# An Adaptive Model for Objects Representation in 3D Computer Graphics

*A. D. Kapustin, J. G. Fedorova,*

*T.V. Firsova, A.V. Churbanov,*

# Goal and Solution

- Create software system to provide fast and quality rendering of surfaces including tens thousands polygons on PC-workstations in real-time

- Use adaptive surface representation (multi-resolution - MRM) based on multi-triangulation structure

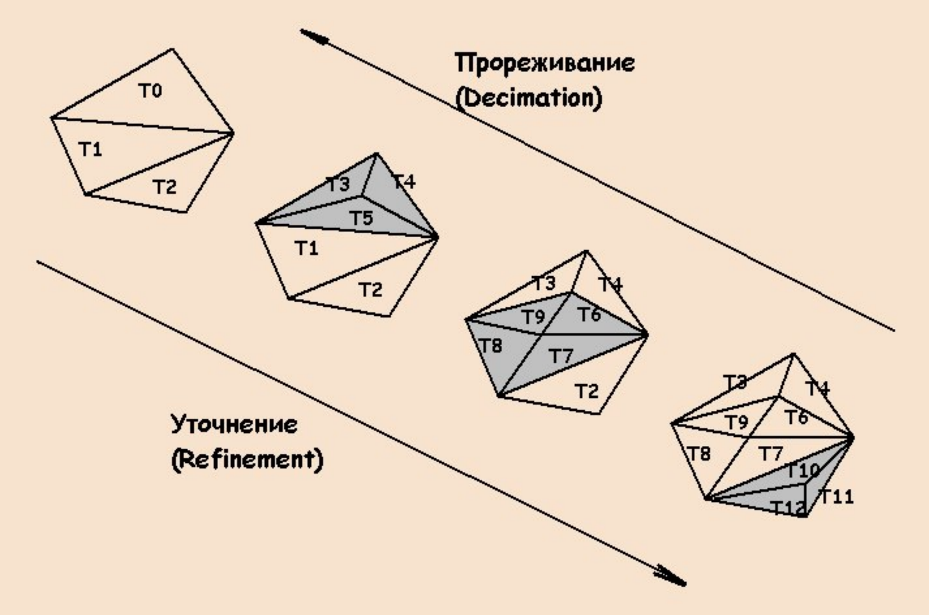
Note: Surface represented by triangulation in 3D

# Motivations

- Adaptive model structure must be suitable for various types of surfaces: terrain and free-form
- Surface LOD (approximation) extraction algorithm should support real-time  rendering
- Provide surface view-dependent LODs with considerable reduction of polygons number
- Provide smooth transitions between sequential frames at small changes of the view parameters (e.g. viewpoint, view direction, field of view)
- Should exist a consistent and direct relationship between the input parameters to the LOD algorithm and resulting image quality

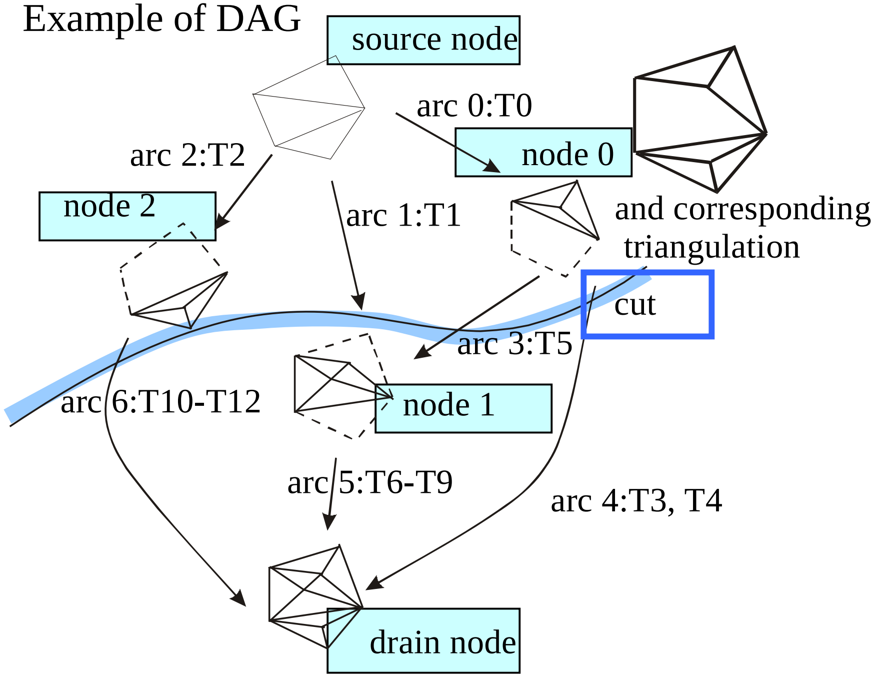
Note: LOD (level-of details)

# Multi-triangulation



Multi-triangulation is represented by Directed Acyclic Graph (DAG)

Example of DAG


source node

arc 0:T0

node 0

arc 2:T2

node 2

arc 1:T1

and corresponding triangulation

cut

arc 3:T5

arc 6:T10-T12

node 1

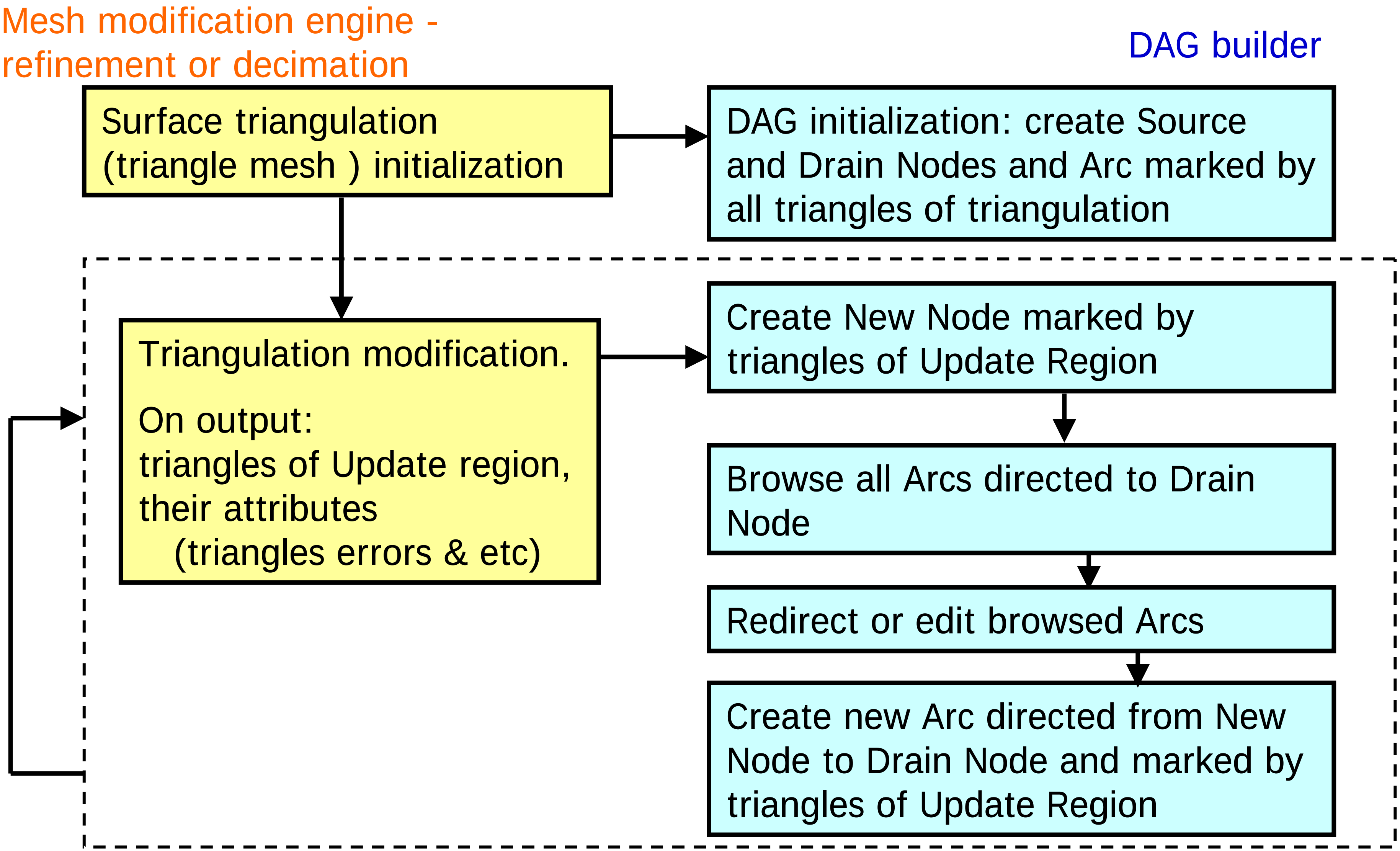
arc 5:T6-T9

arc 4:T3, T4

drain node

Multi-triangulation constructing is guided by local updates of surface triangulation

# DAG construction based on mesh modifications



Mesh modification engine - refinement or decimation

DAG builder

Surface triangulation (triangle mesh ) initialization

DAG initialization: create Source and Drain Nodes and Arc marked by all triangles of triangulation

Triangulation modification.

On output:
triangles of Update region, their attributes
(triangles errors & etc)

Create New Node marked by triangles of Update Region

Browse all Arcs directed to Drain Node

Redirect or edit browsed Arcs

Create new Arc directed from New Node to Drain Node and marked by triangles of Update Region

# Surface LOD extraction

Define in every *P* (point of space *R3*)
threshold function $\tau(P): R3 \rightarrow R$

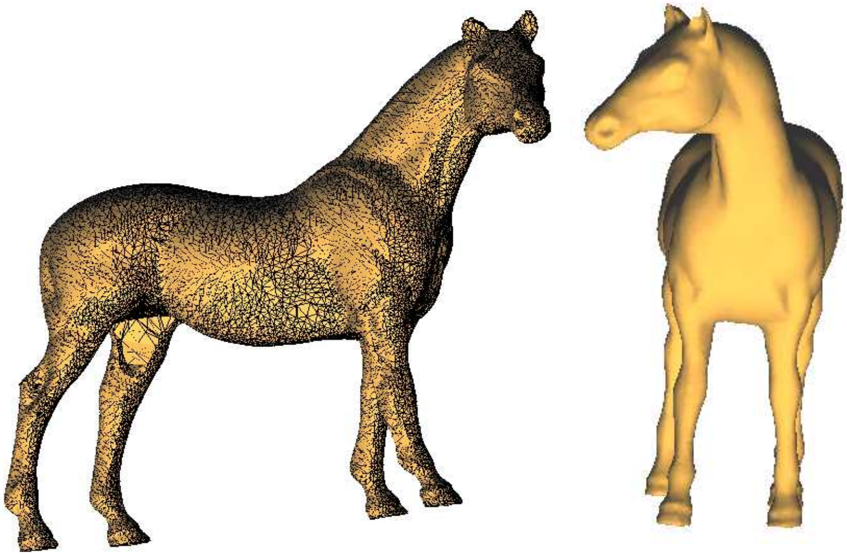
Some *Triangle* will be acceptable in the surface LOD
if ***triangleError***$< min(\tau(P))$,
*min over* $P \subset Triangle$

***triangleError*** is output of Mesh Modification algorithm
$\tau$ is a parameter of Extraction algorithm

Top-down approach

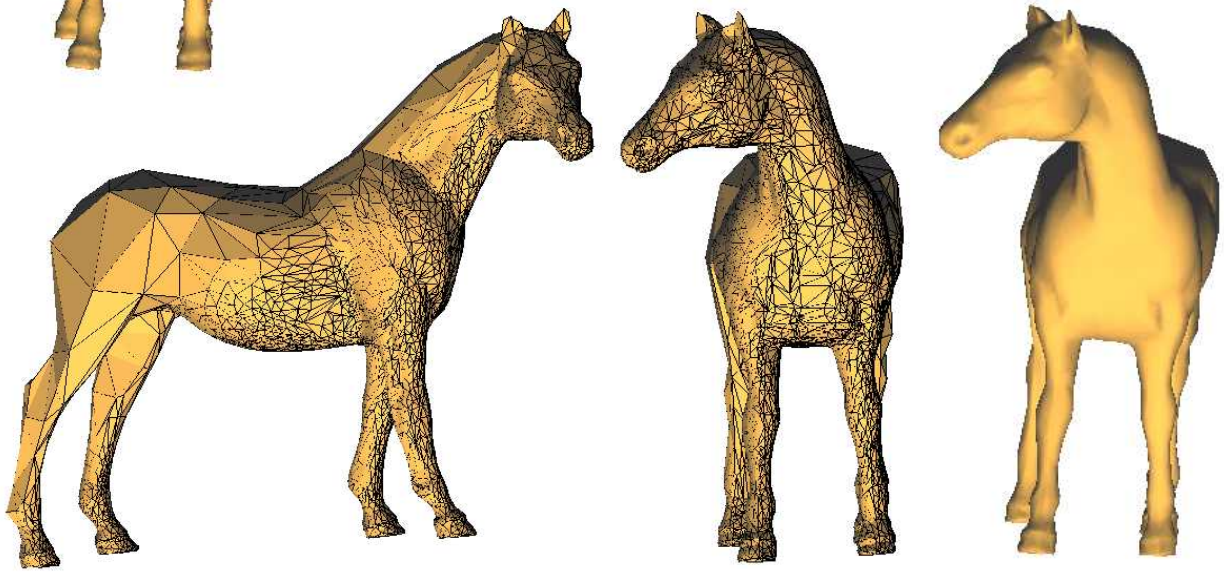
```
Initialize the Cut of the DAG by Source node triangles

For every triangle in the Cut
if $triangleError > min(\tau(P))$
  add triangle into 'Bad Triangles'Queue

While 'Bad Triangles' Queue is not empty:
    For every Triangle of the Queue:
        if $triangleError > min(\tau(P))$
          add downward triangles
           into Cut Queue
        else
          add triangle to the Cut
```

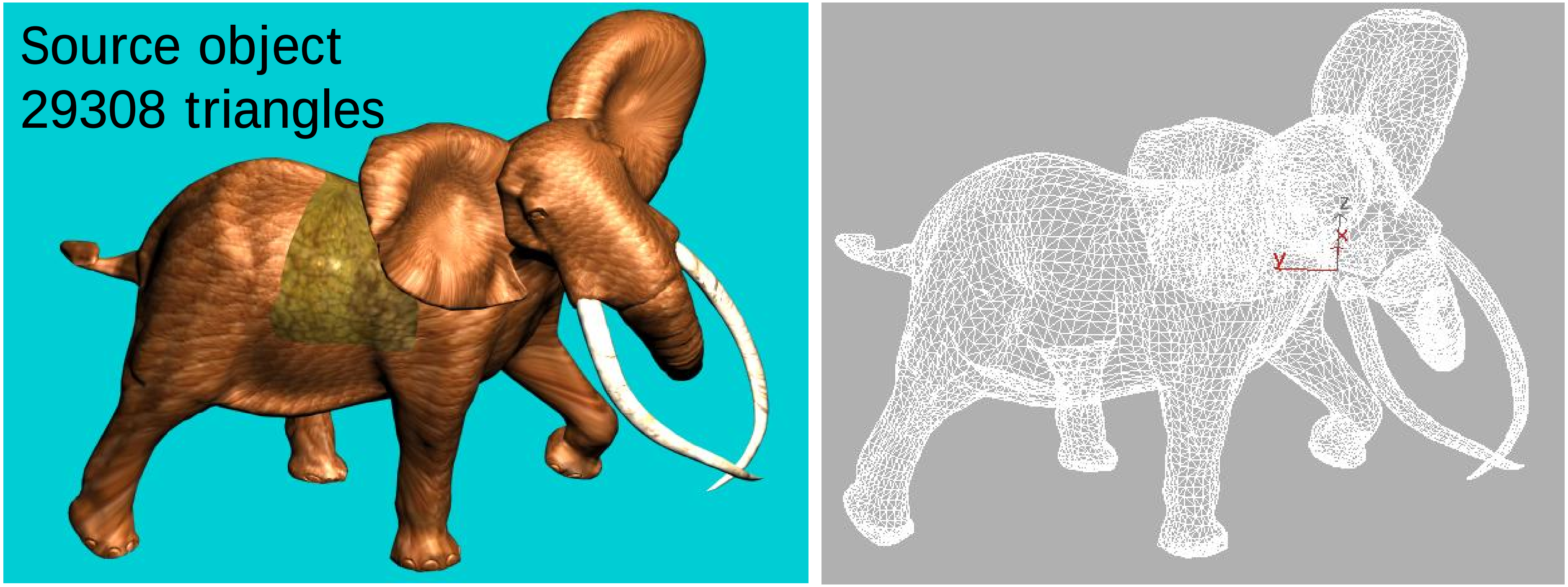
# View-dependent LOD of Free-form surface



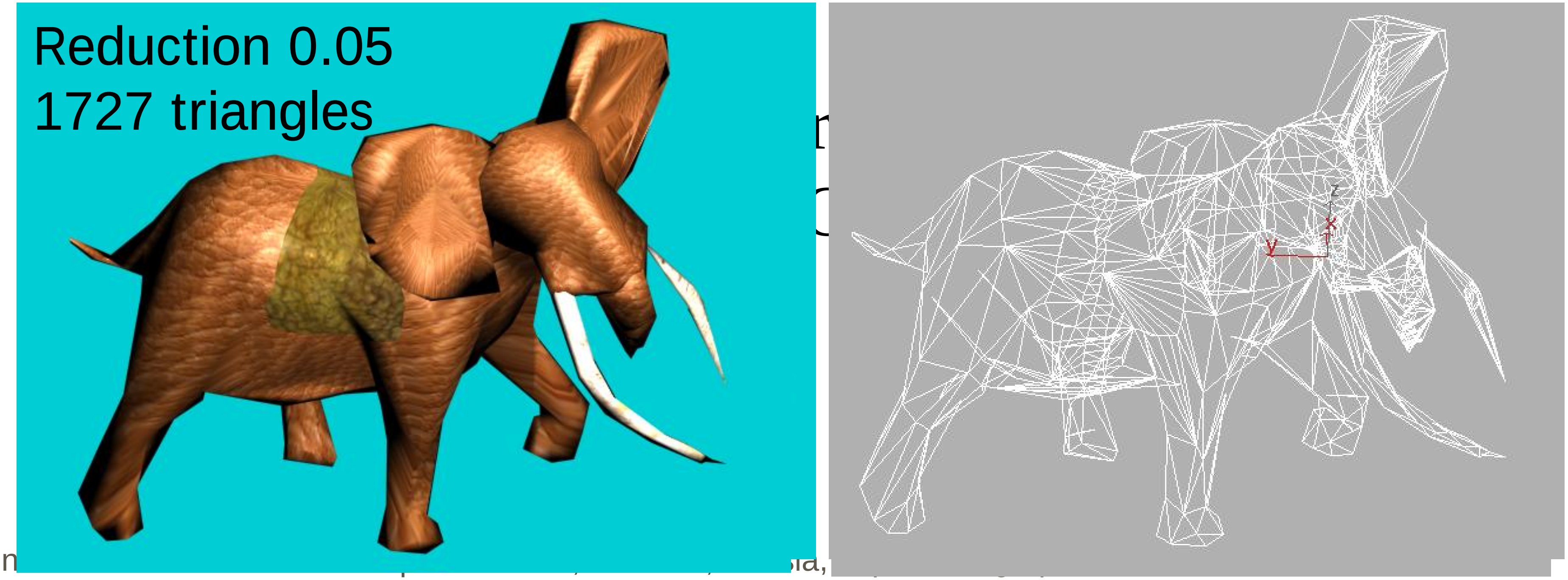
Initial model,
39700 triangles

Model LOD
at front view,
8700 triangles
reduction ~ 0.2





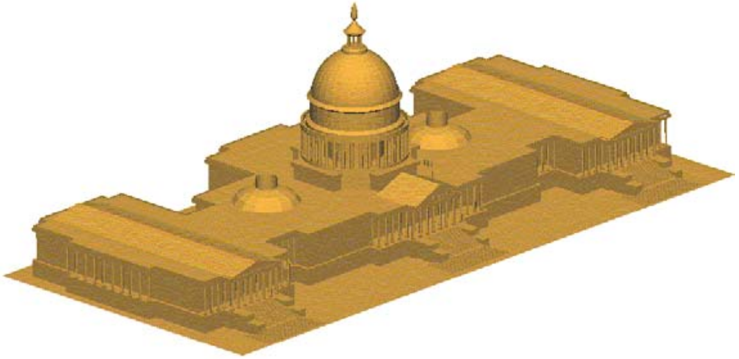
Source object
29308 triangles



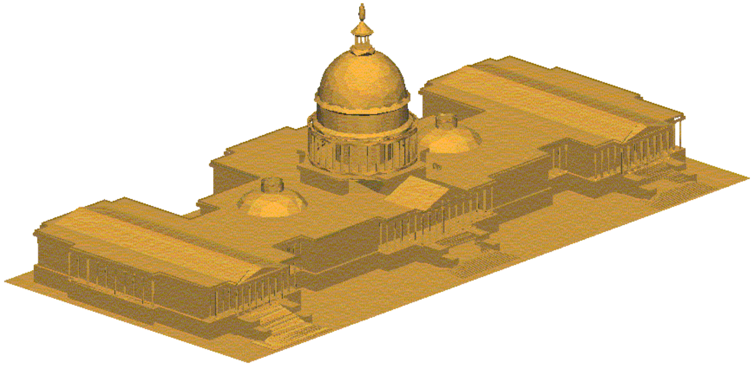
Reduction 0.05
1727 triangles

# LODs of architecture

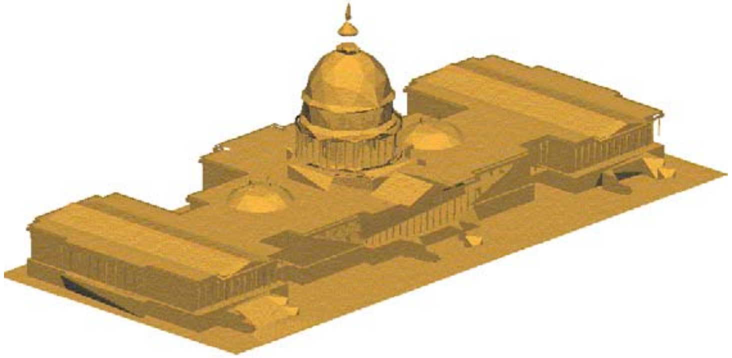
Source model
41259 triangles



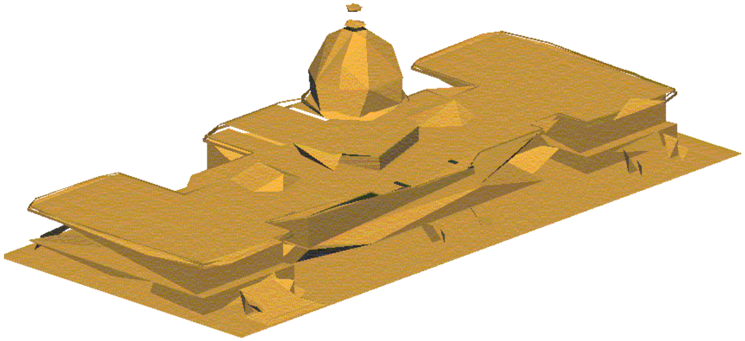
Reduction 0.2
8251 traingles



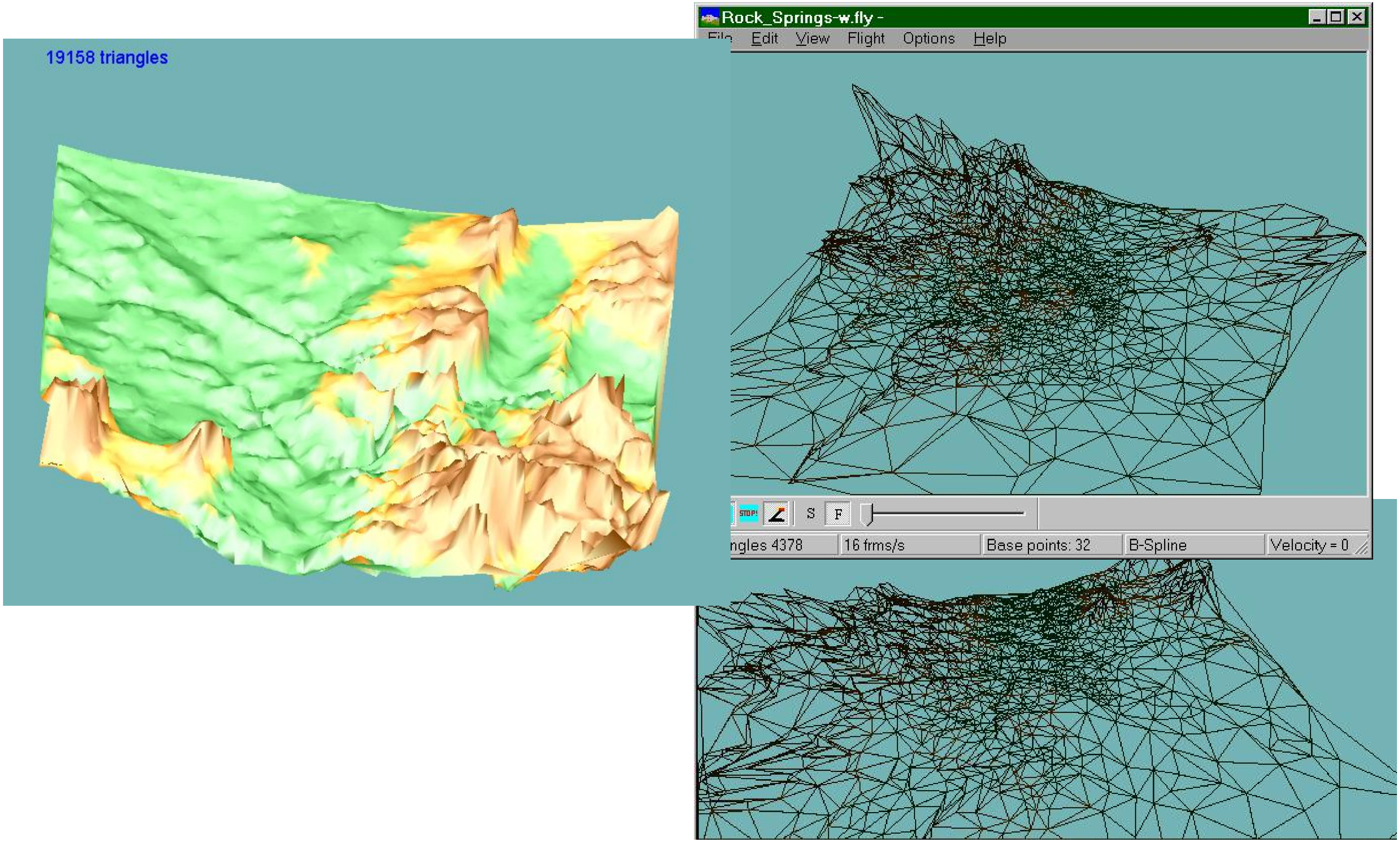
Reduction 0.05
2062 triangles



Reduction 0.01
412 triangles



# View-dependent LODs while rendering



ngles 4378 | 16 frms/s | Base points: 32 | B-Spline | Velocity = 0