


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 Textures** ●●
- lec. 9: **Game Materials** ●
- lec. 8: **Game 3D Animations** ●●●
- lec. 10: **Networking** for 3D Games ●
- lec. 11: **3D Audio** for 3D Games ●
- lec. 12: **Rendering Techniques** for 3D Games ●
- lec. 13: **Artificial Intelligence** for 3D Games ●

★ appearance

106

Summary: mesh as buffers (tables in GPU ram)



tri:	W0:	W1:	W2:
T0			
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)

107

Rendering quality and resolution

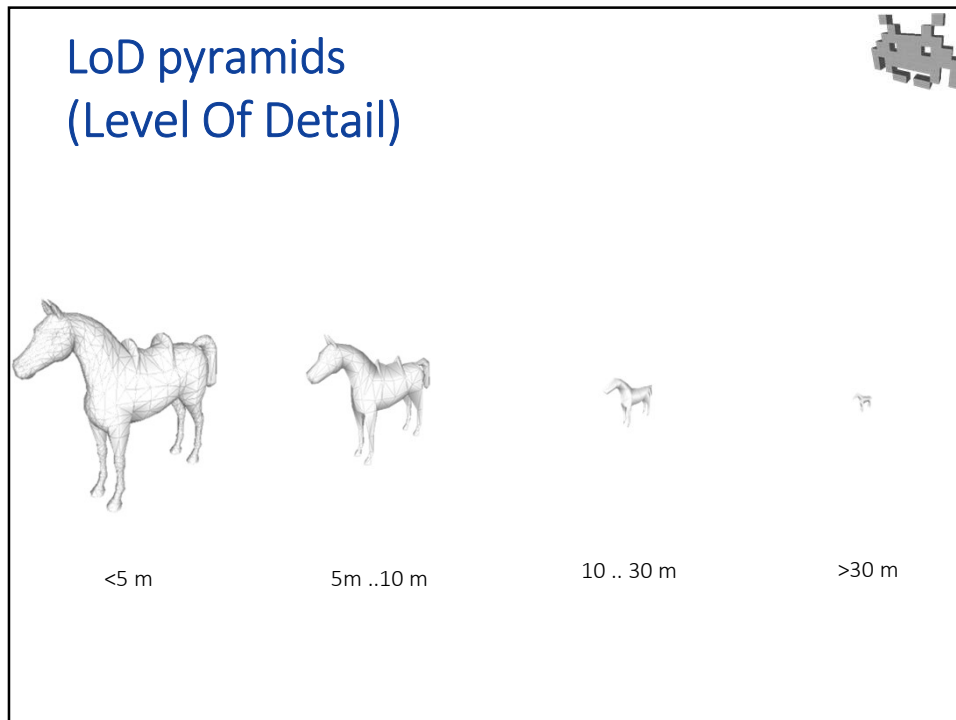
The diagram illustrates the trade-off between performance and quality for different mesh resolutions of a horse. It features a blue arrow pointing right labeled 'performance' and a blue arrow pointing left labeled 'quality'. In the center, four wireframe horse models are shown from left to right, representing increasing levels of detail and decreasing performance. A small 3D puzzle icon is in the top right corner.

108

LoD pyramids (Level Of Detail)

The diagram shows four levels of detail (LoD) for a horse mesh, from left to right: LoD 0 (2K faces), LoD 1 (400 faces), LoD 2 (160 faces), and LoD 3 (60 faces). Below each model is a red box with text: 'use when seen up close' for LoD 0 and 'use when seen from afar' for LoD 3. A small 3D puzzle icon is in the top right corner.

109



110

LoD pyramids (Level Of Detail)

- 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?)

111

LoD pyramids (Level Of Detail)

Total memory usage: limited
 For instance:

$$1 K + \frac{1}{4} K + \frac{1}{4} \frac{1}{4} K + \frac{1}{4} \frac{1}{4} \frac{1}{4} K + \dots$$

$$= (1 + \frac{1}{3}) K$$

112

LoD pyramids: which level to use

- Basic strategy: use a fixed LoD for each interval of distance (from camera)
- ⚠ popping artefacts!
 - to mitigate it: used different thresholds to increase and to decrease the LoD

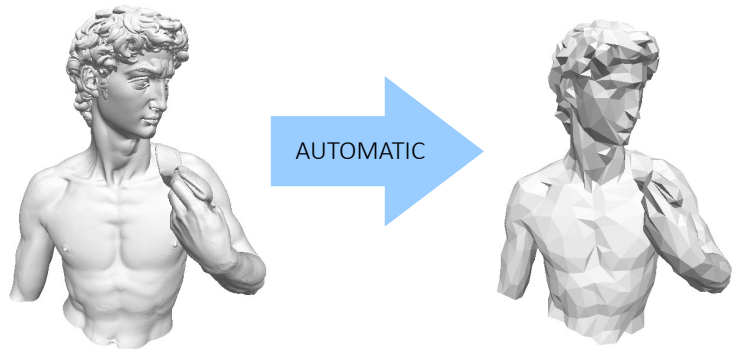
thresholds to **decrease** the LoD level (go higher res):

thresholds to **increase** the LoD level (go lower res):

114

Poly-reduction (aka Mesh “simplification” / “decimation”)

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



Original mesh
500K triangles

AUTOMATIC

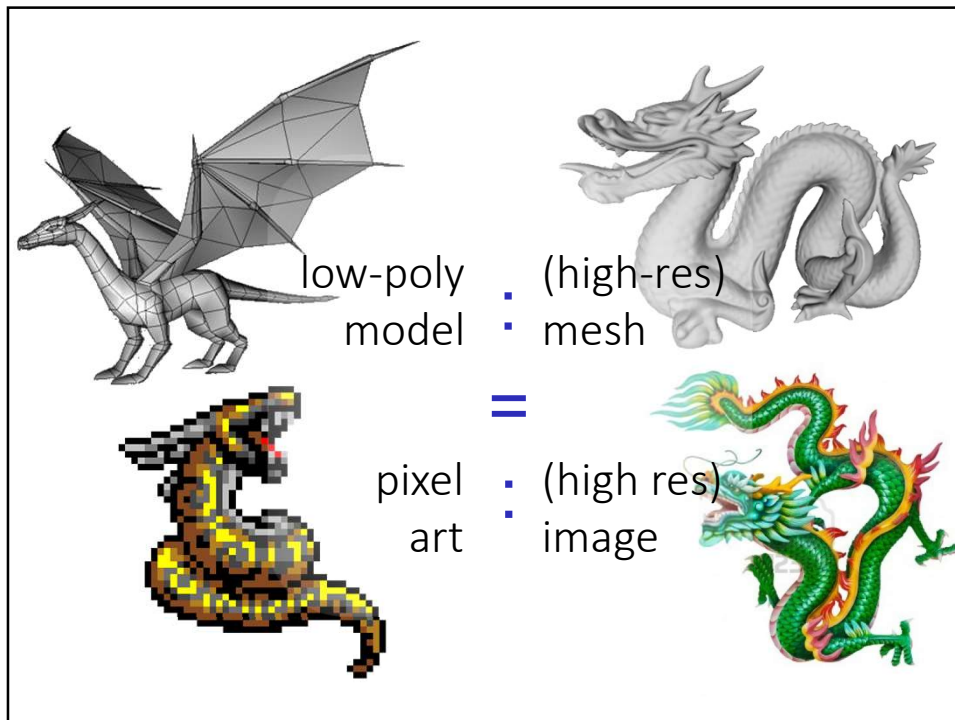
Simplified mesh
2K triangles

115

Poly-reduction (aka mesh simplification, mesh 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
 - ...

116



117

Emerging hi-res mesh formats



- **Nanite** (EPIC GAMES)
- **Micro-Meshes** (NVIDIA) ← we will see this details after textures

Completely different internal structures, common same 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)

118

NANITE: concept

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

```
graph TD; L2((LOD-2: 128▲)) --- L1L((LOD-1: 128▲)); L2 --- L1R((LOD-1: 128▲)); L1L --- L0LL((LOD-0: 128▲)); L1L --- L0LR((LOD-0: 128▲)); L1R --- L0RL((LOD-0: 128▲)); L1R --- L0RR((LOD-0: 128▲));
```

119

NANITE: concept

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

```
graph TD; L2((LOD_(i+1): 128▲)) --- L1L((LOD_(i): 128▲)); L2 --- L1R((LOD_(i): 128▲));
```

120

NANITE: concept

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

draw these for a mixed LOD

121

NANITE: in reality

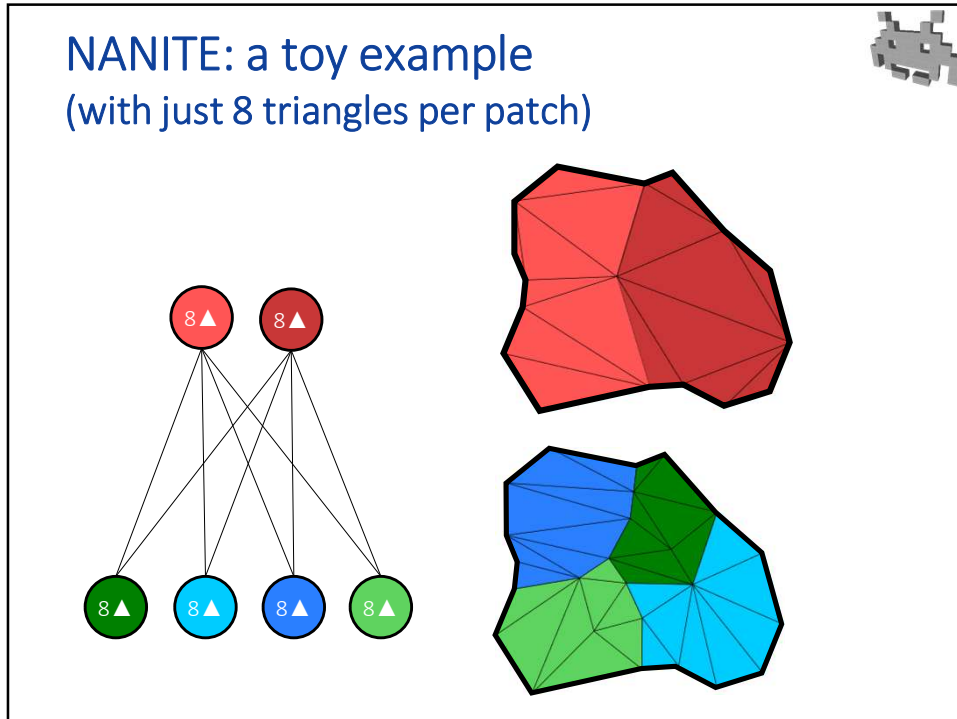
- A ^{DAG} ~~tree~~ of patches
 - 1 patch = small *optimized* mesh of 128 tris

← These 256 tris...

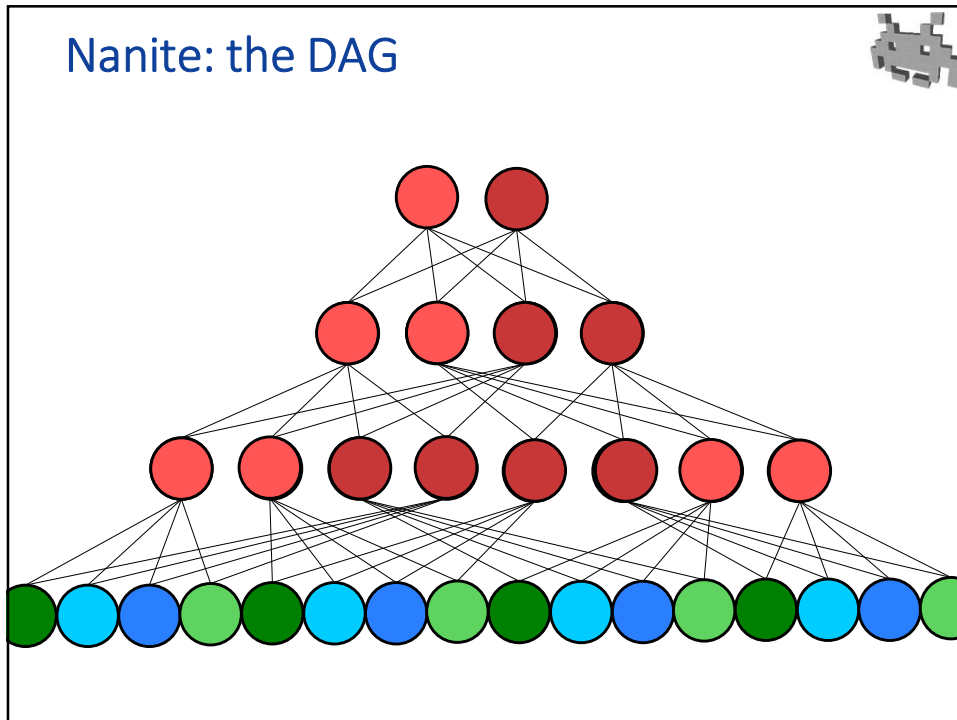
← ...approximate the same surface as these 512 tris...

... but having the **exact same boundary**

122



123



125