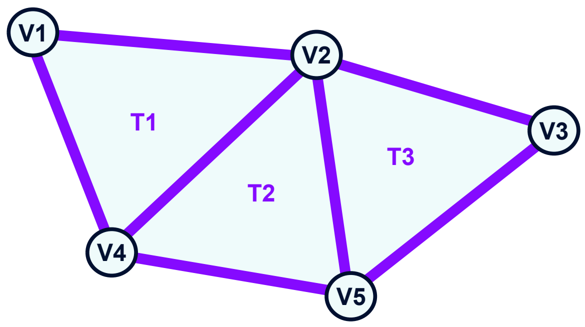
- Set of vertices
  - A position vector (x,y,z) for every vertex
  - Coordinates, by definition, are given in Local space!



7

Mesh: connectivity (or topology)

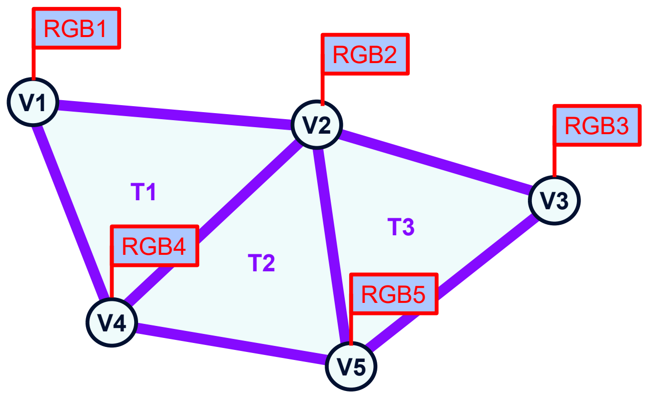
- Faces: triangles connecting vertices
  - More in general, polygons,
  - connecting triplet of *vertices*
  - just as, in a graph, *nodes* are connected by *edges*



8

Mesh: attributes

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



9

Mesh as a data structure: soup of triangles

- Simply, an array of triangles
- Each triangle stored as: sequence of 3 vertices
- Each vertex stored as:
  x,y,z coordinates + attributes
- Problem: data replication
  - Not very memory efficient
  - Inconvenient to update (e.g. to animate)
  - Not very used

most faces are adjacent to each other (adjacent faces share the same vertices)

10

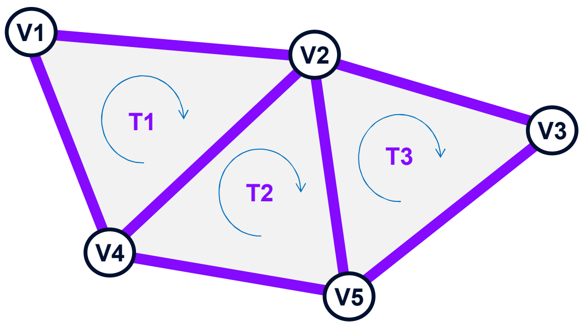
Mesh as a data strucuture: indexed meshes

- array of vertices
  - Each vertex stored as
    - *x,y,z* position (aka the "geometry" of the mesh)
    - attributes: (all vertices, the same ones)
      any data saved on the surface: e.g. color
- array of triangles
  - the "connectivity» (or, "topology") of the mesh
  - Each triangle stored as
    - triplet of indices (referring to a vertex in the array)
- The two arrays can be seen as tables

we can consider positions as attributes too

11

An indexed mesh in GPU ram = two buffers



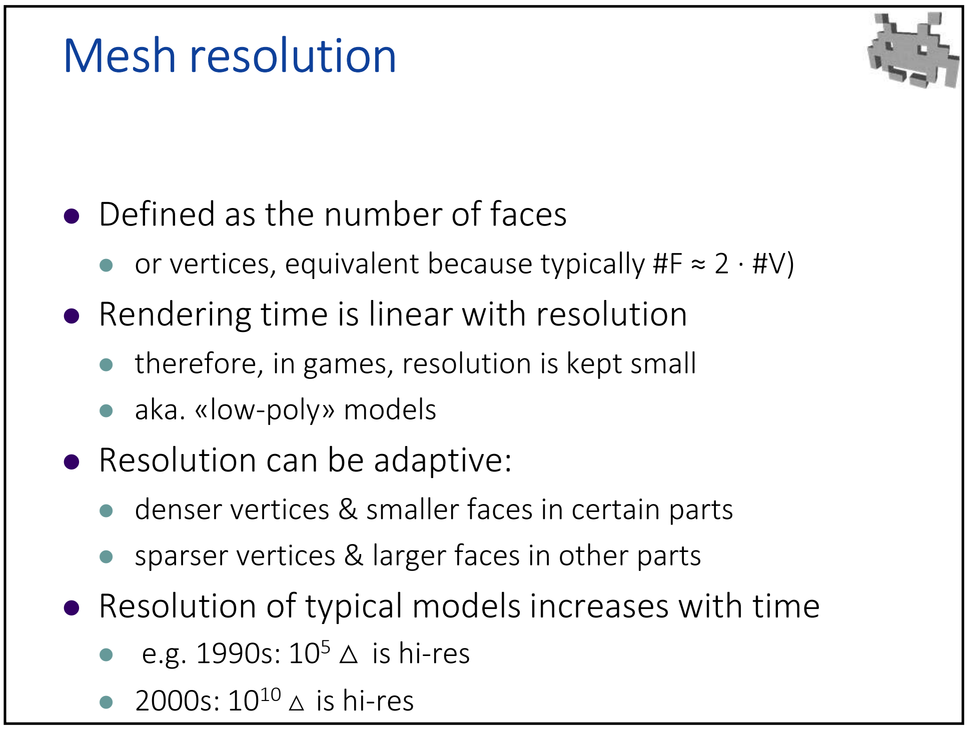
vert	X	Y	Z	R	G	B
V1	x1	y1	z1	r1	g1	b1
V2	x2	y2	z2	r2	g2	b2
V3	x3	y3	z3	r3	g3	b3
V4	x4	y4	z4	r4	g4	b4
V5	x5	y5	z5	r5	g5	b5

GEOMETRY + ATTRIBUTES

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

CONNECTIVITY

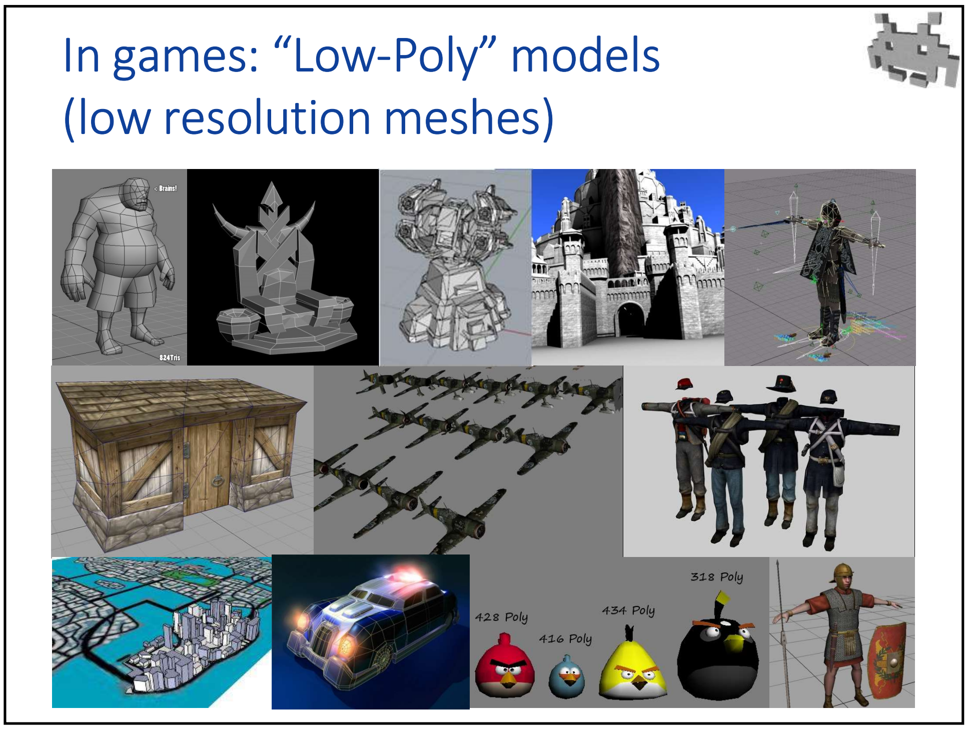
12



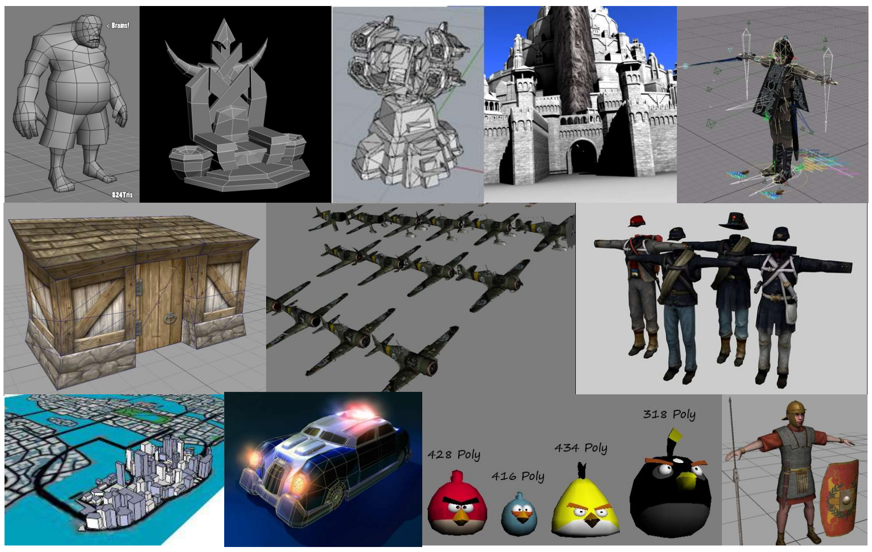
Mesh resolution

- Defined as 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

13

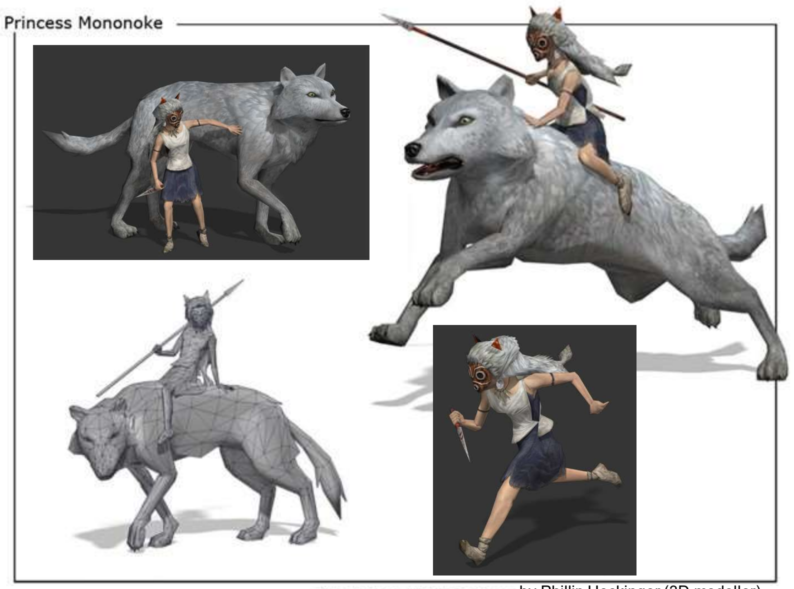


In games: "Low-Poly" models (low resolution meshes)



14

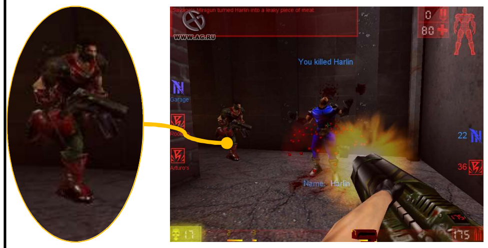
Low-poly models



by Phillip Heckinger (3D modeller)

16

Resolution increases over time

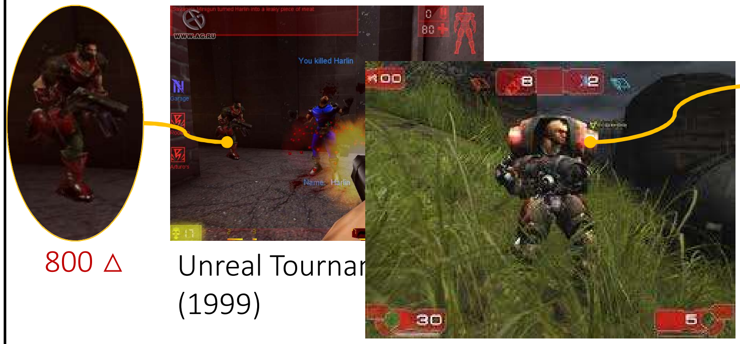


800 △

Unreal Tournament
(1999)

17

Resolution increases over time



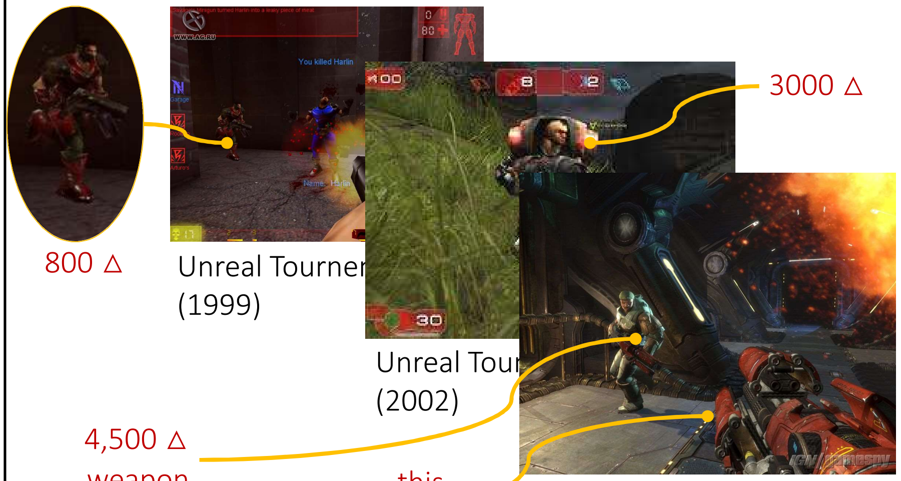
800 △

Unreal Tournar
(1999)

3000 △

Unreal Tournament 2K3
(2002)

Resolution increases over time



800 △

Unreal Tourner
(1999)

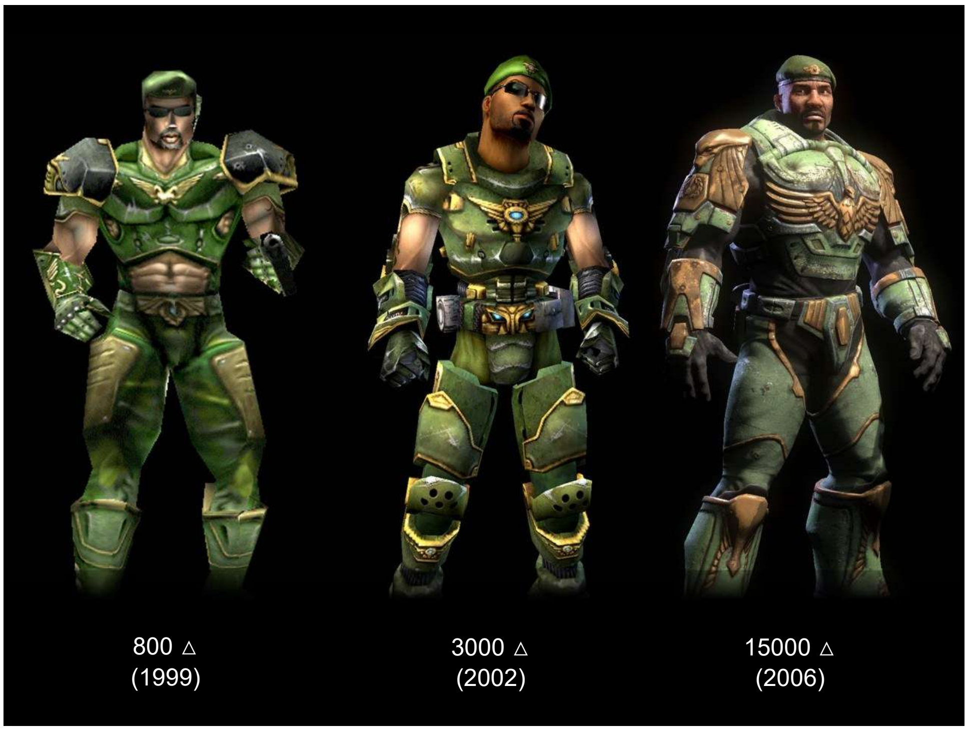
3000 △

Unreal Tour
(2002)

4,500 △
weapon

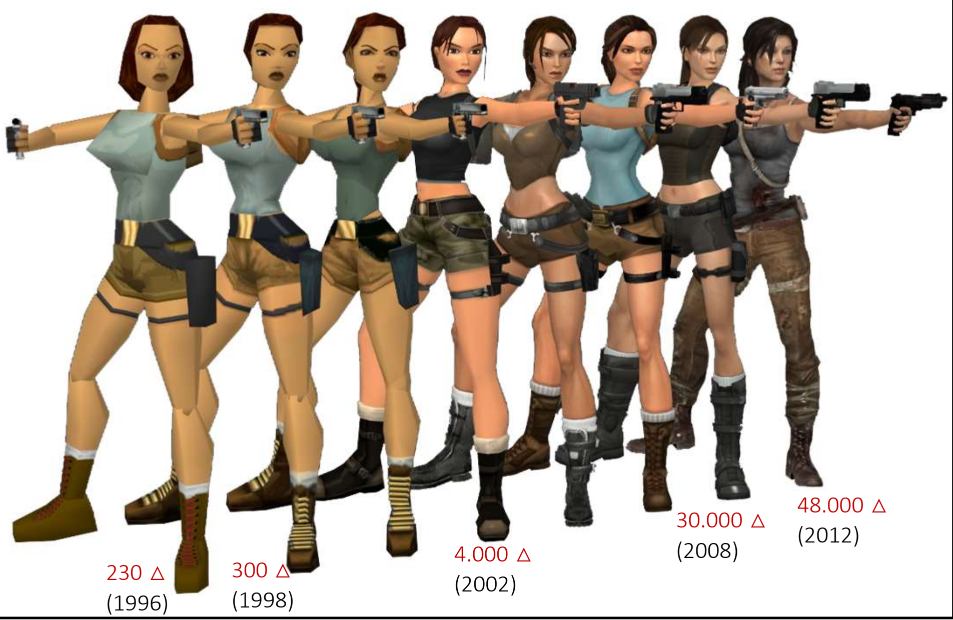
this
12,000 △

Unreal Tournament 3
(2007)

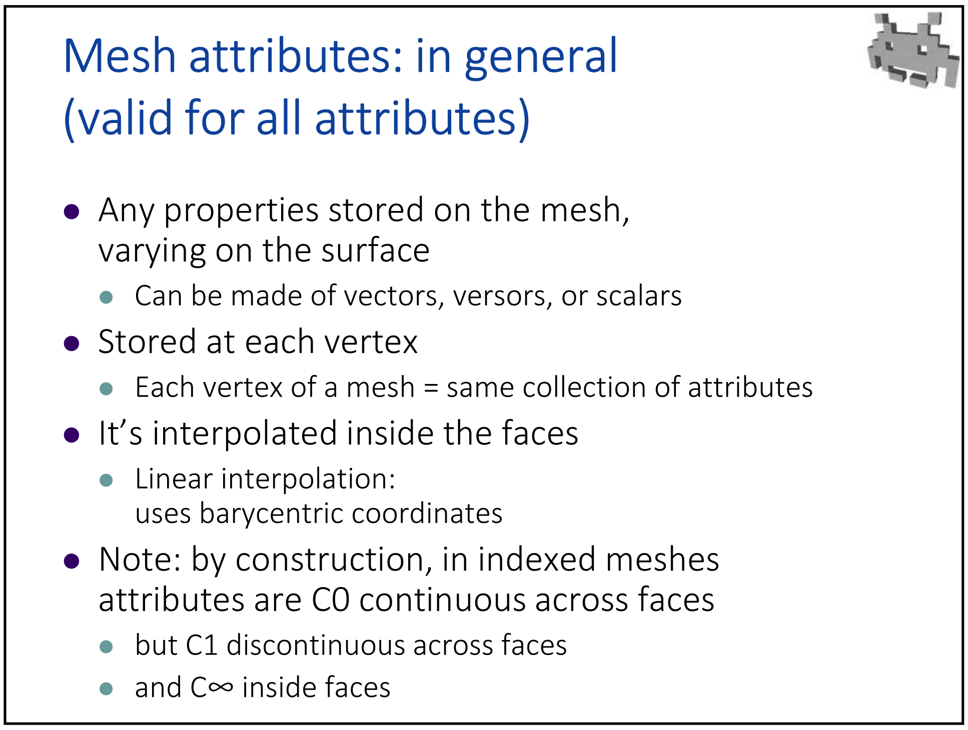


20

Resolution increases over time



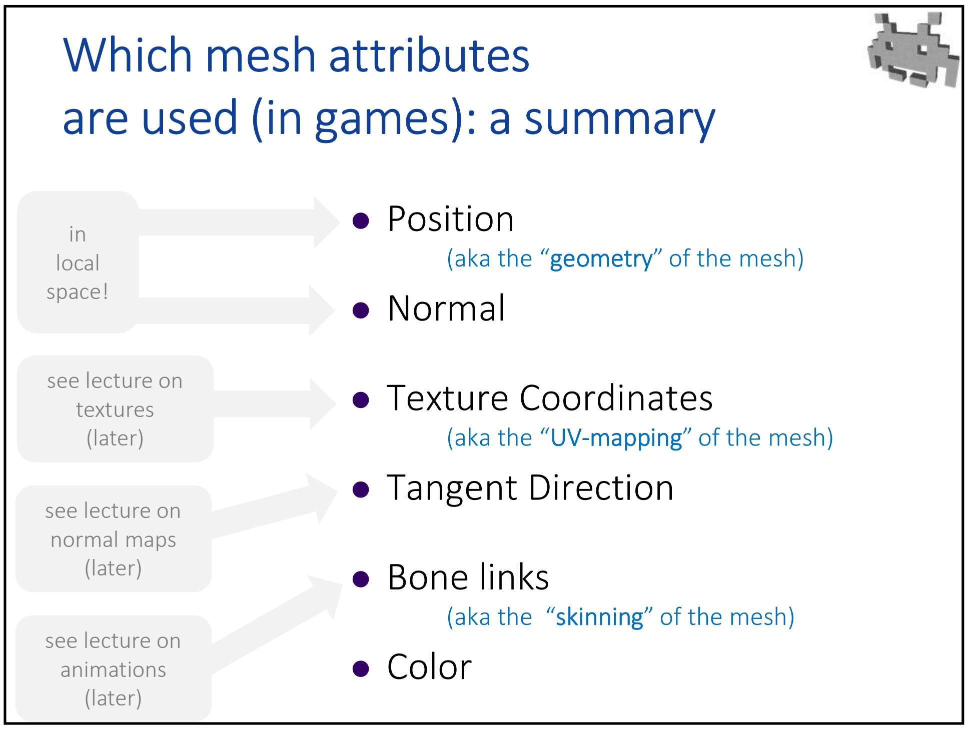
21



Mesh attributes: in general (valid for all attributes)

- Any properties stored on the mesh, varying on the surface
  - Can be made of vectors, versors, or scalars
- Stored at each vertex
  - Each vertex of a mesh = same collection of attributes
- It's interpolated inside the faces
  - Linear interpolation: uses barycentric coordinates
- Note: by construction, in indexed meshes attributes are C0 continuous across faces
  - but C1 discontinuous across faces
  - and C∞ inside faces

22



Which mesh attributes are used (in games): a summary

in local space!

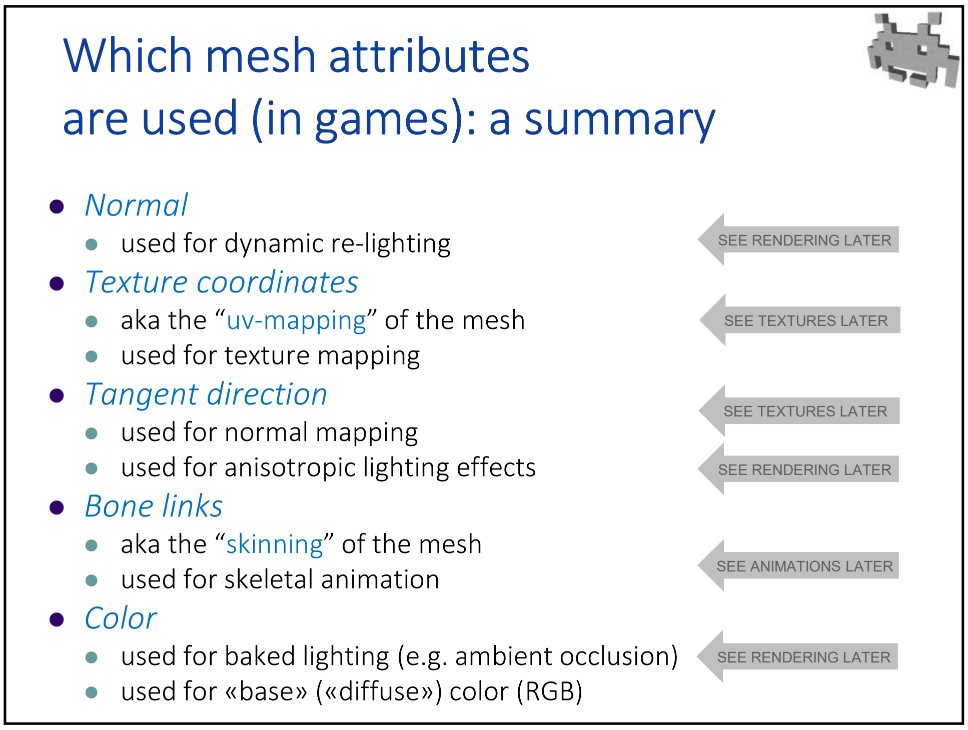
see lecture on textures (later)

see lecture on normal maps (later)

see lecture on animations (later)

- Position
  (aka the "**geometry**" of the mesh)
- Normal
- Texture Coordinates
  (aka the "**UV-mapping**" of the mesh)
- Tangent Direction
- Bone links
  (aka the "**skinning**" of the mesh)
- Color

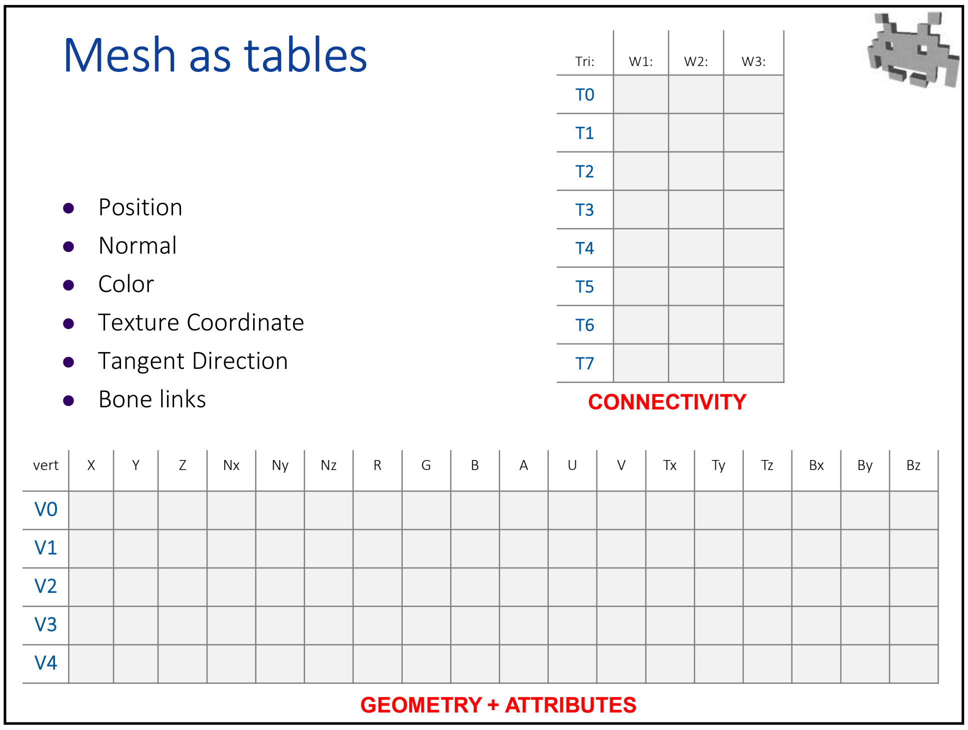
23



Which mesh attributes are used (in games): a summary

- *Normal*
  - used for dynamic re-lighting — SEE RENDERING LATER
- *Texture coordinates*
  - aka the "uv-mapping" of the mesh — SEE TEXTURES LATER
  - used for texture mapping
- *Tangent direction*
  - used for normal mapping — SEE TEXTURES LATER
  - used for anisotropic lighting effects — SEE RENDERING LATER
- *Bone links*
  - aka the "skinning" of the mesh — SEE ANIMATIONS LATER
  - used for skeletal animation
- *Color*
  - used for baked lighting (e.g. ambient occlusion) — SEE RENDERING LATER
  - used for «base» («diffuse») color (RGB)

24



Mesh as tables

- Position
- Normal
- Color
- Texture Coordinate
- Tangent Direction
- Bone links

Tri:	W1:	W2:	W3:
T0			
T1			
T2			
T3			
T4			
T5			
T6			
T7			

CONNECTIVITY

vert	X	Y	Z	Nx	Ny	Nz	R	G	B	A	U	V	Tx	Ty	Tz	Bx	By	Bz
V0																		
V1																		
V2																		
V3																		
V4																		

GEOMETRY + ATTRIBUTES

25

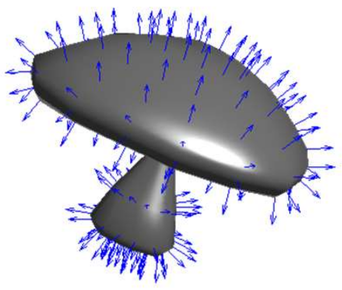
Mesh attributes: colors

- In games, colors on 3D models are usually determined by textures (not by mesh colors)
  - reason: more resolution in signal
- Per vertex colors can be used…
  - To cheaply add variations models
    - Red guards, blue guards SEE RENDERING LATER
  - To bake lighting
    - e.g. baked per-vertex ambient occlusion see rendering later
  - To dynamically recolor mesh parts
    - e.g. redden the tip of a sword which is blood soaked
    - e.g. accumulate dirty

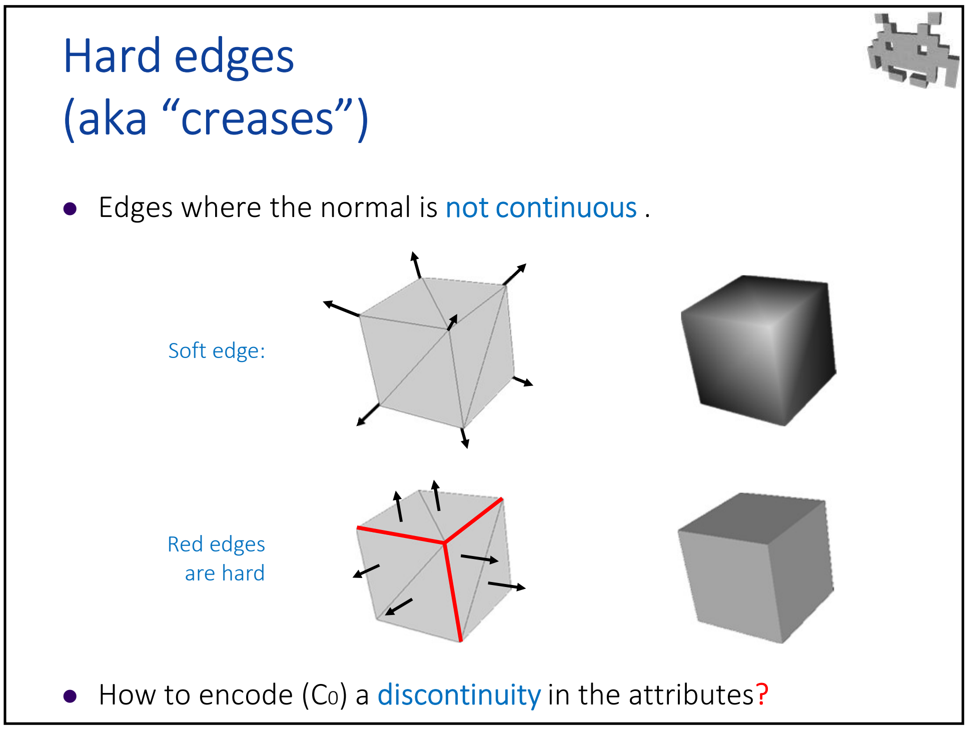
26

Mesh attributes: normals

- A versor
- Representing the surface orientation
- Main use: lighting computation
- Can be computed automatically from geometry…
- But it is a part of the mesh assets:
  - the artist is in control of which edges are *soft* and which are *hard*

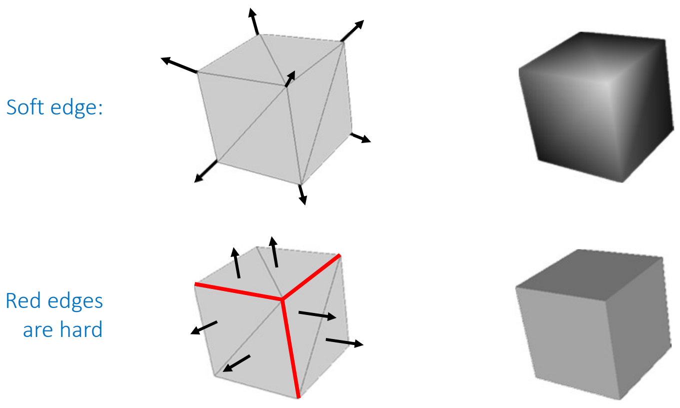


27



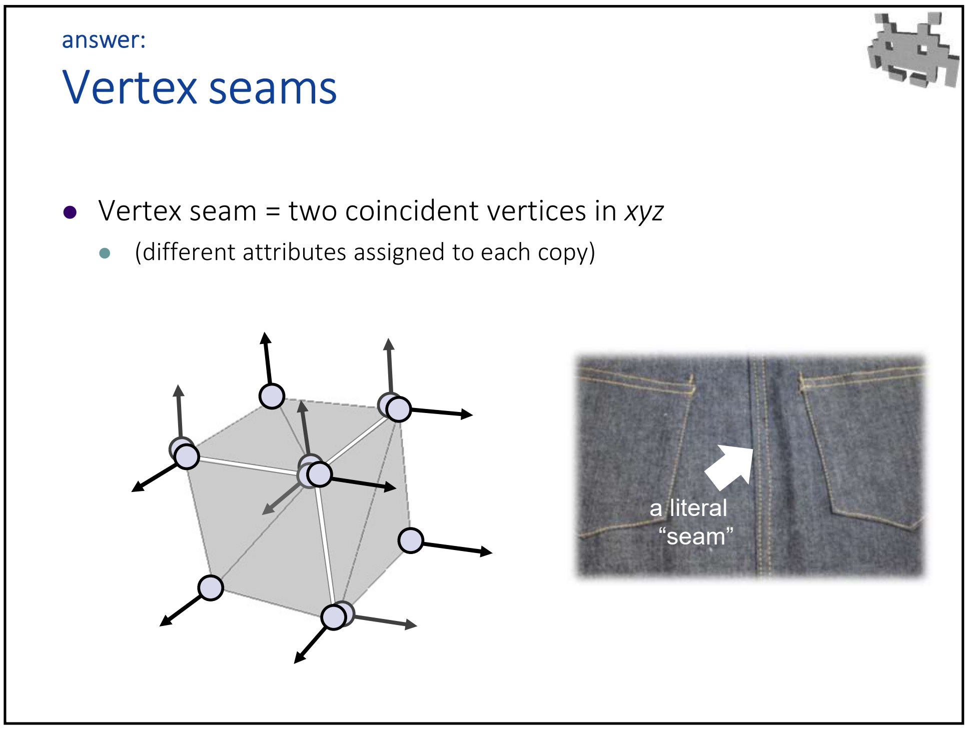
Hard edges (aka “creases”)

- Edges where the normal is not continuous .



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

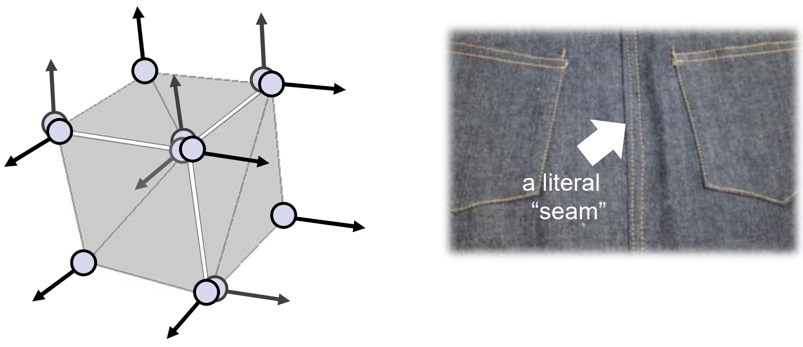
28



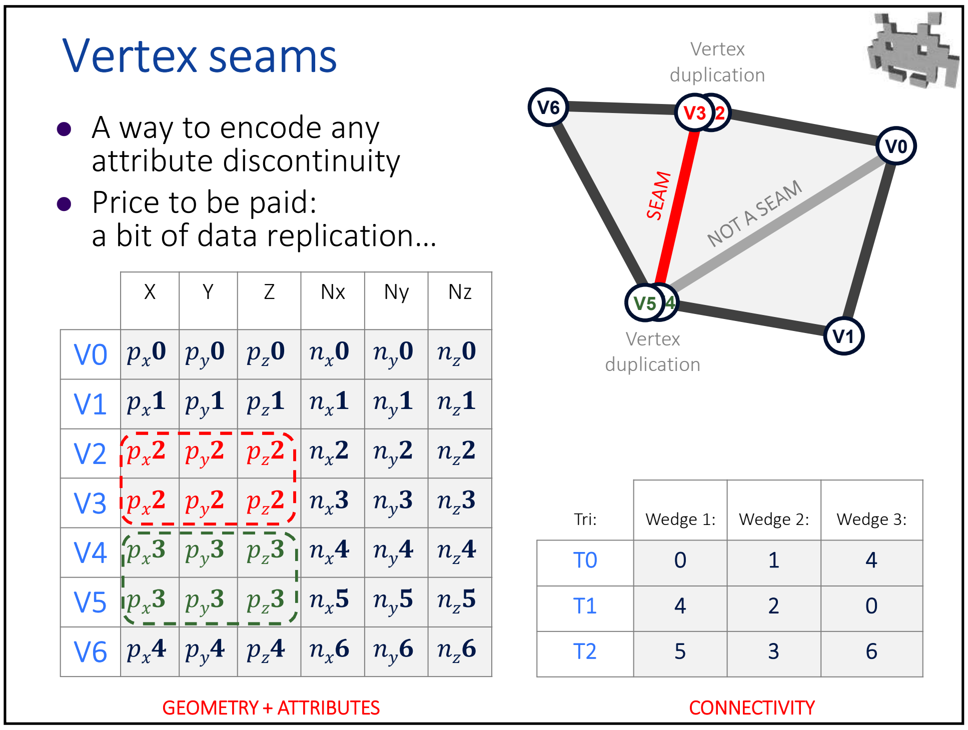
answer:

Vertex seams

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



29

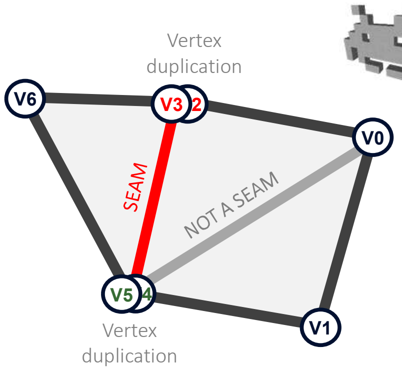


Vertex seams

- A way to encode any attribute discontinuity
- Price to be paid: a bit of data replication…

	X	Y	Z	Nx	Ny	Nz
V0	p_x0	p_y0	p_z0	n_x0	n_y0	n_z0
V1	p_x1	p_y1	p_z1	n_x1	n_y1	n_z1
V2	p_x2	p_y2	p_z2	n_x2	n_y2	n_z2
V3	p_x2	p_y2	p_z2	n_x3	n_y3	n_z3
V4	p_x3	p_y3	p_z3	n_x4	n_y4	n_z4
V5	p_x3	p_y3	p_z3	n_x5	n_y5	n_z5
V6	p_x4	p_y4	p_z4	n_x6	n_y6	n_z6

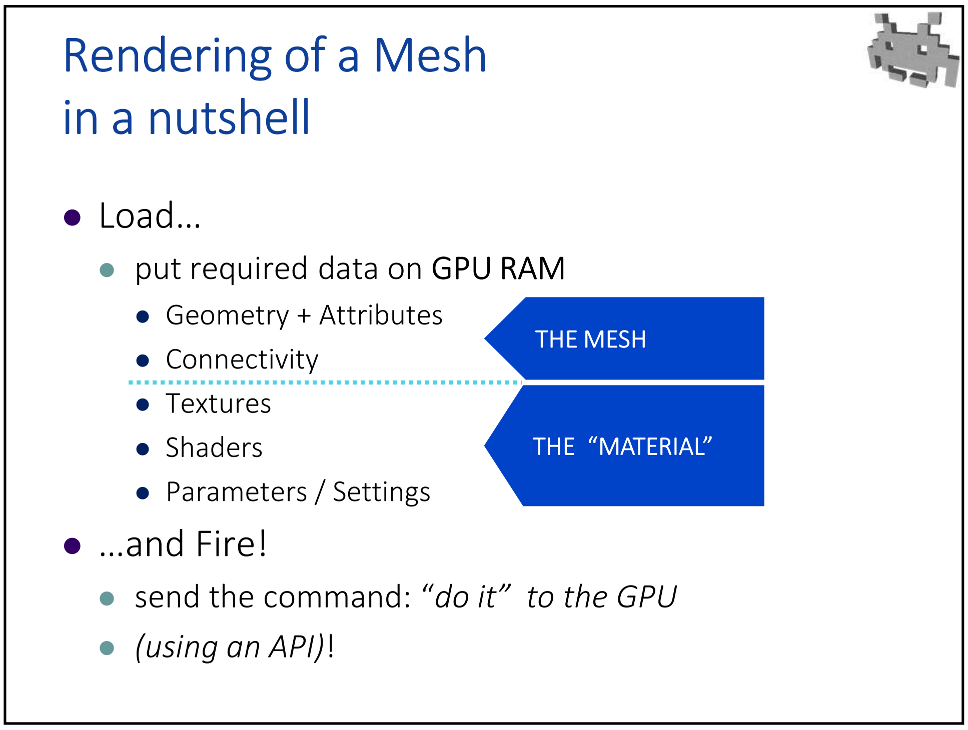
GEOMETRY + ATTRIBUTES



Tri:	Wedge 1:	Wedge 2:	Wedge 3:
T0	0	1	4
T1	4	2	0
T2	5	3	6

CONNECTIVITY

30



Rendering of a Mesh in a nutshell

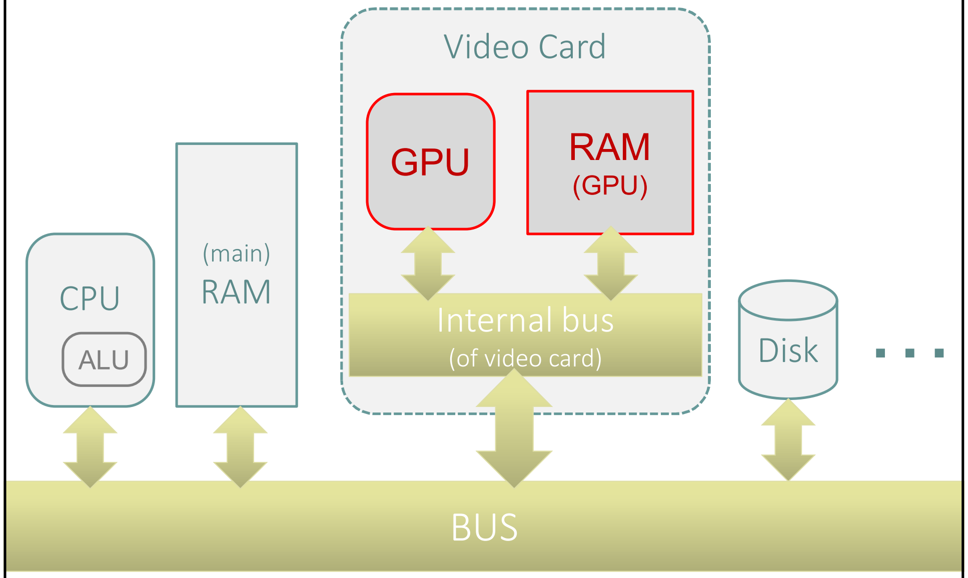
- Load…
  - put required data on GPU RAM
    - Geometry + Attributes
    - Connectivity
    - Textures
    - Shaders
    - Parameters / Settings
- …and Fire!
  - send the command: *"do it" to the GPU*
  - *(using an API)*!

THE MESH

THE "MATERIAL"

31

Simplified architecture of PC with Video Card



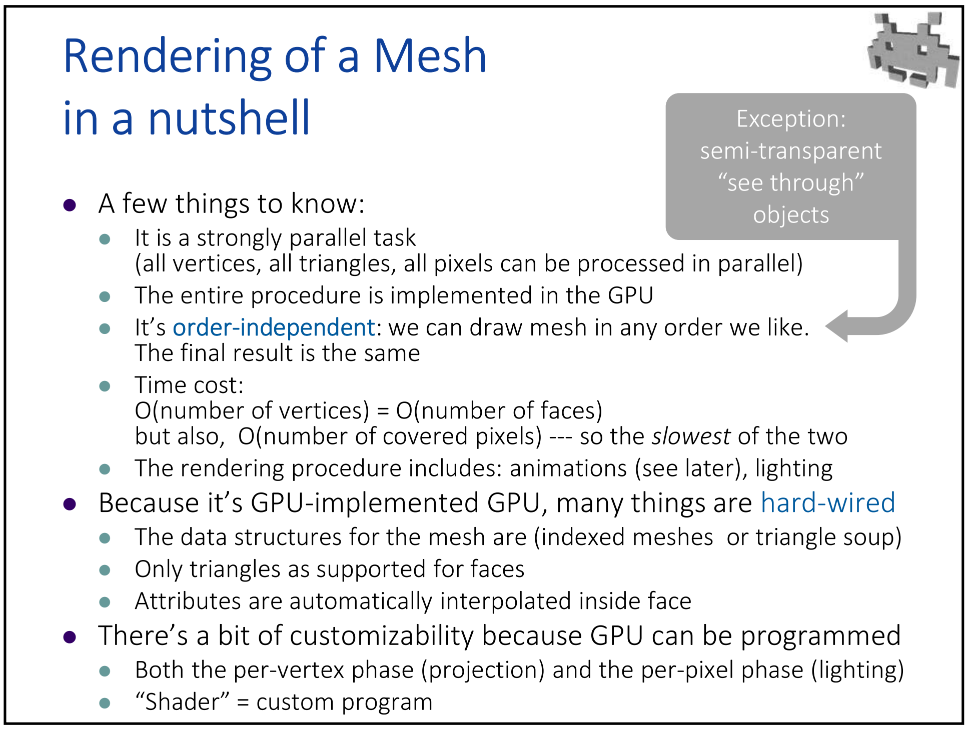
32

Rendering of a Mesh in a nutshell

Might change in the future?

- The algorithm to render a mesh (in games) is based on rasterization
  - It is outside the scope of this course. See CG course.
  - In brief three phases in cascade:
    each vertex is projected on screen ("transform"), PER VERTEX PHASE
    (find where the vertex will be seen on the screen)
    then each triangle is rasterized (converted into pixels) PER TRIANGLE PHASE
    then each pixel is processed (find the final color) PER PIXEL PHASE
- For our purposes, rendering a mesh means just: load all required data on the card on the GPU and send the command to render it (the "draw call")
  - data includes the mesh itself (the two tables)
  - plus the current transformations (from local space to view space)
  - plus data describing the view: the "material", including textures

33



Rendering of a Mesh in a nutshell

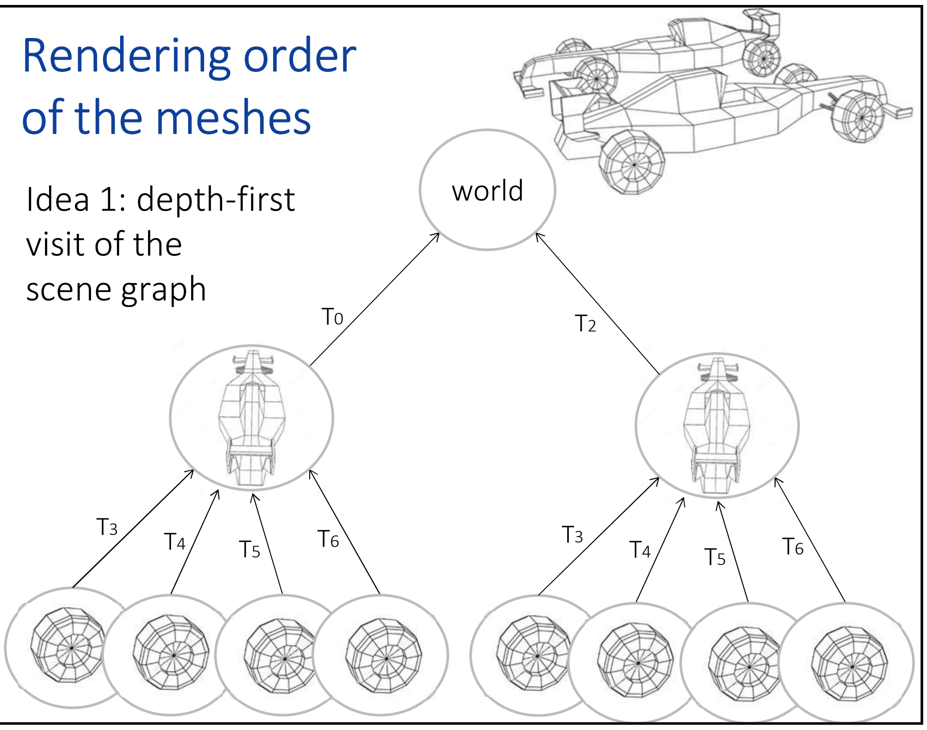
Exception: semi-transparent “see through” objects

- A few things to know:
  - It is a strongly parallel task (all vertices, all triangles, all pixels can be processed in parallel)
  - The entire procedure is implemented in the GPU
  - It’s order-independent: we can draw mesh in any order we like. The final result is the same
  - Time cost:
    O(number of vertices) = O(number of faces)
    but also, O(number of covered pixels) --- so the *slowest* of the two
  - The rendering procedure includes: animations (see later), lighting
- Because it’s GPU-implemented GPU, many things are hard-wired
  - The data structures for the mesh are (indexed meshes or triangle soup)
  - Only triangles as supported for faces
  - Attributes are automatically interpolated inside face
- There’s a bit of customizability because GPU can be programmed
  - Both the per-vertex phase (projection) and the per-pixel phase (lighting)
  - “Shader” = custom program

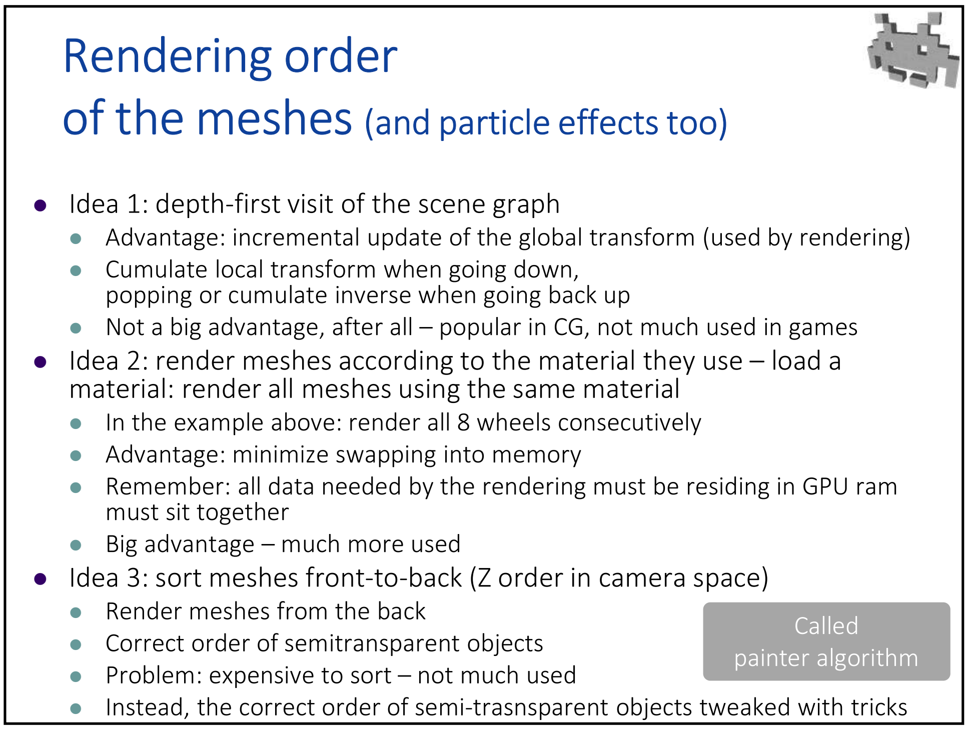
34

Rendering order of the meshes

Idea 1: depth-first visit of the scene graph



35

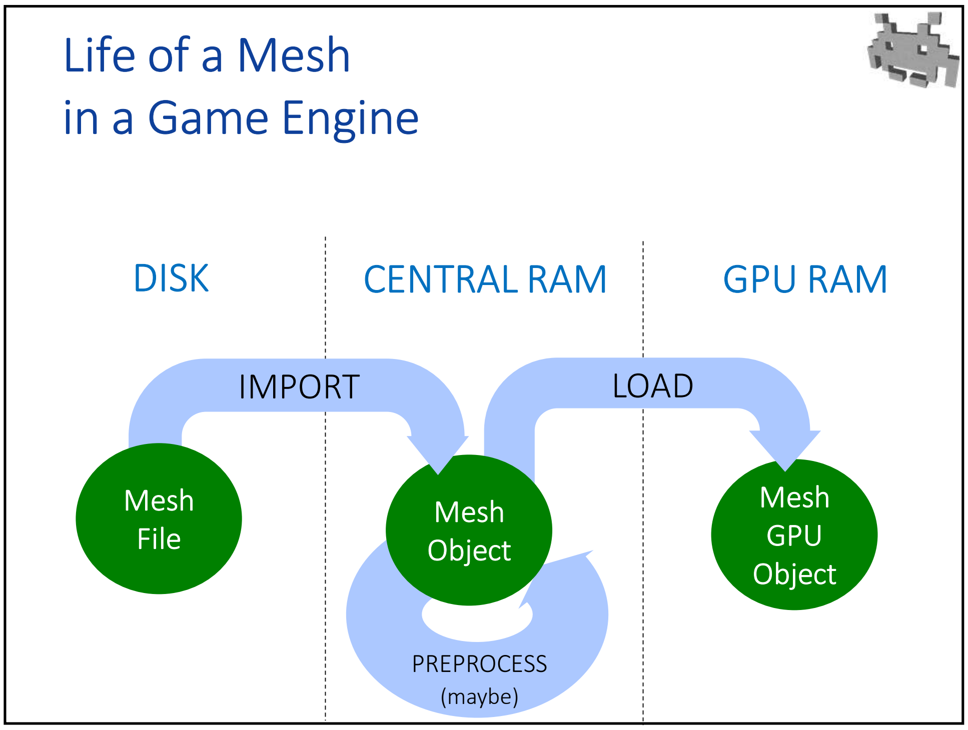


Rendering order of the meshes (and particle effects too)

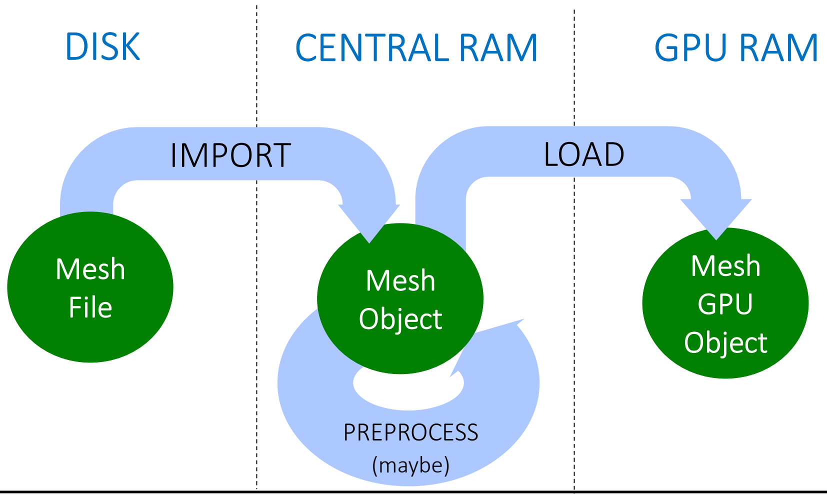
- Idea 1: depth-first visit of the scene graph
  - Advantage: incremental update of the global transform (used by rendering)
  - Cumulate local transform when going down, popping or cumulate inverse when going back up
  - Not a big advantage, after all – popular in CG, not much used in games
- Idea 2: render meshes according to the material they use – load a material: render all meshes using the same material
  - In the example above: render all 8 wheels consecutively
  - Advantage: minimize swapping into memory
  - Remember: all data needed by the rendering must be residing in GPU ram must sit together
  - Big advantage – much more used
- Idea 3: sort meshes front-to-back (Z order in camera space)
  - Render meshes from the back
  - Correct order of semitransparent objects
  - Problem: expensive to sort – not much used
  - Instead, the correct order of semi-trasnsparent objects tweaked with tricks

Called painter algorithm

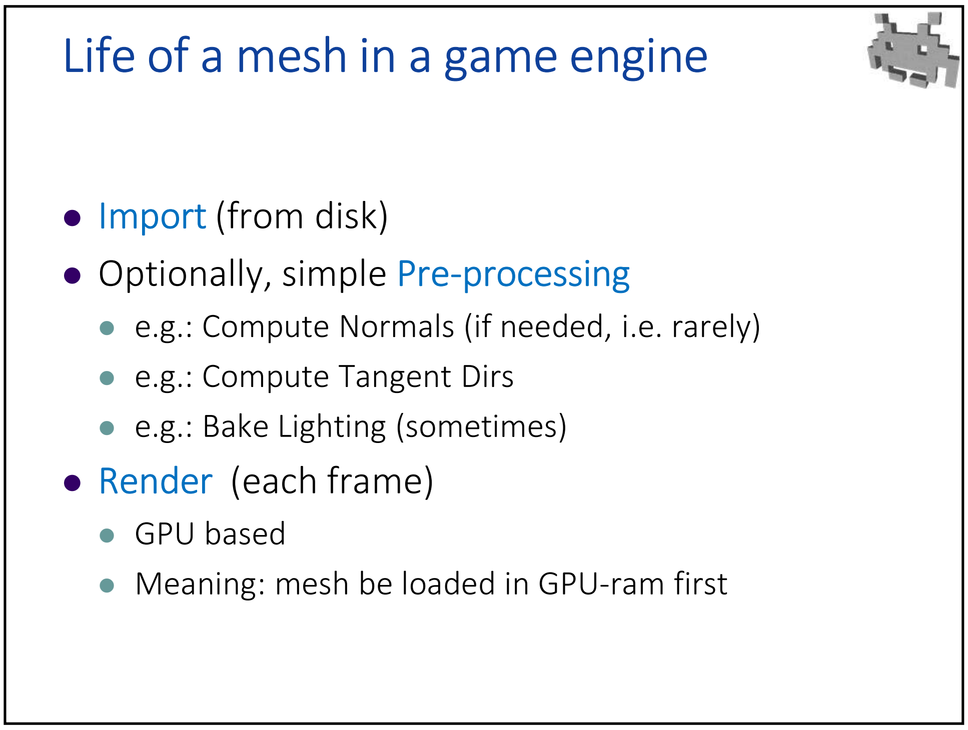
36



Life of a Mesh in a Game Engine



37

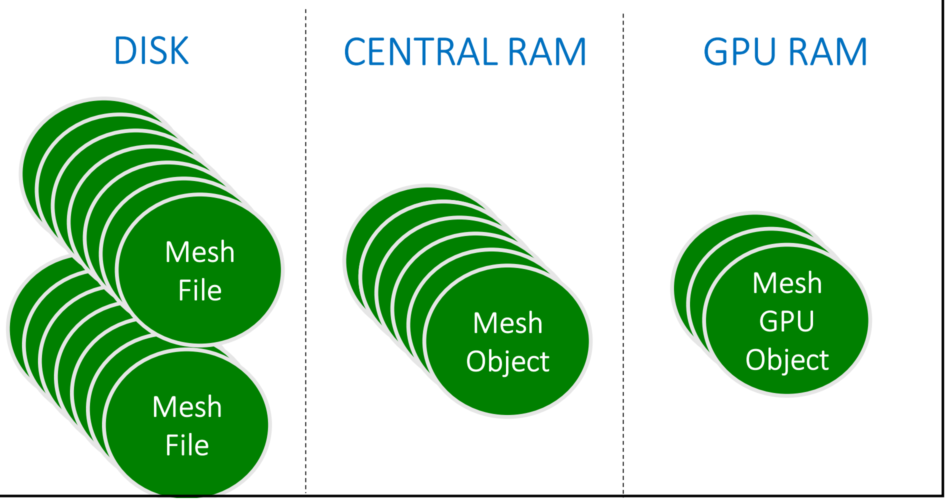


Life of a mesh in a game engine

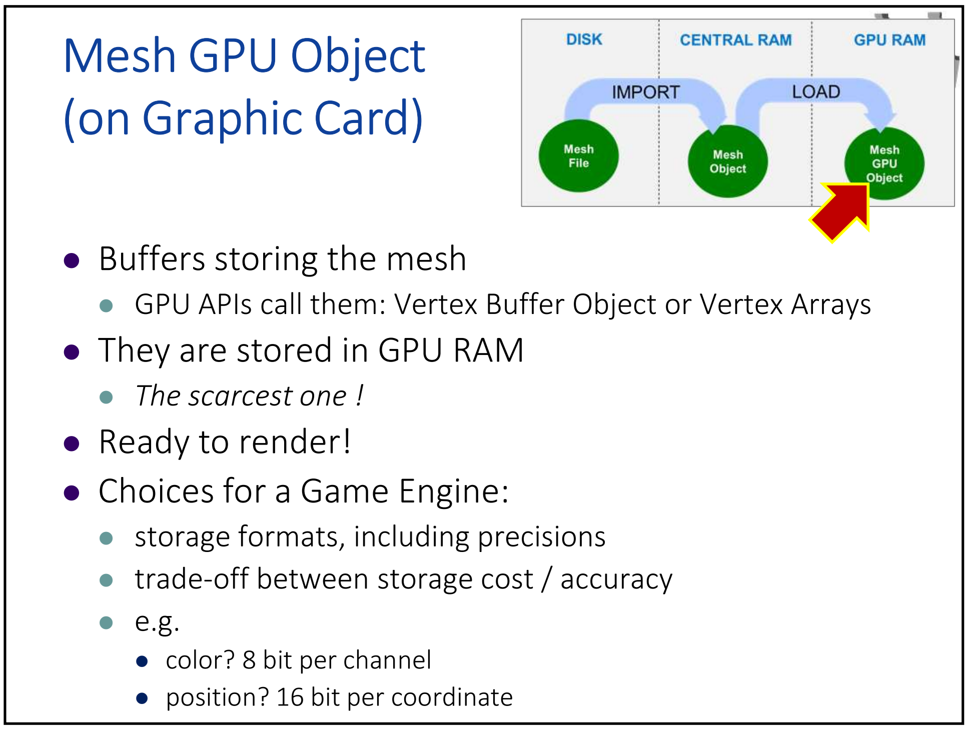
- Import (from disk)
- Optionally, simple Pre-processing
  - e.g.: Compute Normals (if needed, i.e. rarely)
  - e.g.: Compute Tangent Dirs
  - e.g.: Bake Lighting (sometimes)
- Render (each frame)
  - GPU based
  - Meaning: mesh be loaded in GPU-ram first

38

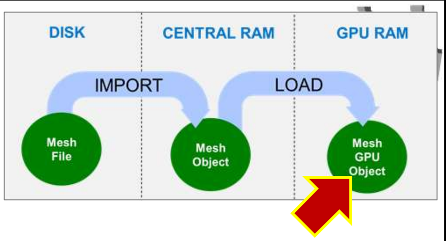
Memory Management (during game execution)



39

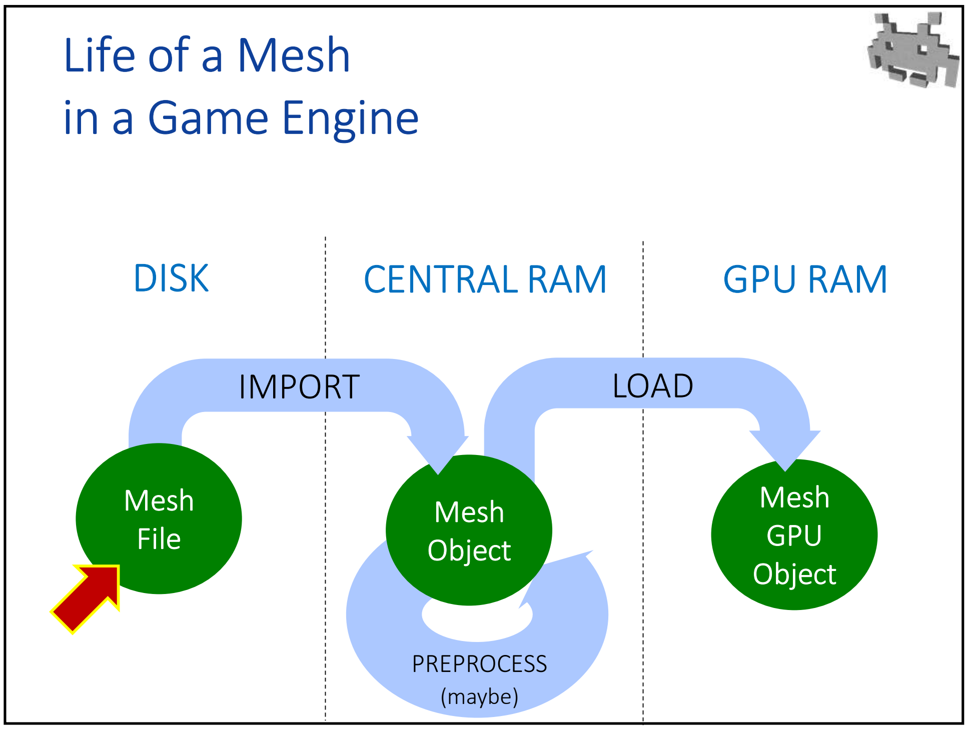


Mesh GPU Object (on Graphic Card)

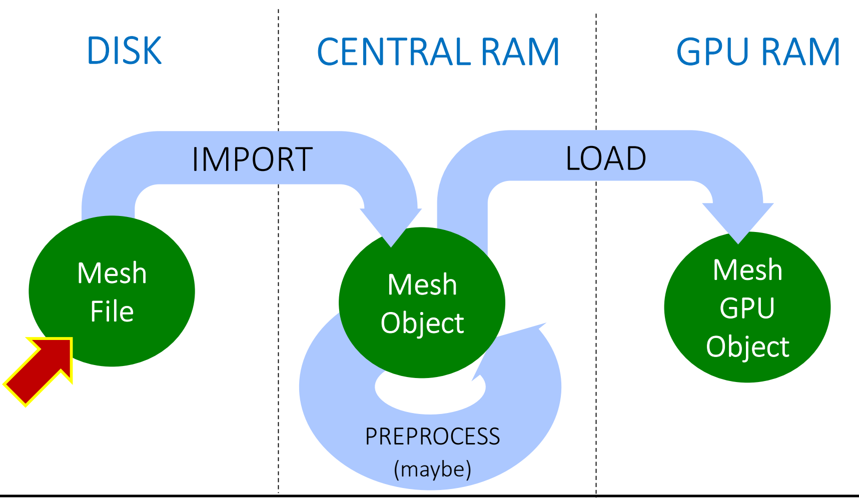


- Buffers storing the mesh
  - GPU APIs call them: Vertex Buffer Object or Vertex Arrays
- They are stored in GPU RAM
  - *The scarcest one !*
- Ready to render!
- Choices for a Game Engine:
  - storage formats, including precisions
  - trade-off between storage cost / accuracy
  - e.g.
    - color? 8 bit per channel
    - position? 16 bit per coordinate

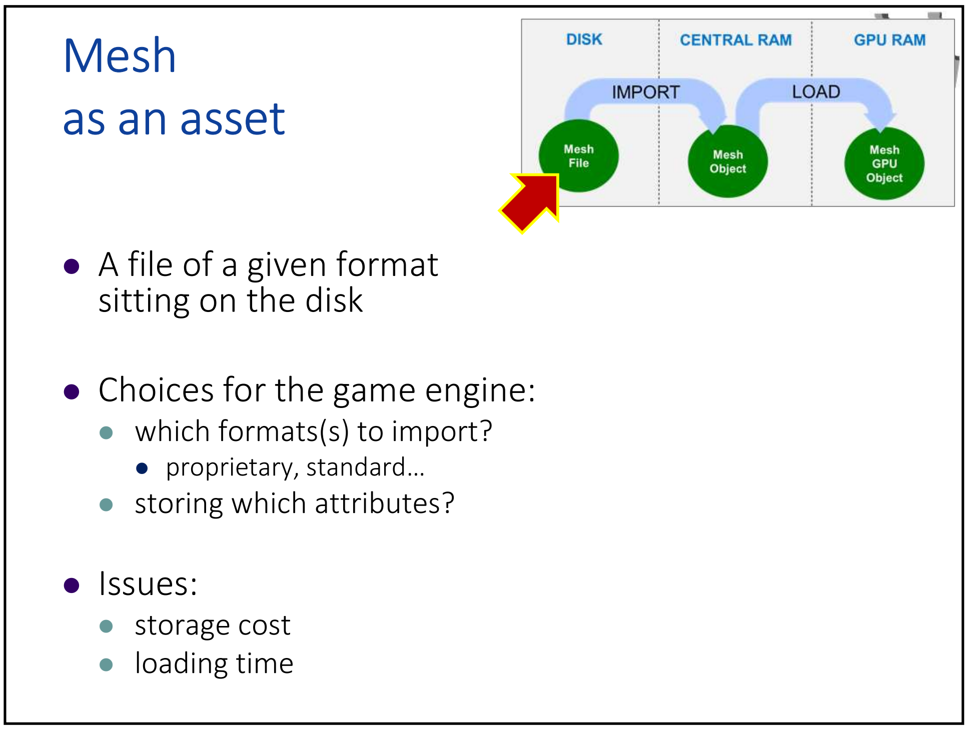
40



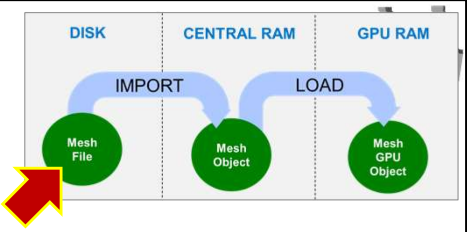
Life of a Mesh in a Game Engine



41

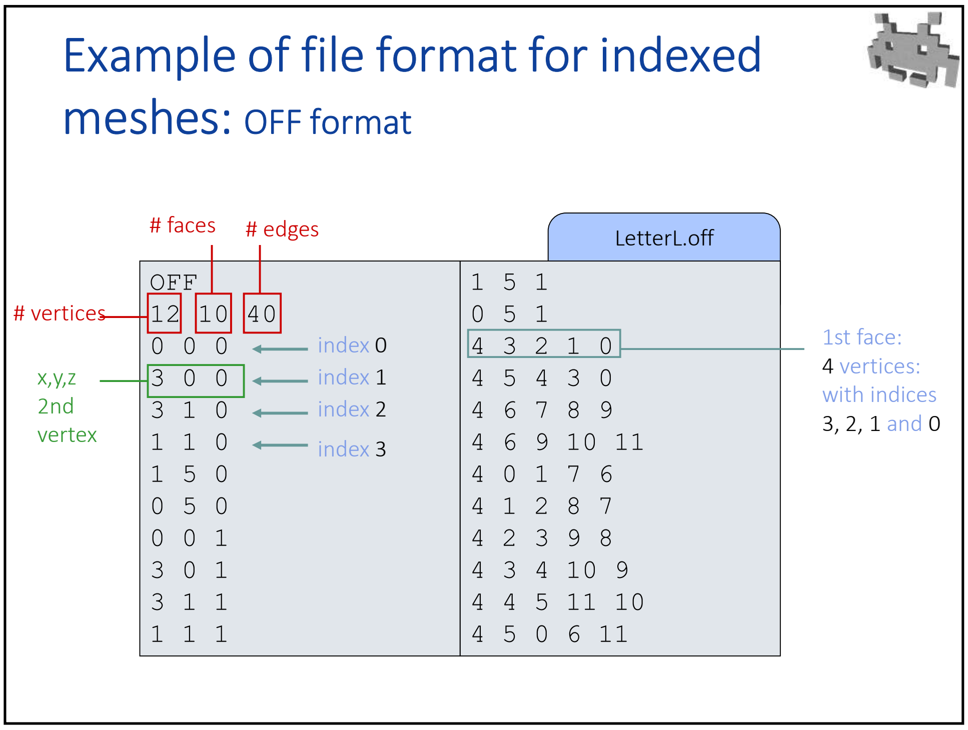


Mesh
as an asset

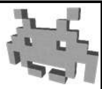


- A file of a given format sitting on the disk
- Choices for the game engine:
  - which formats(s) to import?
    - proprietary, standard…
  - storing which attributes?
- Issues:
  - storage cost
  - loading time

42



Example of file format for indexed meshes: OFF format



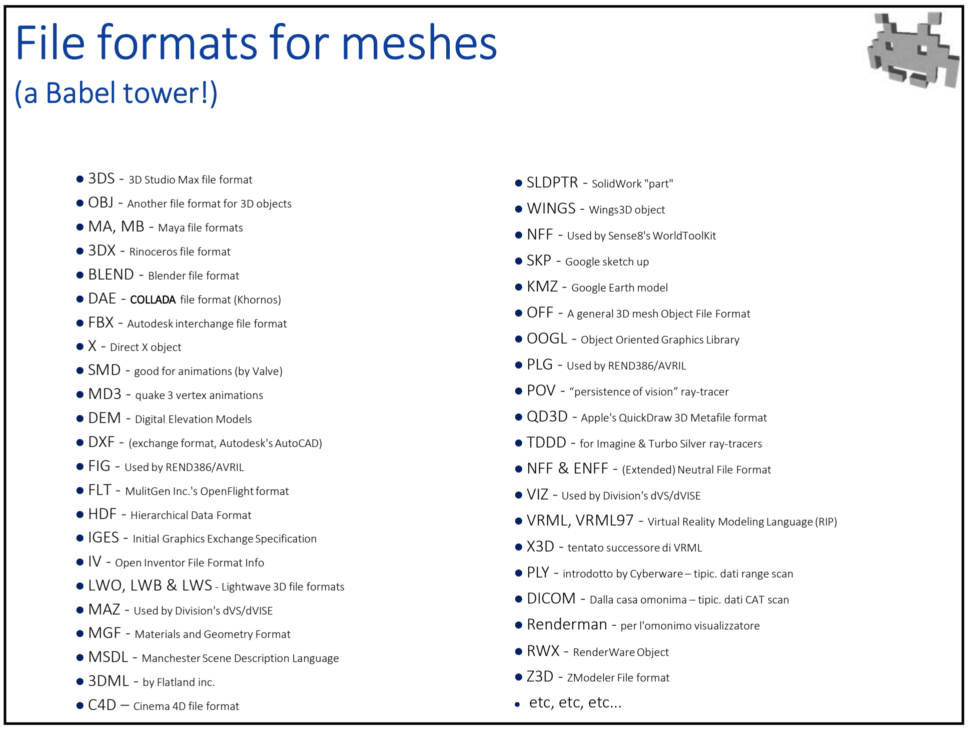
LetterL.off

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

# faces
edges
# vertices
x,y,z 2nd vertex
index 0
index 1
index 2
index 3

1st face: 4 vertices: with indices 3, 2, 1 and 0

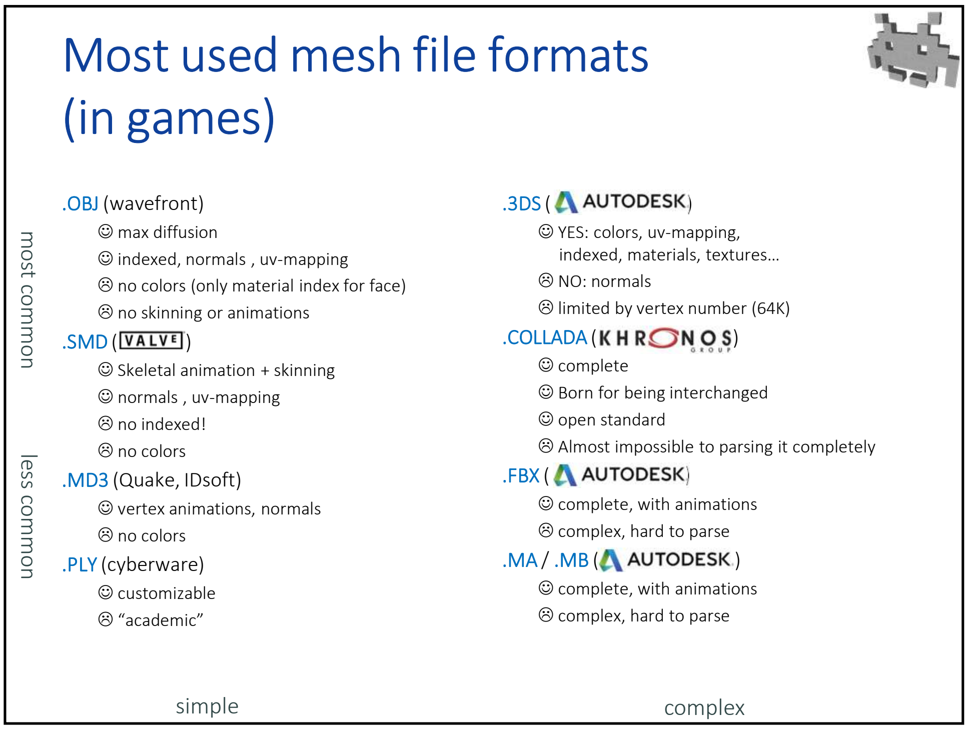
43



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...

45



Most used mesh file formats (in games)

most common

less common

.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
- ☹ "academic"

.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

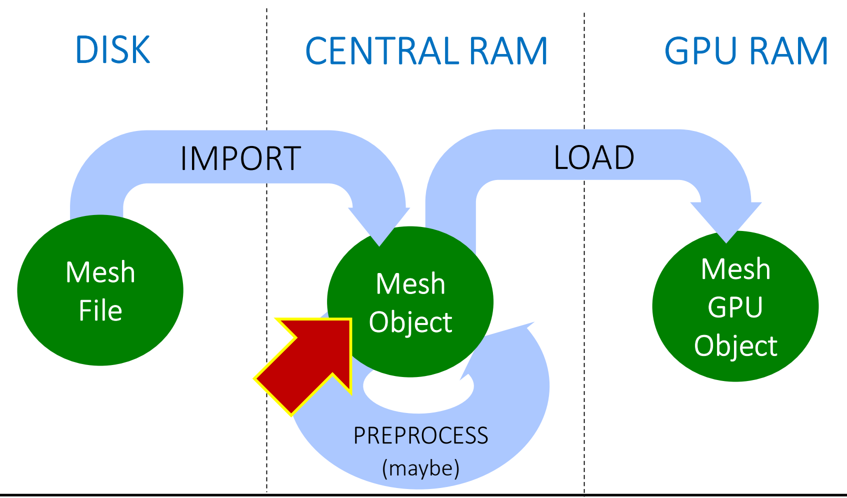
.MA / .MB (AUTODESK)
- ☺ complete, with animations
- ☹ complex, hard to parse

simple

complex

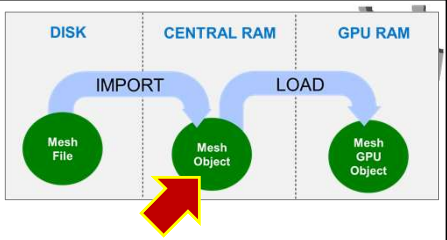
46

Life of a Mesh in a Game Engine



47

Mesh Object (in RAM)



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

48

How to represent a mesh?
(which data structures)

- Indexed mode in C++ :

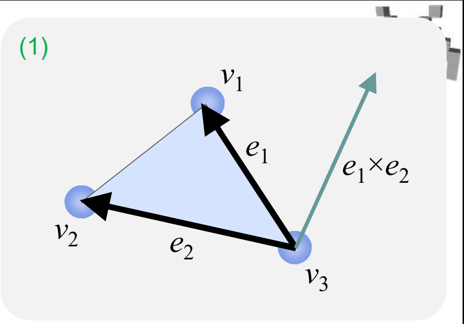
```
class Vertex {
  vec3 pos;
  rgb color;    /* attribute 1 */
  vec3 normal;  /* attribute 2 */
};

class Face{
   int vertexIndex[3];
};

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

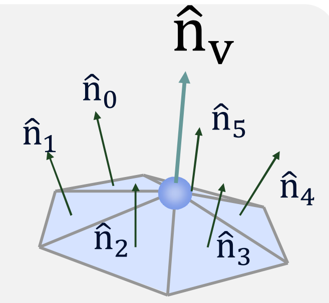
49

Computing normals from geometry

(1) compunte
normals of faces

(2) compute
normals of vertices





$$\hat{n}_v = \frac{\hat{n}_0 + \cdots + \hat{n}_k}{\|\hat{n}_0 + \cdots + \hat{n}_k\|}$$

52

Mesh processing:
(or, more in general, Geometry Processing)

- The algorithm above
  (for the computation of per vertex normal)
  is a tiny example of processing done over a mesh
- Mesh processing: the discipline of creating, transforming, computing meshes
  - inputs and/or outputs are meshes
- Part of, geometry processing:
  - when the input and output are other data structure for 3D models
  - See CG course for a very brief overview

53

Mesh processing:
typical tasks for the game industry

- Poly reduction / Retopology / Simplification
  - e.g. LOD construction
  - e.g. transition from (initial) hi-res to (final) low-poly
- Light baking LATER
  - Light precomputation
  - e.g.: Ambient Occlusion
- U-V map construction LATER
  - parametrization / unwrapping
- Texturing LATER
  - creation of different types of textures
- Rigging / Skinning / Animation LATER
  - to animate

56

Useful general tools: attribute transfer

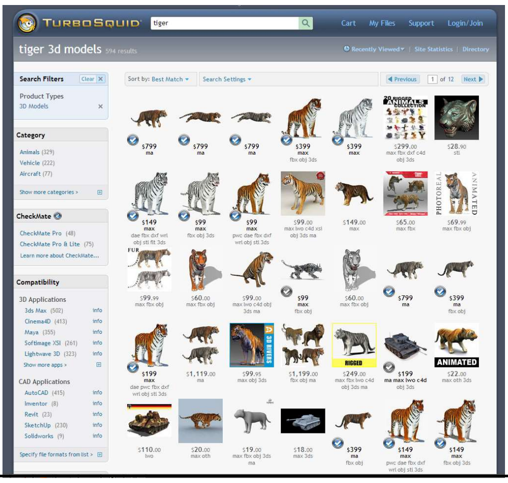
(any, see the list!)

- Given
  - a source mesh M_0 with attribute A
  - a target mesh M_1 similar (but not identical) to M_0 lacking that attribute
- Define attribute A in the vertices of M_1
  - Copying the attributes from M_0
- Result: “retargeting” of…
  - Animations, UV-mapping, textures, etc
- Results aren’t always perfect, but can be useful as a starting point

57

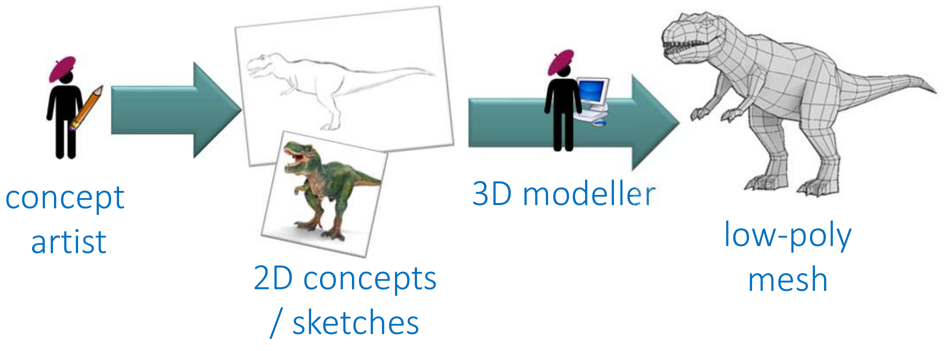
3D models: suorces

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

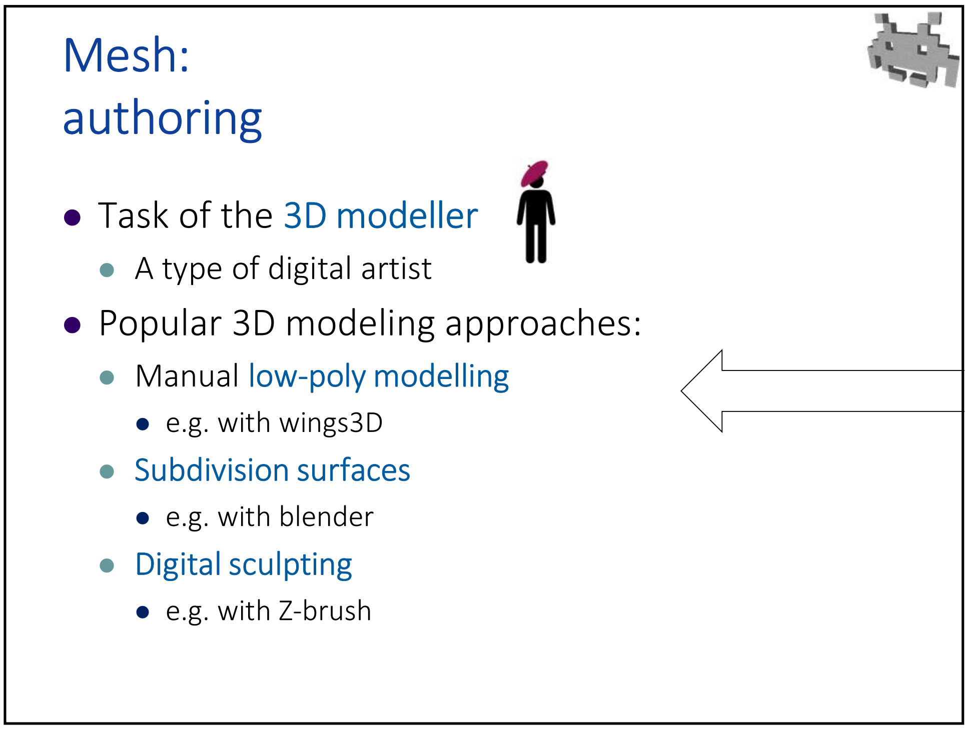


60

3D models: authoring pipeline



61



Mesh: authoring

- Task of the 3D modeller
  - A type of digital artist
- Popular 3D modeling approaches:
  - Manual low-poly modelling
    - e.g. with wings3D
  - Subdivision surfaces
    - e.g. with blender
  - Digital sculpting
    - e.g. with Z-brush

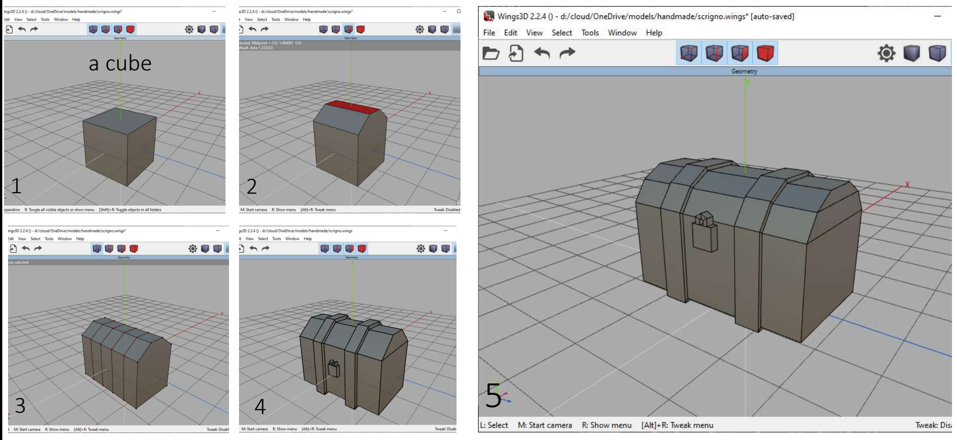
62

Mesh authoring: 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 modelling & 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...)

63

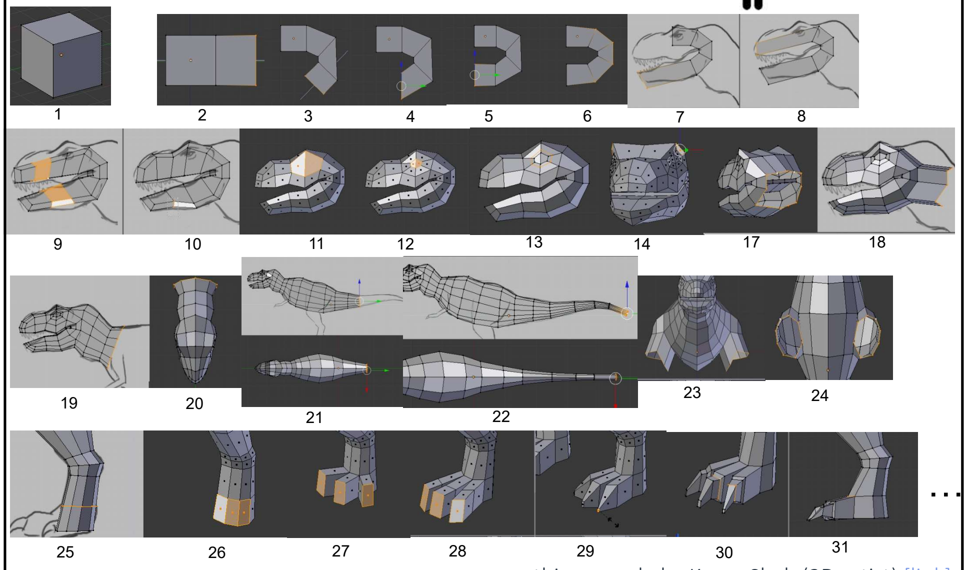
Low-poly modelling (demo)


a cube
1
2
3
4
5
Wings3D 2.2.4 () - d:/cloud/OneDrive/models/handmade/scrigno.wings* [auto-saved]
File Edit View Select Tools Window Help
Geometry
L: Select M: Start camera R: Show menu [Alt]+R: Tweak menu

Note: during creation, the meshes can be polygonal instead of triangle based, but is simple to decompose any polygon into triangles
E.g. this can be done by the game engine as a simple preprocessing.

64

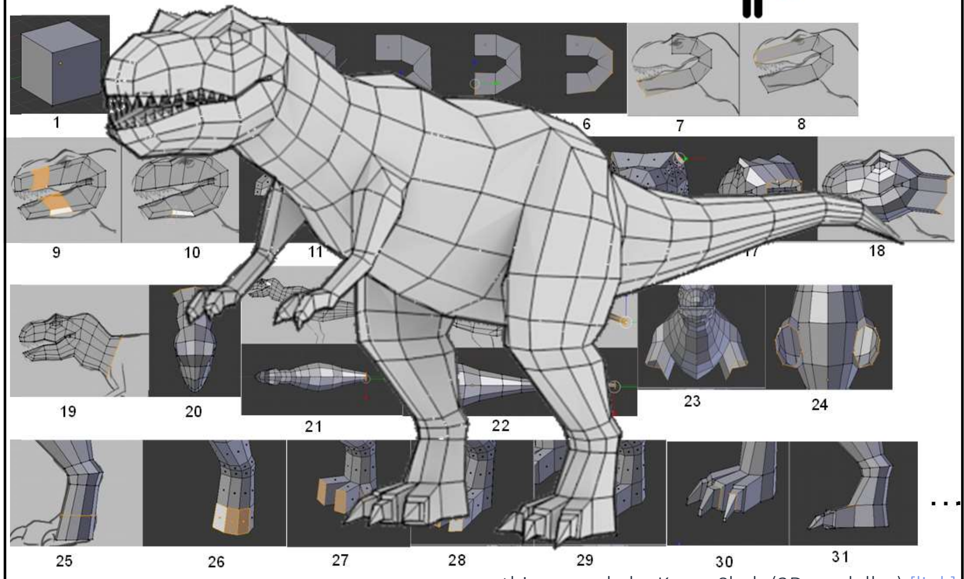
Low-poly modelling



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

66

Low-poly modelling



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

67