

Course Plan

lec. 1: Introduction

lec. 2: Mathematics for 3D Games

lec. 3: Scene Graph

lec. 4: Game 3D Physics

lec. 5: Game Particle Systems

lec. 6: Game 3D Models

lec. 7: Game Materials

lec. 9: Game Textures

lec. 8: Game 3D Animations

lec. 10: 3D Audio for 3D Games

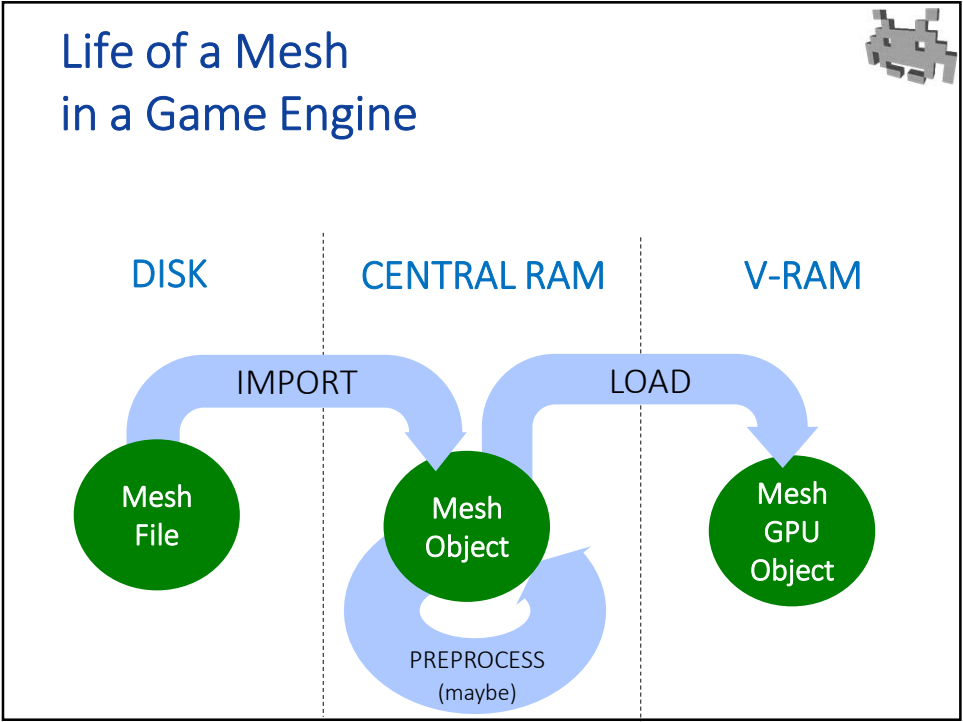
lec. 11: Networking for 3D Games

lec. 12: Artificial Intelligence for 3D Games

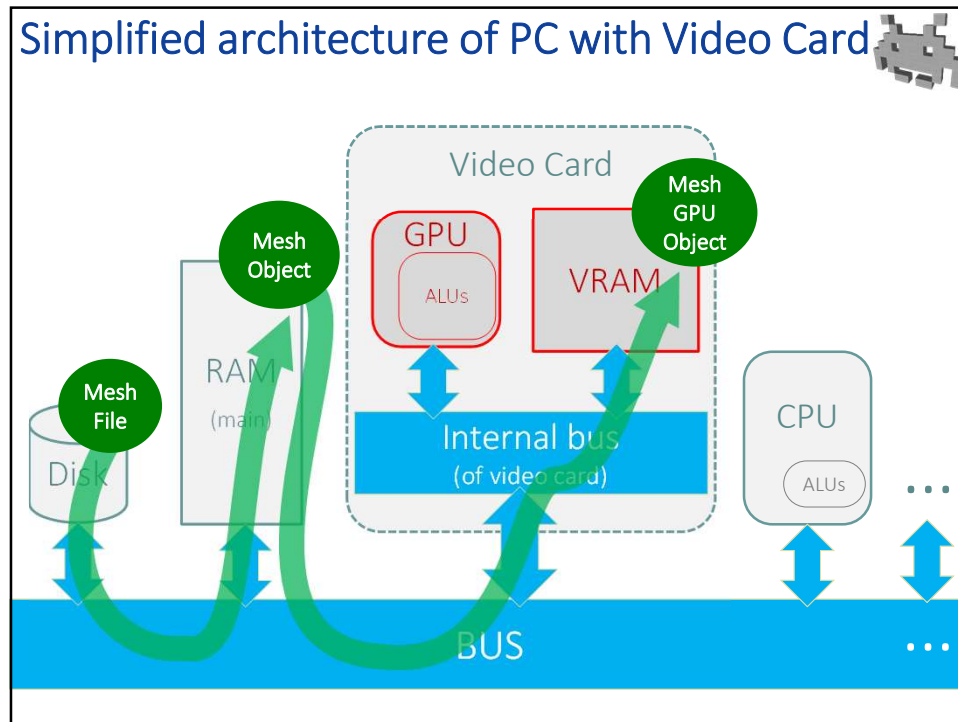
lec. 13: Rendering Techniques for 3D Games

appearance

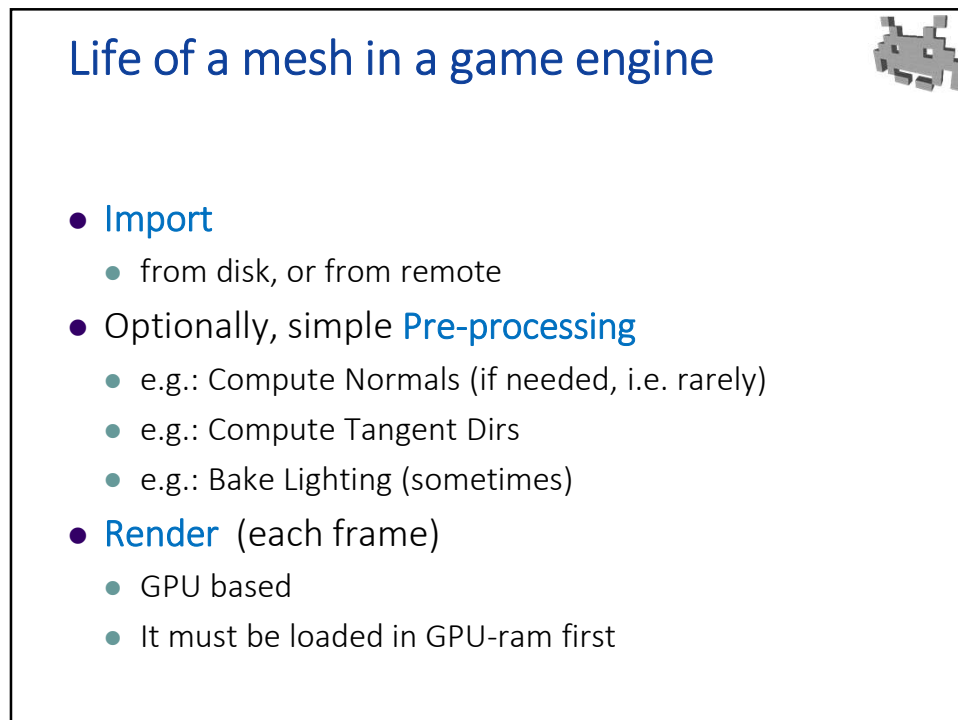
43



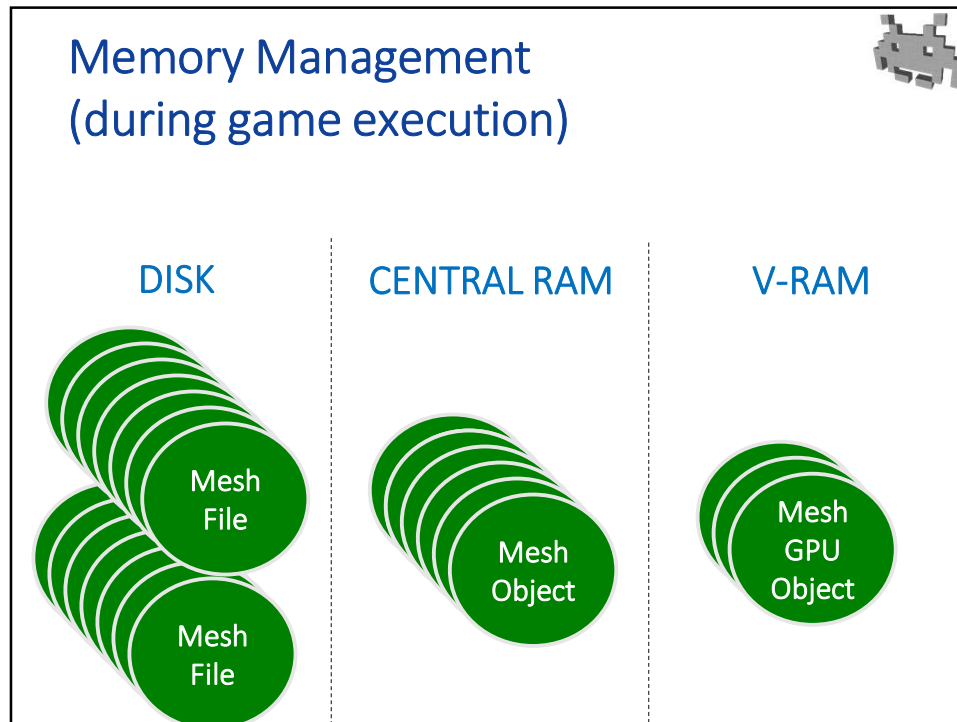
44



45



46



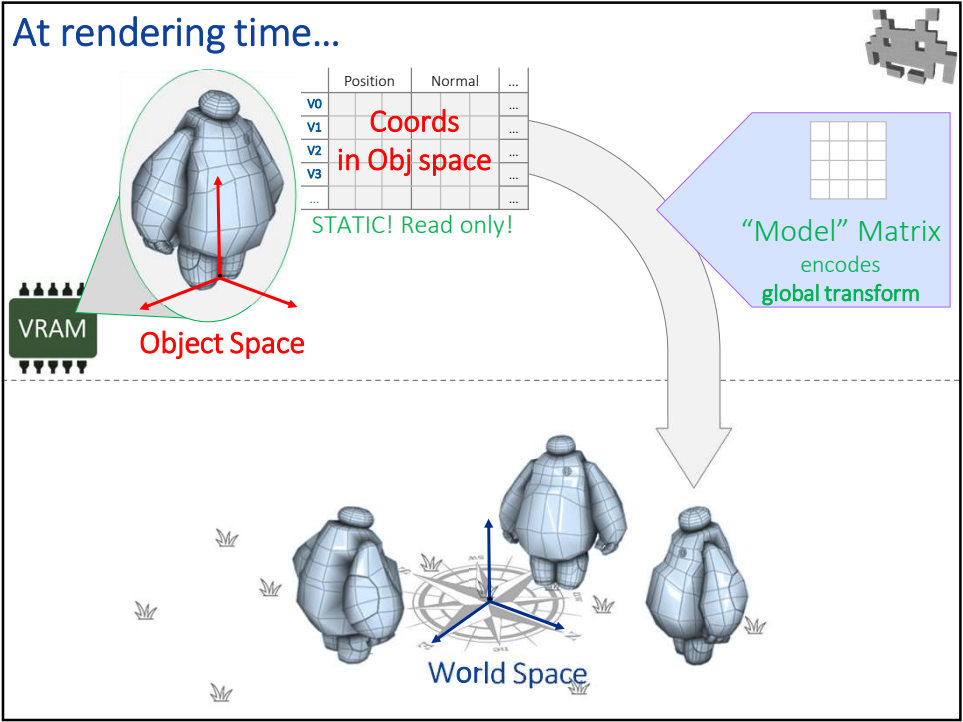
47

Mesh in VRAM

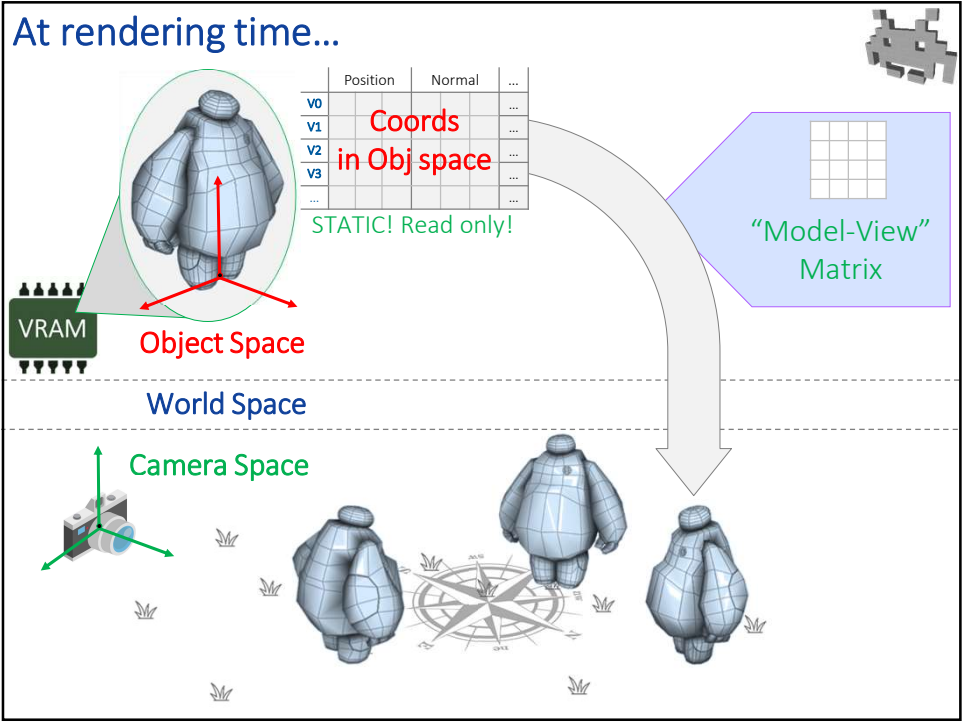
This diagram shows the data flow for a mesh into VRAM. It is divided into three vertical sections: DISK, CENTRAL RAM, and GPU RAM. In the DISK section, a green circle labeled 'Mesh File' has an arrow labeled 'IMPORT' pointing to a green circle labeled 'Mesh Object' in the CENTRAL RAM section. From there, an arrow labeled 'LOAD' points to a green circle labeled 'Mesh GPU Object' in the GPU RAM section. A red arrow with a yellow outline points to the 'Mesh GPU Object'.

- Mesh stored as VRAM buffers
 - Named (by Graphics APIs):
Vertex Buffer Object (VBO) or **Vertex Arrays** (VA)
- Coords (vertex pos, normal) in object space (by def)
- VRAM is a scarce, precious resource!
- Choices for a Game Engine:
 - storage formats, including precisions for each attribute, and how many bits per index. e.g.:
 - color? 8 bits per channel is usually fine
 - position? is 16 bit float per coordinate enough?
 - Vertex indices? Is 16 bit enough?
 - trade-off between storage cost / accuracy

49



50



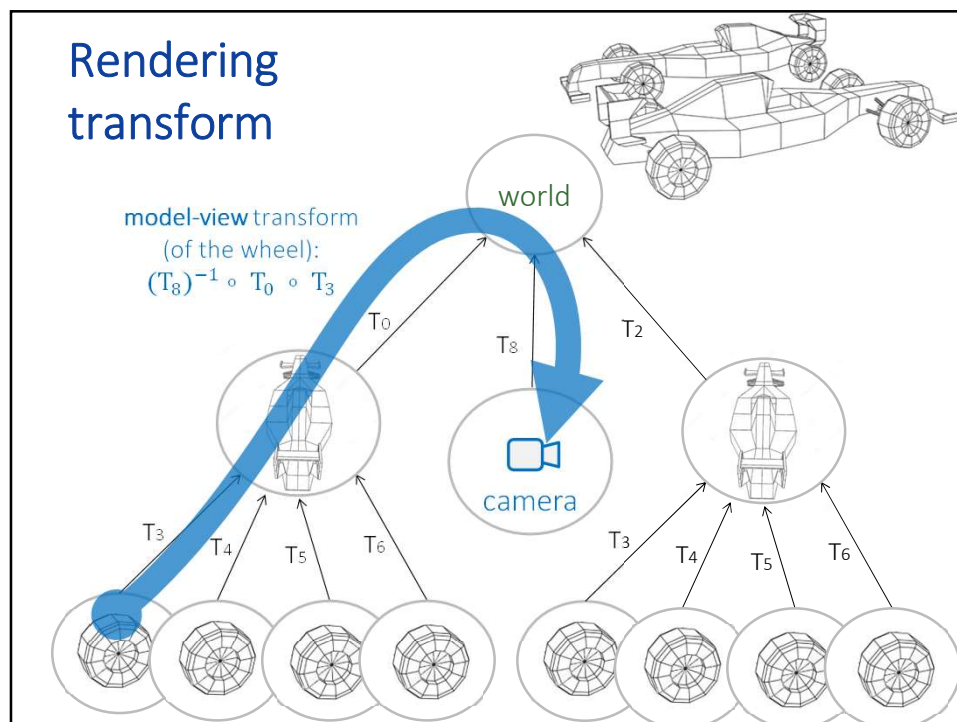
51

Rendering transforms

- The mesh buffers, once loaded in VRAM, are **static**.
 - When the object is rigid, at least
 - (Not a rope / a dynamic cloth for example)
 - The only thing that changes between frames is the associated **transform**
 - The transform:
 - Is sent to the GPU, to be applied to each vertex
 - Is applied on the fly to vertex position / normals, during rendering
 - Is typically expressed as a **4x4 matrix**, which is ideal for the task (it's the quickest to apply)
 - Names of that matrix (in CG jargon):
 - model matrix**: from object space to global space
 - model-view matrix**: from object space to **view space**
- that is, the matrix expressing the global transform.*
- The local space of the camera. Ideal, for the rendering*

52

Rendering transform



53

Mesh as an asset

- A file (or se of file) of a given format sitting on the disk
- Choices for the game engine:
 - which formats(s) to use?
 - which attributes are needed?
 - Etc.
- Issues:
 - storage cost
 - loading time

55

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: 12, # edges: 10, # vertices: 40

x,y,z 2nd vertex: 3 0 0

index 0: 0 0 0, index 1: 3 0 0, index 2: 3 1 0, index 3: 1 1 0

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

56

Most common file formats for meshes in games



.OBJ (wavefront)

- ☉ very broad diffusion
- ☉ mesh basics: indexed, normals, uv-mapping
- ☉ trivial to parse / read
- ☉ simple materials too
- ☉ material index for face (no colors)
- ☉ no skinning, animations, scenegraph...

.SMD (VALVE)

- ☉ Skeletal animations + skinning
- ☉ normals, uv-mapping
- ☉ meshes: not indexed, no colors...

.MD3 (Quake, IDsoft)

- ☉ good for blendshapes
- ☉ meshes: no colors

.PLY (cyberware)

- ☉ customizable
- ☉ "academic"

.DAE (collada: SONY + KHRONOS)

A format for the Exchange of Digital Assets

- ☉ complete:
 - particle systems, physics attributes, scenegraph, skinned meshes, blend shapes, geometric proxies...
- ☉ open standard
- ☉ too complete? to parsing it completely

.FBX (AUTODESK)

- ☉ abbastanza
- ☉ proprietary

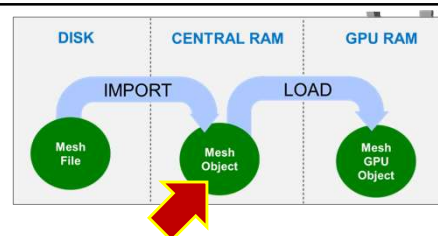
.glTF (Chronos, opensource)

Graphic Library Transmission Format
A dump of memory structures for OpenGL rendering

- ☉ very complete, and customizable
- ☉ open standard
- ☉ includes animations, etc

59

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)

61

Data structure for a mesh (to be used, e.g., for processing)

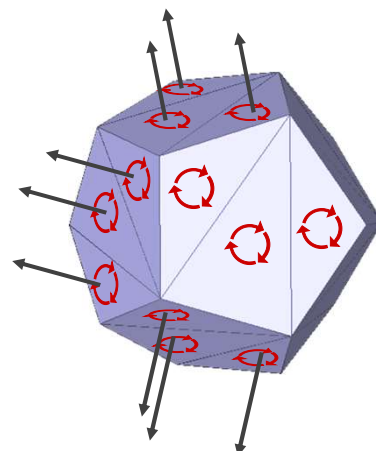
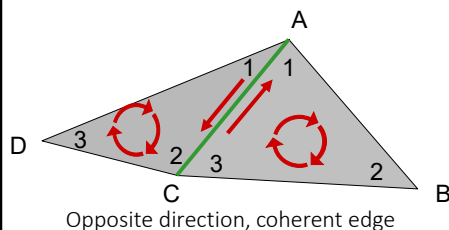
- Indexed mode in C++ :

```
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 */  
};
```

62

Compute normals from geometry

- Note:
the faces **orientation**
must be **coherent**



63

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



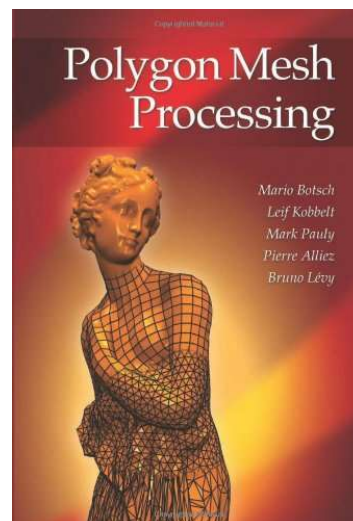
- The algorithm above for the computation of per vertex normal is one example of processing done over a mesh
- **Mesh processing**: the discipline of generating, processing, meshes
 - Algorithms having inputs and/or outputs as meshes
- See CG course for a very brief overview

67

Mesh processing: (or more in general Geometry Processing)



- A good textbook to mesh processing



68

C++ libraries for mesh processing




VCG-Lib

vision and
computer graphic library

CNR ()



computational geometry
algorithms library

INRIA ()




OpenMesh

+



OpenFlipper

RWTH ()

libigl



simple geometry
processing library

NYU ()

69

Mesh processing: a software suite



MeshLab

MeshLab

- open-source,
- A big collection of geometry processing algorithms
- ...

70

Mesh processing: examples of tasks commonly employed in games



- 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

71

3D Models: sources. How to we get our 3D models?



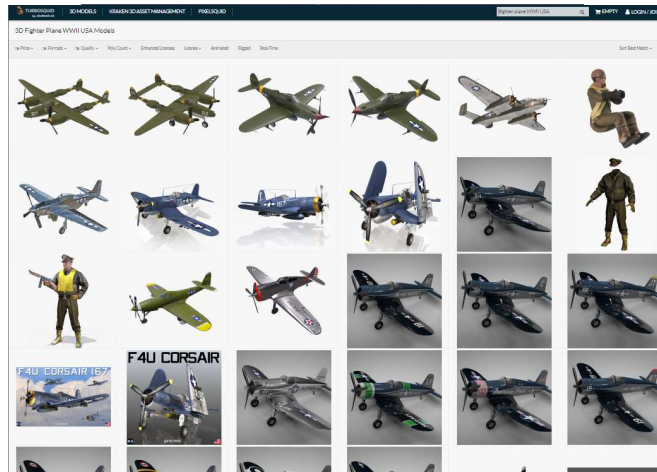
Will we'll discuss...

- **Modelling** by **digital artists**
 - Note: the 3D modelling phase is just a step of the **Asset Creation pipeline**
- **Procedural modelling**
 - As for any asset, the procedural approach is an option
 - Same trade-offs as usual for prececurality
 - Note: results can be baked (during production), or, generation can happen at game execution
- **3D acquisition**
 - Scanning a real world, physical model

73

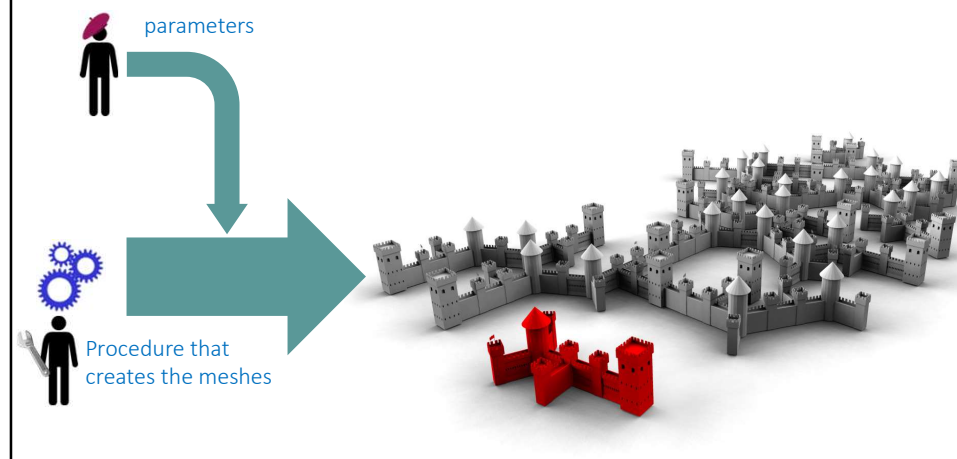
3D models: sources

- Or, like any asset, can be just bought / off-sourced
 - Try looking any repository for a given type of object



74

Sources for 3D models: procedural modelling



75

Procedural modelling – see also...

Everything Procedural
Conference 2020

EPC2021
EVERYTHING PROCEDURAL
CONFERENCE ON PROCEDURAL CONTENT GENERATION FOR GAMES
30-04-2021

EPC2022
EVERYTHING PROCEDURAL
CONFERENCE ON PROCEDURAL CONTENT GENERATION FOR GAMES
22-04-2022
MASTERCLASS 21-04-2022

EPC 2023
EVERYTHING PROCEDURAL
CONFERENCE ON PROCEDURAL GENERATION FOR GAMES

EVERYTHING PROCEDURAL
EPC2024
15-19 APRIL 2024
CONFERENCE ON PROCEDURAL GENERATION FOR GAMES

EPC 2025
CONFERENCE ON PROCEDURAL GENERATION FOR GAMES
TUESDAY APRIL 22 - FRIDAY APRIL 25



this week
Game-of-the-Week

76

3D models:
manual modelling

concept
artist







2D concepts
/ sketches



3D modeller





low-poly
mesh

77

Marco Tarini
Università degli studi di Milano

13

Mesh modelling

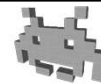


- Task of the **3D modeller**
 - A type of digital artist
- Different approaches:
 - Direct **low-poly modelling**
 - Potentially, using **subdivision surfaces** too
 - **Digital sculpting**



78

Mesh modelling: a few popular suites



used in classroom demos

- **3D Studio Max** (autodesk) ,
Maya (autodesk) ,
Cinema4D (maxon)
Lightweight 3D (NewTek),
Modo (The Foundry) , ...
 - all-purpose, powerful, complete
- **Blender**
 - the same, plus open-source and freeware
 - compare: Gimp VS. Adobe Photoshop for 2D images
- **AutoCAD** (autodesk),
SolidWorks (SolidThinking)
 - for CAD
- **ZBrush** (pixologic)
(+ **Sculptris alpha**, a toy),
Mudbox (autodesk)
 - Sculpting (including texturing)
- **Wings3D**
 - low-poly modelling (& subdivision surfaces)
open-source, small, specialized
- **[RhinoCeros]**
 - parametric surfaces (NURBS)
- **FragMotion**
 - small, specialized on animated meshes
- + a many more for specific contexts
 - editing of human models, of architectural interiors, environments, or specific editors for game-engines, etc...

79

Low-poly modelling (demo)

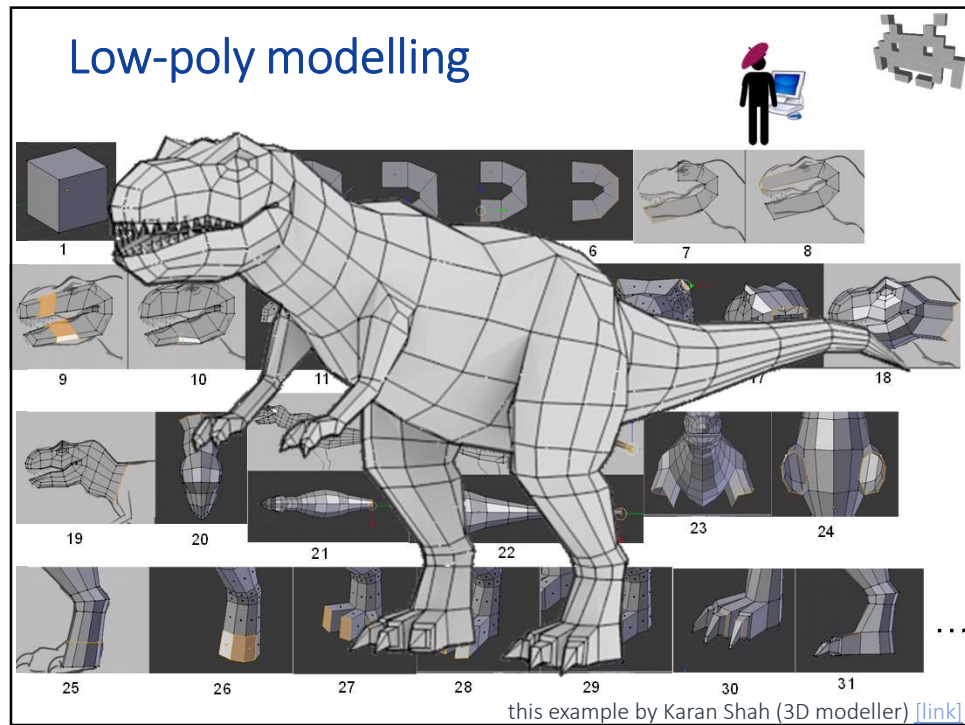
Note: during creation, the meshes can be polygonal instead of triangle based, but is simple to decompose any polygon into triangles.
(can be done by the game engine as a simple preprocessing)

80

Low-poly modelling



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

82

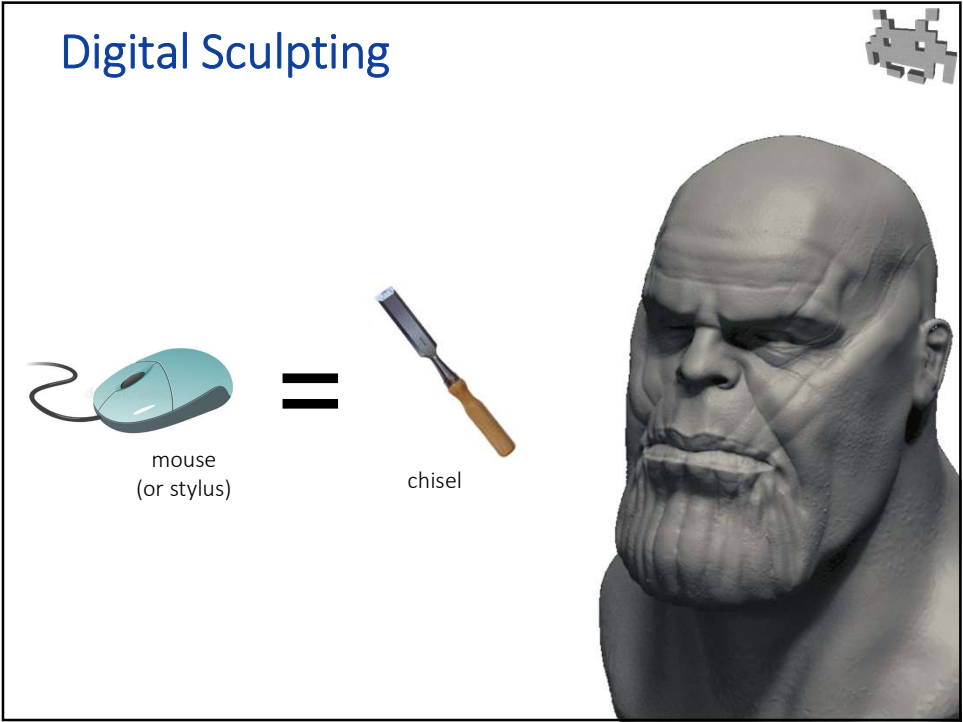


83

Mesh modelling

- Task of the **3D modeller** 
 - A type of digital artist
- Different approaches:
 - Direct **low-poly modelling**
 - Potentially, using **subdivision surfaces** too
 - **Digital sculpting** 

103



104

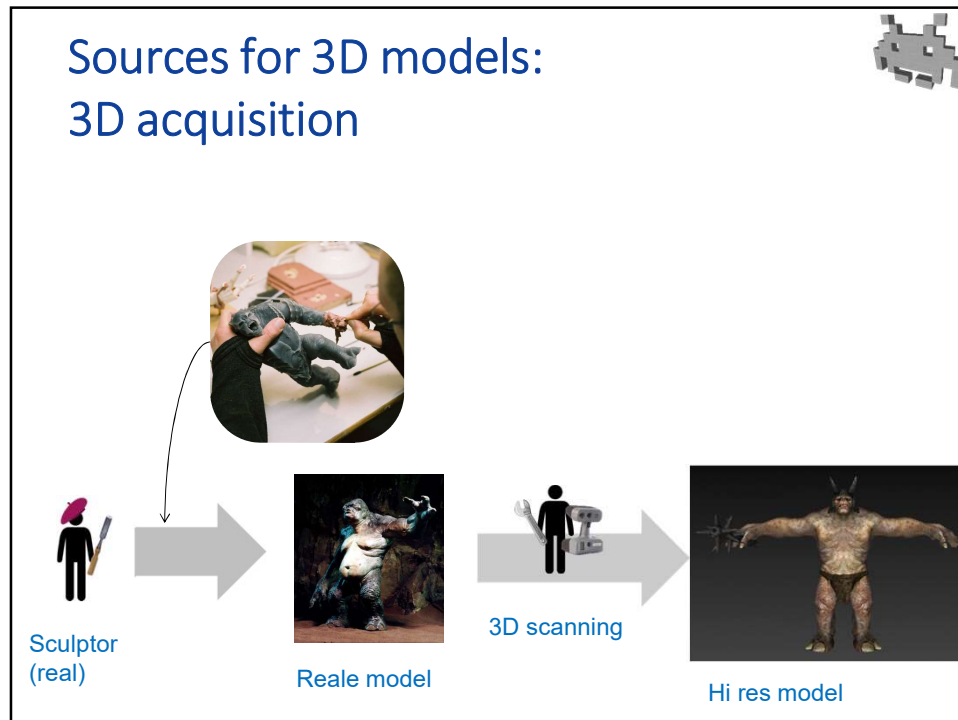
Sources for 3D models:
3D acquisition

For more info, see **Computer Graphics** course

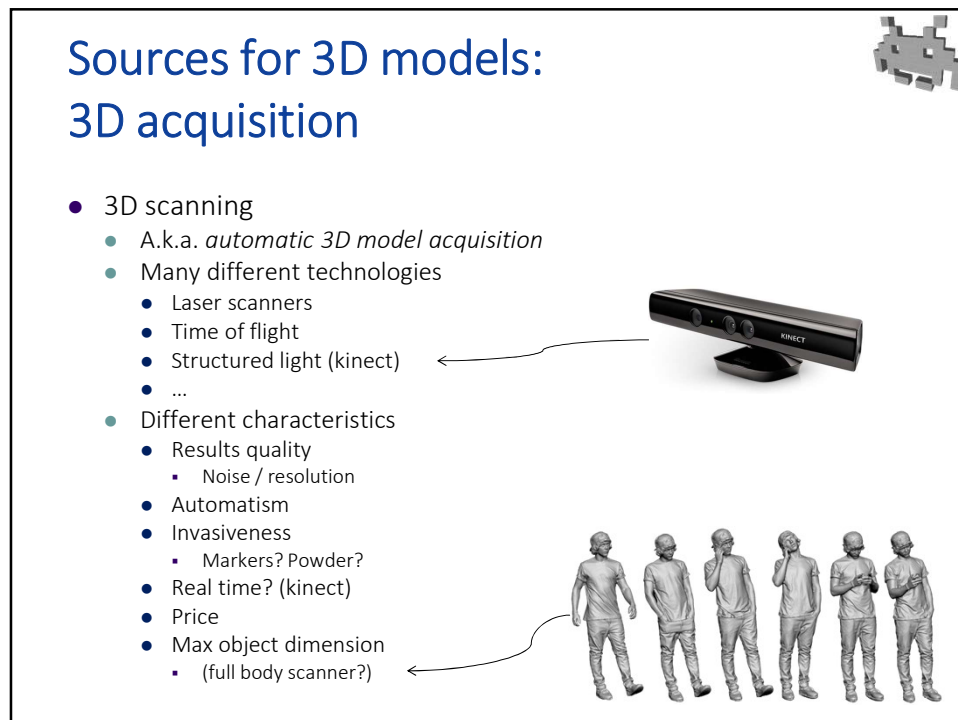
- 3D acquisition / 3D scanning
 - Technologies for obtaining 3D digital models from real-world objects

3D acquisition
(e.g., range scanning)
(specifically, here, laser scanning)

106



107



108

Example: a repository of production-ready 3D meshes for games scanned from real objects



The screenshot shows the Quixel Megascans web interface. On the left is a sidebar with a list of categories under 'Essential' and '3D', including 'Swords and Daggers' which is highlighted. The main area displays a grid of various scanned weapons like swords, daggers, and axes. At the top right of the interface is a small 3D wireframe cube icon.

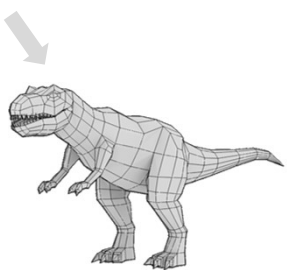
<https://quixel.com/>

of course, scanning is just one phase of the Production pipeline

109


3D models sources:
a comparison

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



manually edited
low-poly mesh
(2K triangles)

VS



Dino,
scanned
by artec3d

scanned & cleaned
hi res mes
(30K triangles)

(sculpted meshes are similar)

110

Notes about mesh resolution

- Costs: **linear** on the triangles number
 - in memory (disk, CPU RAM, GPU RAM)
 - in time (rendering, loading, etc)
- (also, **linear** with number of vertices)
 - (because, as a rule of thumb: n verts $\rightarrow 2n$ tris)
- adaptive resolution is possible in a mesh
 - higher-res (more numerous, smaller triangles) in some parts
 - lower-res (sparser vertices, connected by larger triangles) in others

111

Summary:
mesh as buffers
(tables in GPU ram)

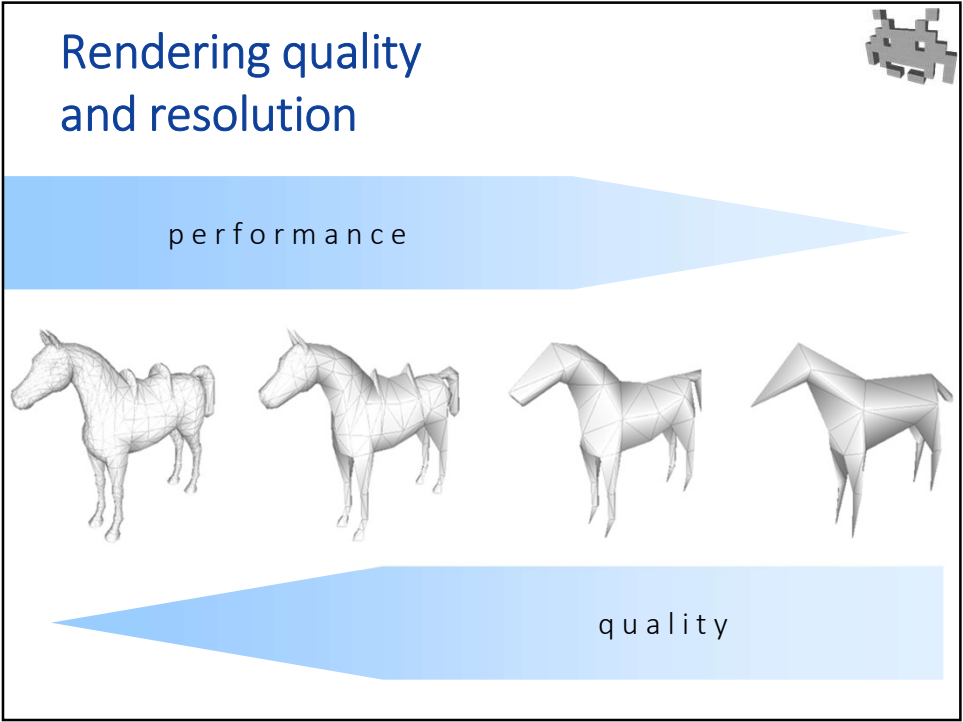
tri:	W0:	W1:	W2:
T0	The mesh "connectivity"		
T1			
T2			
T3			
T4			
T5			
...			

INDEX BUFFER

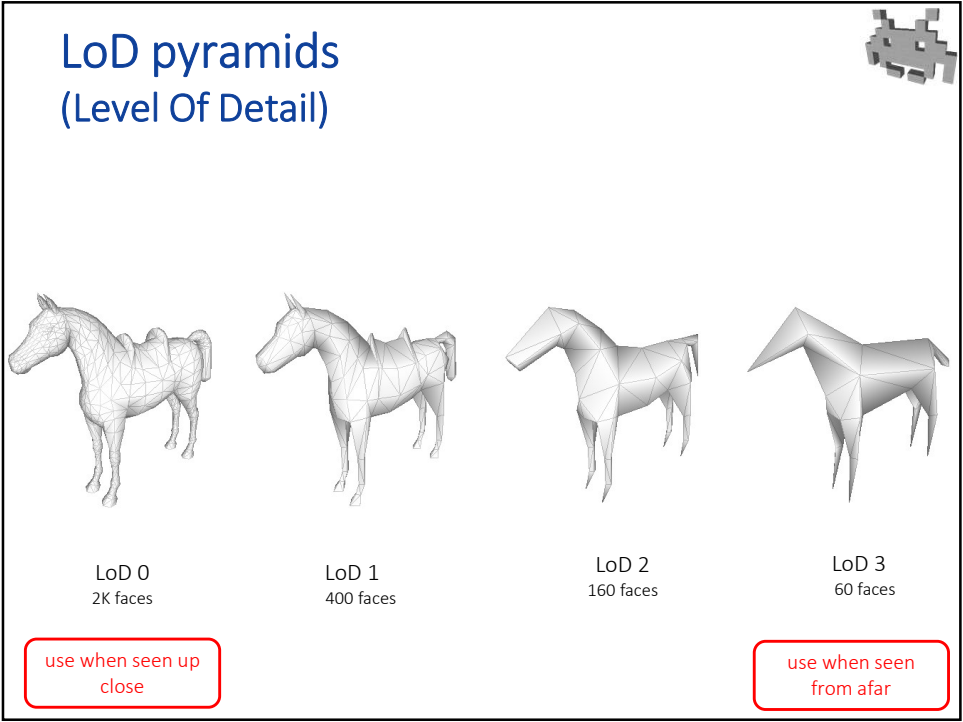
	vertex positions			vertex normals			vertex colors			other attributes							
vert	Px	Py	Pz	Nx	Ny	Nz	Cr	Cg	Cb
V0	the mesh "geometry"	per-vertex normals	per-vertex colors	...													
V1																	
V2																	
V3																	
V4																	

VERTEX BUFFER (Geometry + Attributes) – or buffers (e.g. one per attribute)

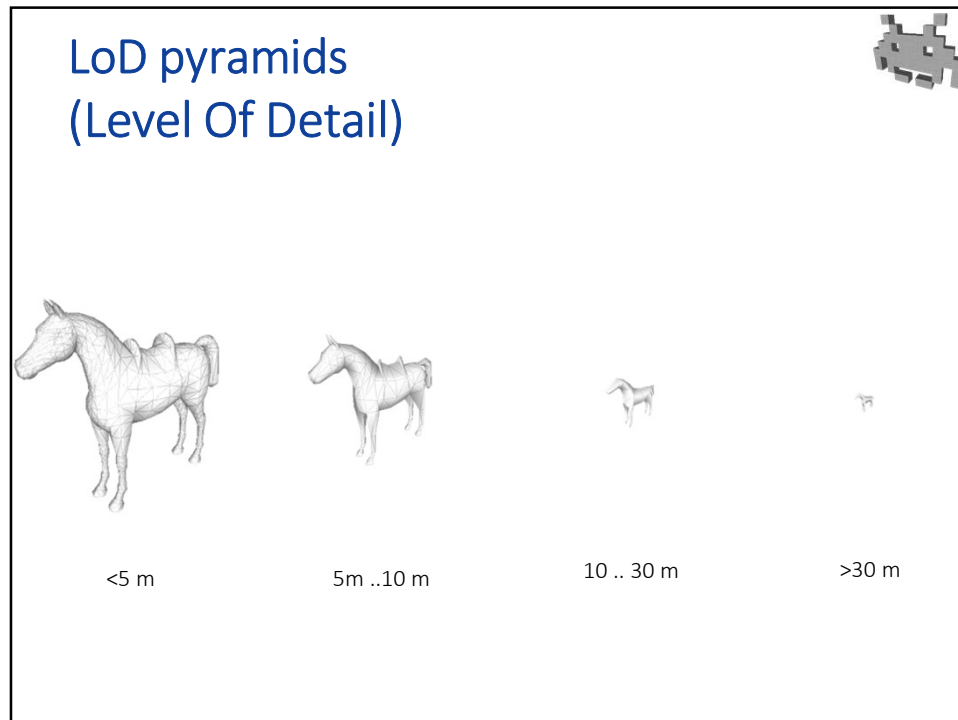
112



113



114



115

LoD pyramids (Level Of Detail)

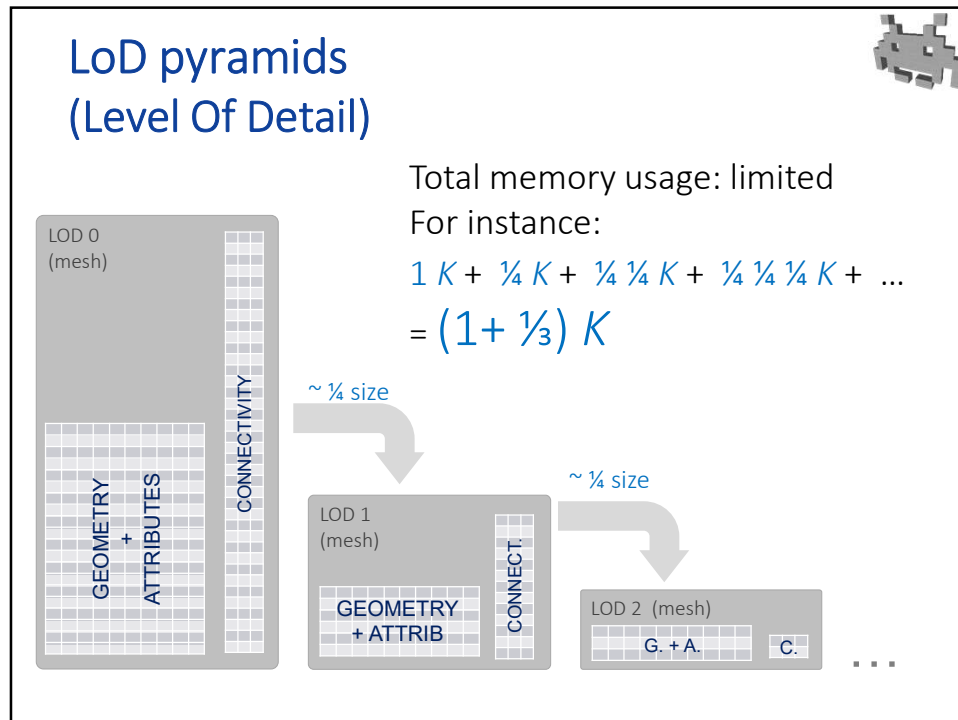
The diagram illustrates four levels of detail for a horse model, each corresponding to a specific distance range. The models are shown in a wireframe style, with the number of visible triangles decreasing as the distance increases. The distance ranges are labeled below each model: <5 m, 5m ..10 m, 10 .. 30 m, and >30 m. A small, pixelated horse icon is visible in the top right corner of the slide.

Distance Range	Level of Detail
<5 m	High detail (many triangles)
5m ..10 m	Medium detail (fewer triangles)
10 .. 30 m	Low detail (very few triangles)
>30 m	Very low detail (minimal triangles)

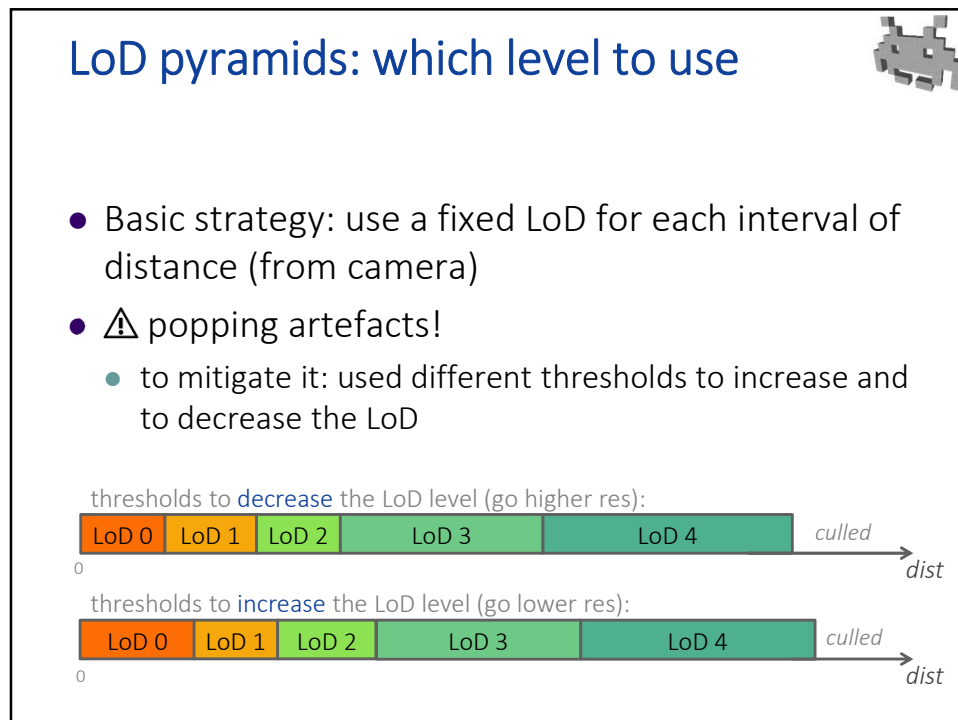
- Goal:
 - decrease the **geometry budget** (total number of vertices)
 - ideal: size of triangles in screen space (in pixel): constant
 - importance / geometrical complexity being the same
- Task: determining the level to use (**dynamically**, at runtime)
 - depending on observer distance
 - and/or, depending on rendering workload
 - e.g.: rendering is lagging \Rightarrow decrease LoD
 - this is task of the rendering engine
- Task: LOD creation or “LOD-ding” (during **asset creation**)
 - starting from LOD-0 (higher-res)
 - manual, or **automatic**
 - difficult to automatize well, for very coarse LODs
 - note: sometimes “LoD-0” is used only in special cases
 - e.g., for cut-scenes

computed from scene graph (how?)

116



117

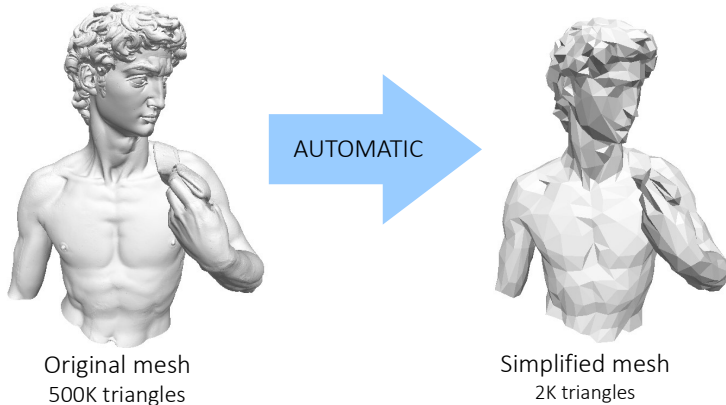


119

Poly-reduction

(aka Mesh “simplification” / “decimation” / “coarsening”)

- parameters:
 - a maximum error
 - or number of faces objective



Original mesh
500K triangles

Simplified mesh
2K triangles

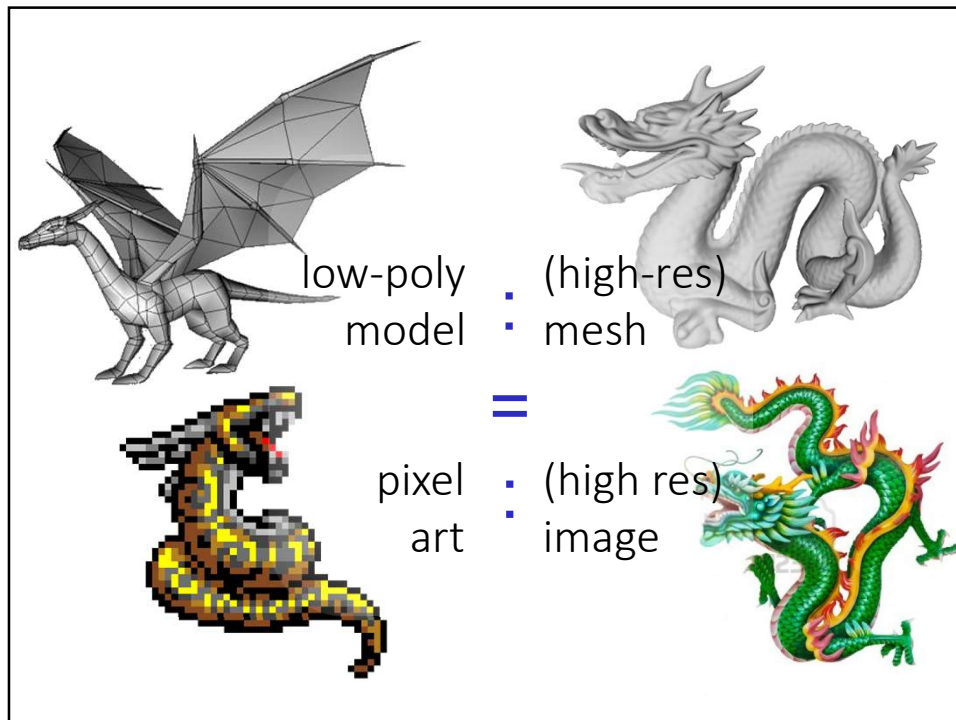
120

Poly-reduction

(aka Mesh “simplification” / “decimation” / “coarsening”)

- Different approaches are studied (in Geometry Processing)
 - Can be adaptive or not
 - Adaptive = strive to use more triangles where needed
 - Maximum error introduced:
 - can be measured and/or limited
 - or not
 - Topology (e.g. holes, handles):
 - can be kept
 - or not
 - Streamable
 - Possible
 - or not
 - ...

121



122

Emerging new formats for hi-res mesh



- **Nanite** (EPIC GAMES)
- **Micro-Meshes** (NVIDIA)

Completely different internal structures, similar objectives:

- Cheaper per-triangle VRAM cost
 - Compressed, but
 - on-the-fly decompression during rendering
- Cheaper per-triangle rendering cost
 - Micro-Meshes: intended for ray-tracing too
- Multiresolution, that is, *intrinsic* LODs
 - We can decide *on the fly* which LOD to show
 - Nanite only: the resolution level can vary over the mesh
- Reduced need for UV-maps (see next lecture)

123

NANITE: the concept

- A tree of *patches*
 - 1 patch = small *optimized* mesh of 128 tris

```
graph TD; LOD2((LOD-2: 128▲)) --> LOD1L((LOD-1: 128▲)); LOD2 --> LOD1R((LOD-1: 128▲)); LOD1L --> LOD0L1((LOD-0: 128▲)); LOD1L --> LOD0L2((LOD-0: 128▲)); LOD1R --> LOD0R1((LOD-0: 128▲)); LOD1R --> LOD0R2((LOD-0: 128▲));
```

124

NANITE: the concept

- A tree of *patches*
 - 1 patch = small *optimized* mesh of 128 tris

```
graph TD; LODi1((LOD_(i+1): 128▲)) --> LODiL((LOD_(i): 128▲)); LODi1 --> LODiR((LOD_(i): 128▲));
```

125

NANITE: the concept

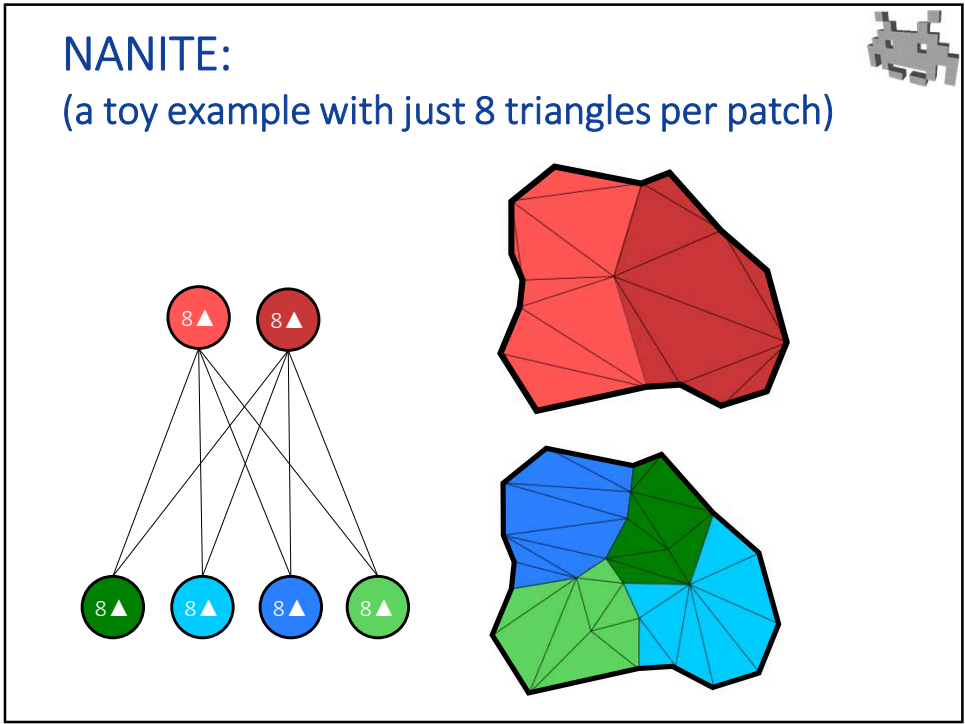
- A tree of *patches*
 - 1 patch = small *optimized* mesh of 128 tris

126

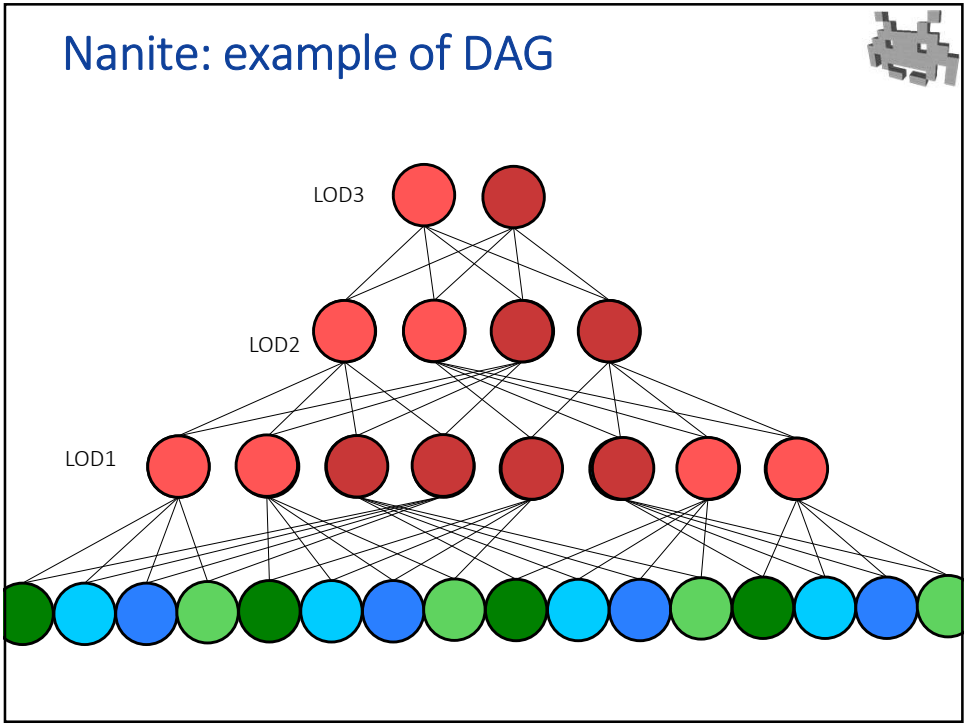
NANITE: in reality

- A ^{DAG}~~tree~~ of *patches*
 - 1 patch = small *optimized* mesh of 128 tris

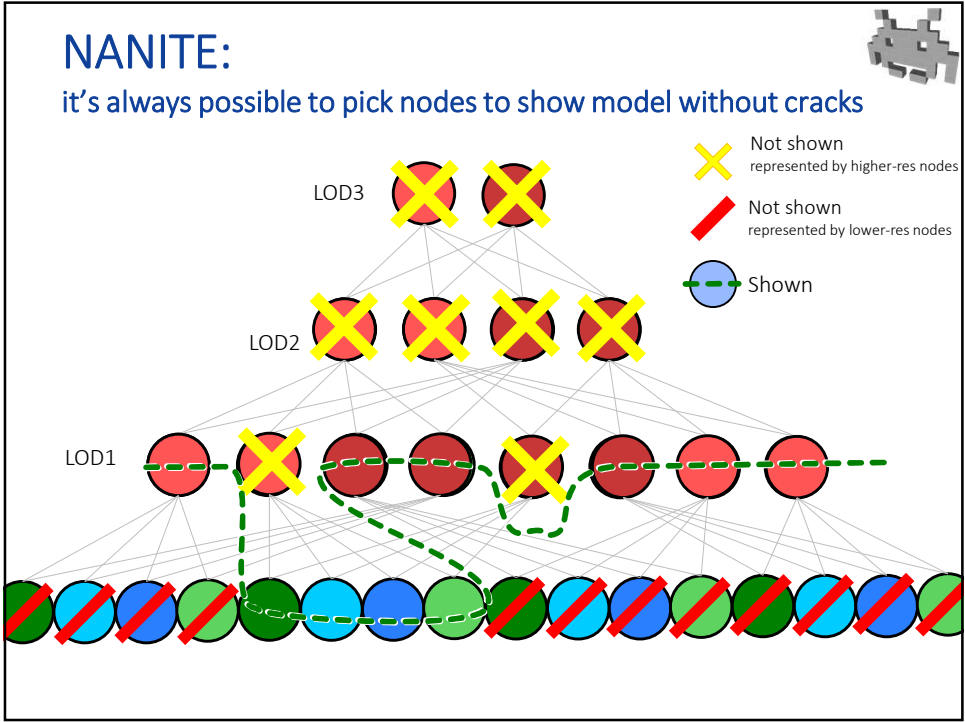
127



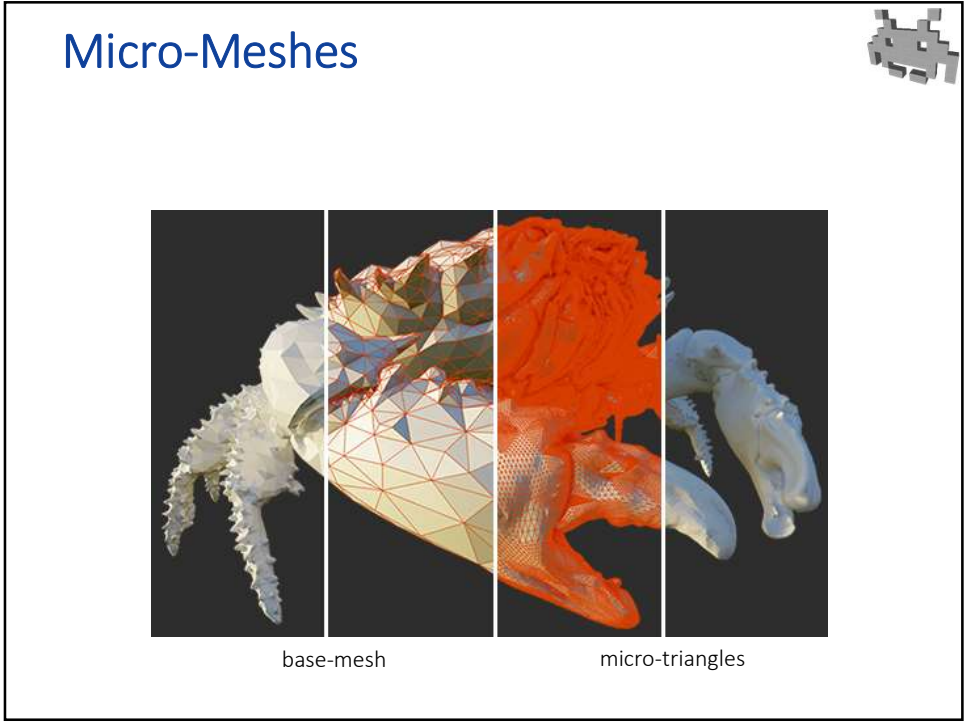
128



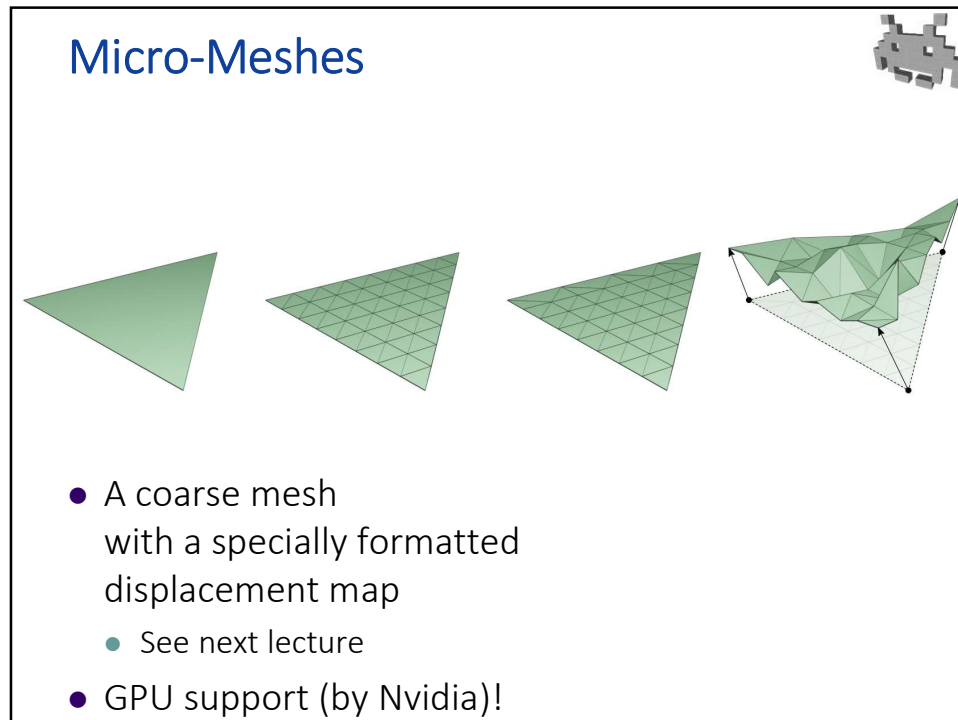
130



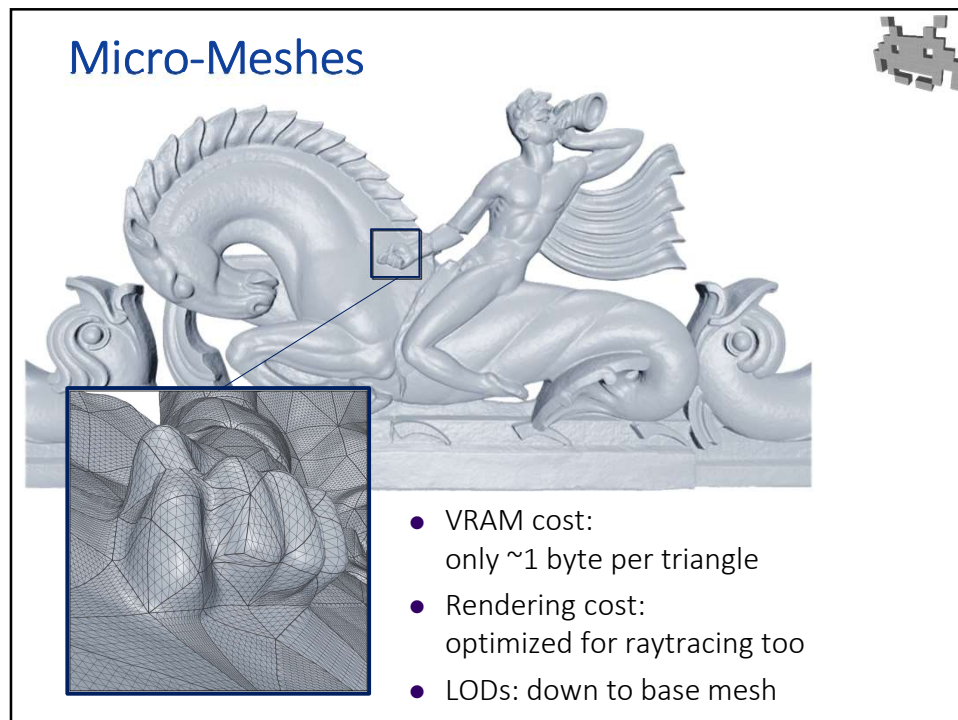
131



133



134



135