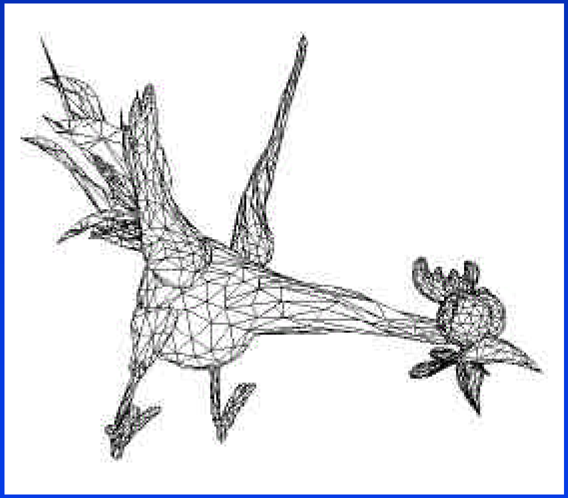
# Temporal & Spatial Level of Details for Dynamic Meshes



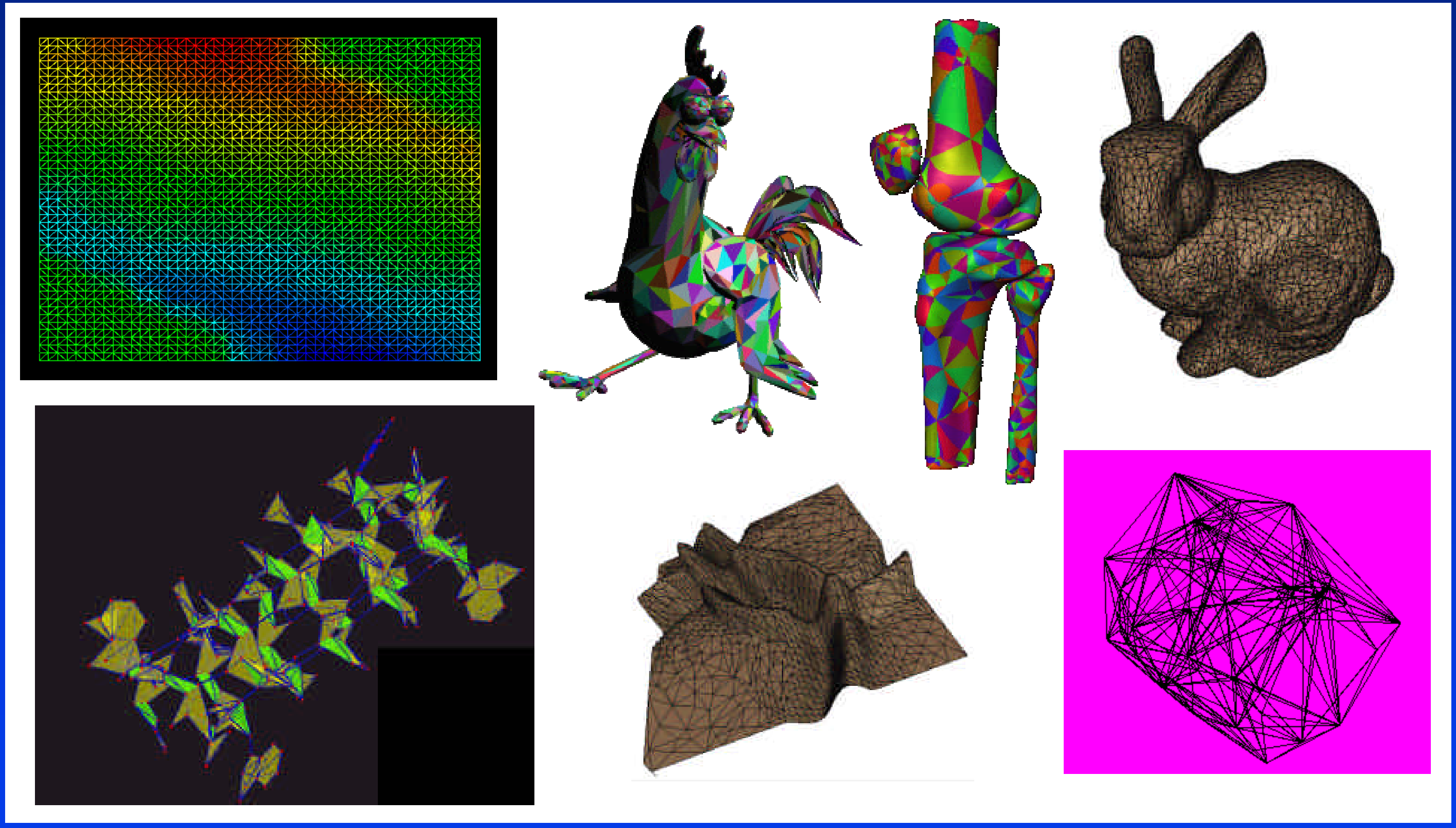

***Ariel Shamir***

***The interdisciplinary center***

***Valerio Pascucci***

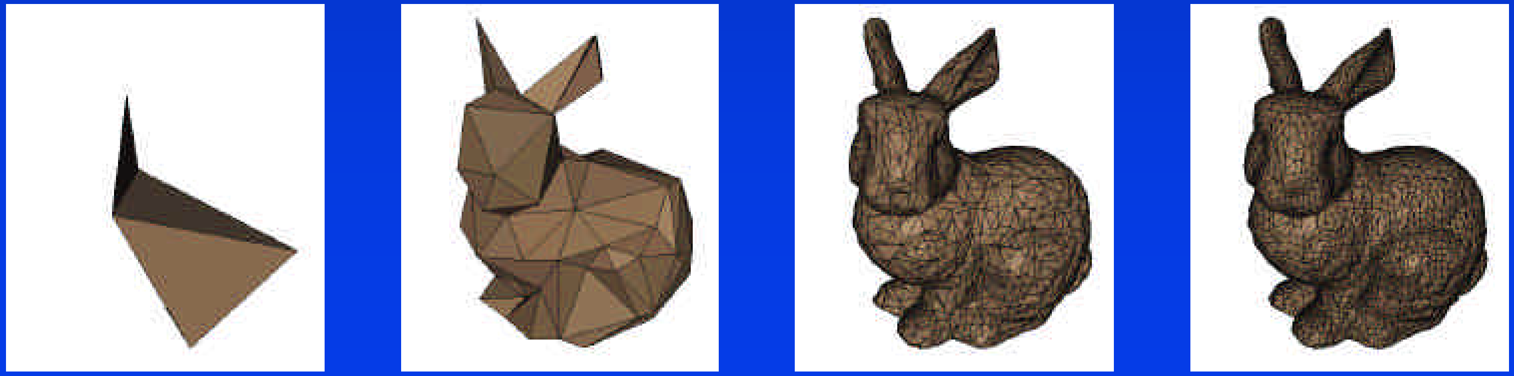
***Lawrence Livermore National Lab***

# Meshes



# Multi Resolution Representations

***An object is represented by multiple models in various resolutions, used according to time and/or space limitations***



# Multi Resolution Motivation

- *Level of detail rendering and visualization*
- *Progressive transmission*
- *Multi-scale modeling*
- *Multi-grid simulation and computation*

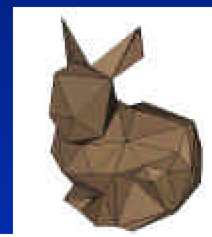
*In virtual reality and dynamic scenes time and space limitations are ever more critical!*

# Multi Resolution Scheme (Bottom Up)

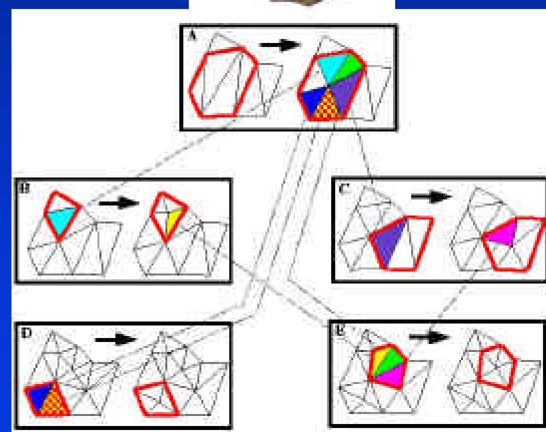
- *Define decimation primitive*
- *Define error (estimate) for decimation priority and traversal*
- *Create multi-resolution structure (decimate according to queue)*
- *Render: traverse the structure (graph)*

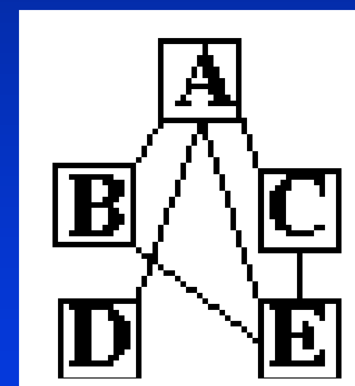
# Multi-resolution Model: DAG Creation

Low Resolution



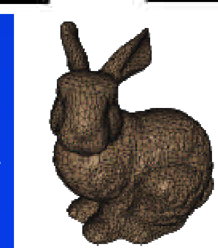
*Decimation*



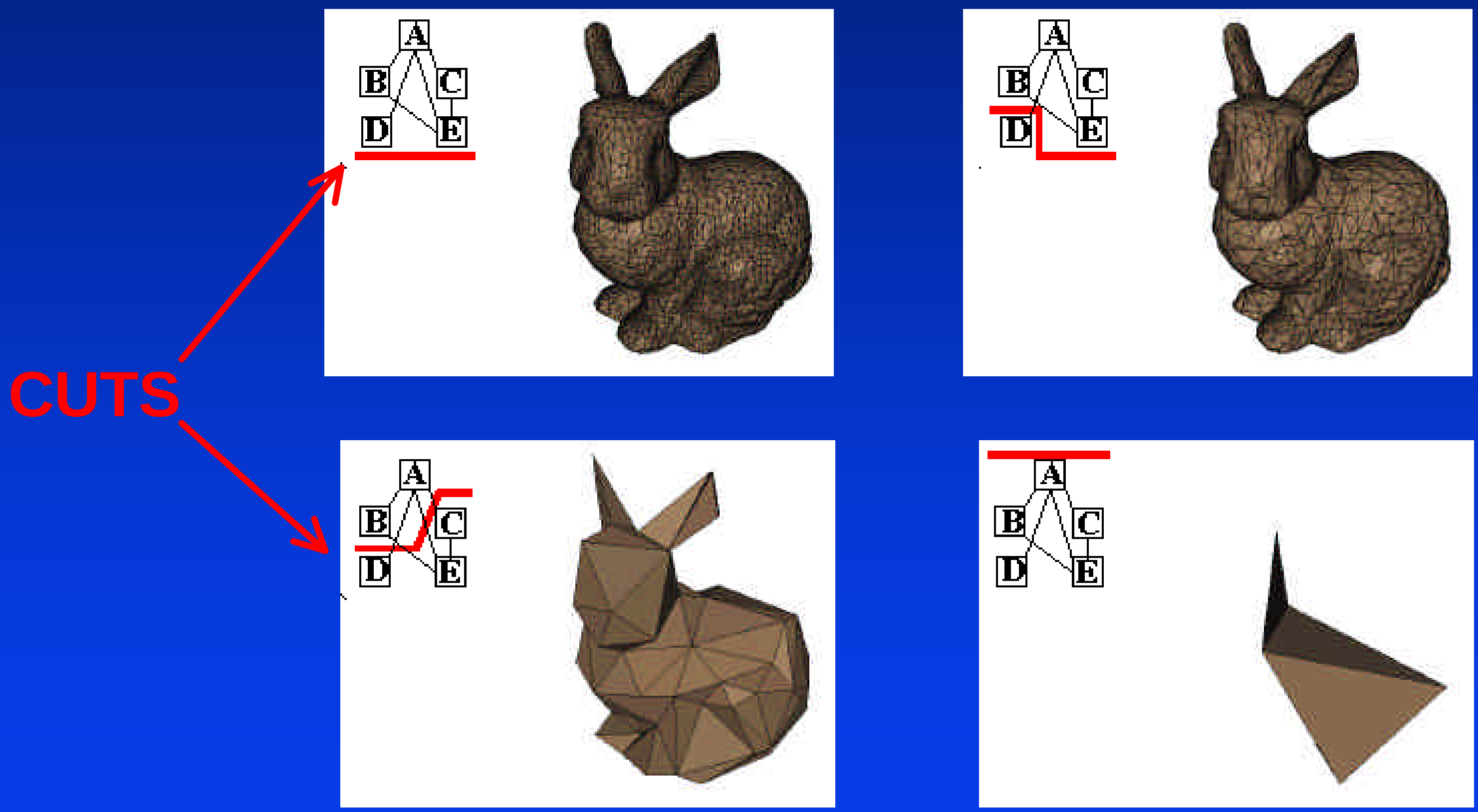



DAG

High Resolution



# Multi-resolution Model: DAG Traversal



# Previous Work

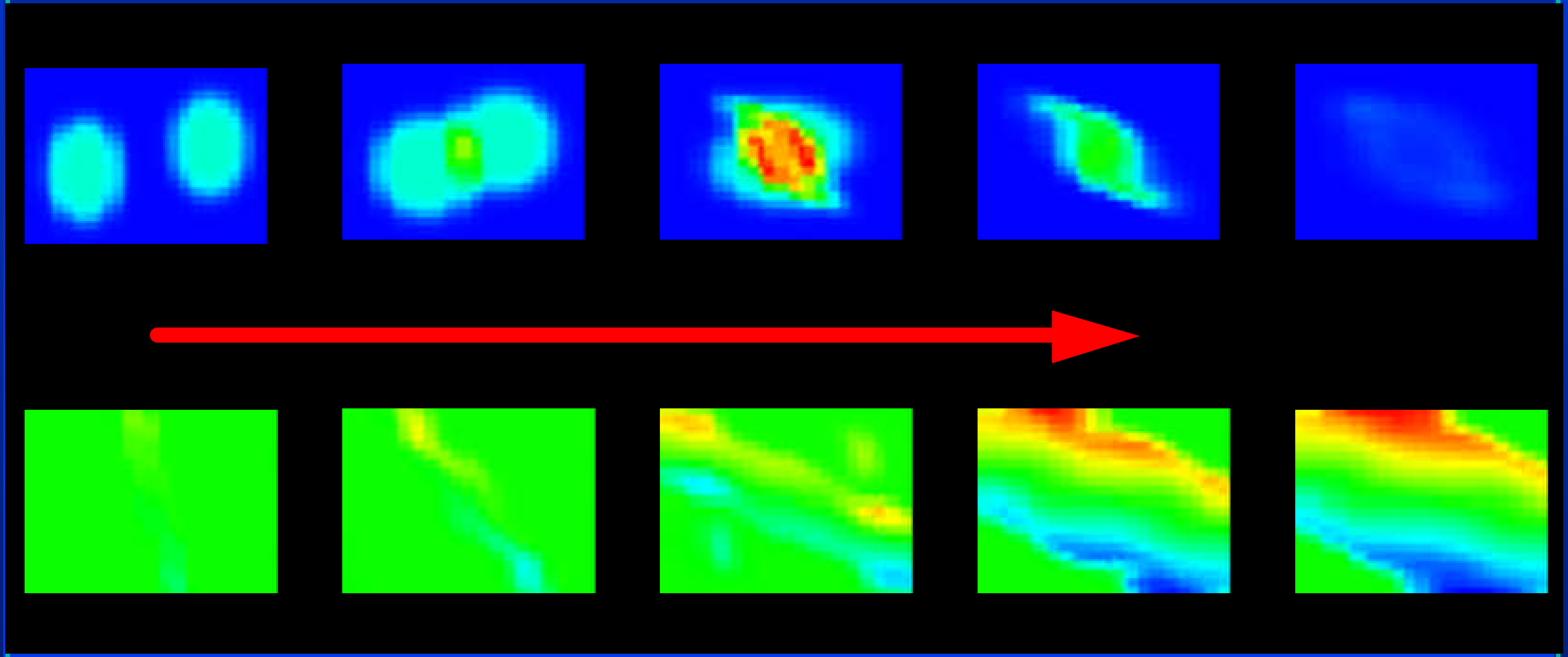
***Static multi-resolution (numerous)***

***Dynamic (fewer):***

- Volume rendering (Shen et al. Vis’99)
- Iso-surfaces (Sutton et al. Vis’99)
- Video (Finkelstein et al. Siggraph’96)
- Compression (Lengyel ACM I3DG’99)
- Industry standards (MPEG, VRML etc.)

# Dynamic Mesh Changes

## *Attributes change*



# Dynamic Mesh Changes

*Attributes change*

*Positions change*

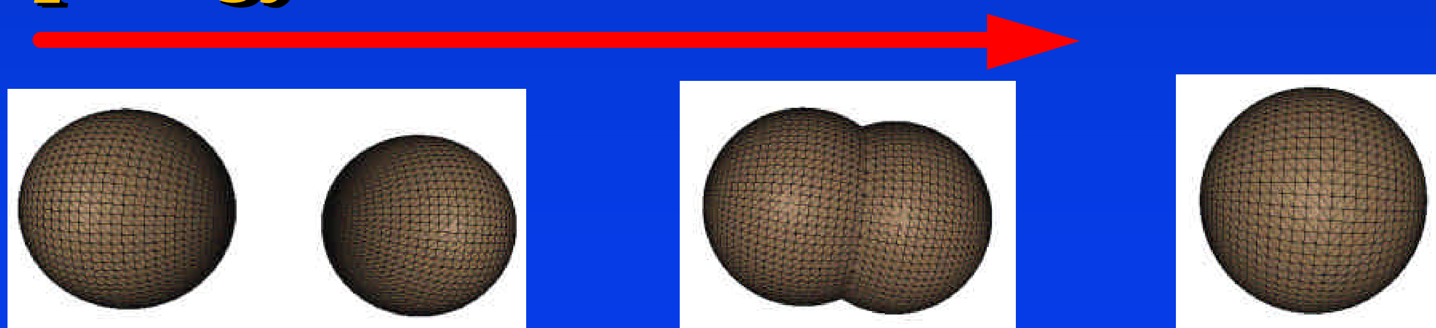
# Dynamic Mesh Changes

***Attributes change***

***Positions change***

***Connectivity***

***Topology***



# Dynamic Changes Encoding

*Attributes: Movies, simulations*

*Positions: VRML, MPEG4, Java3D, Animation systems*

*Connectivity, Topology: much more difficult!*

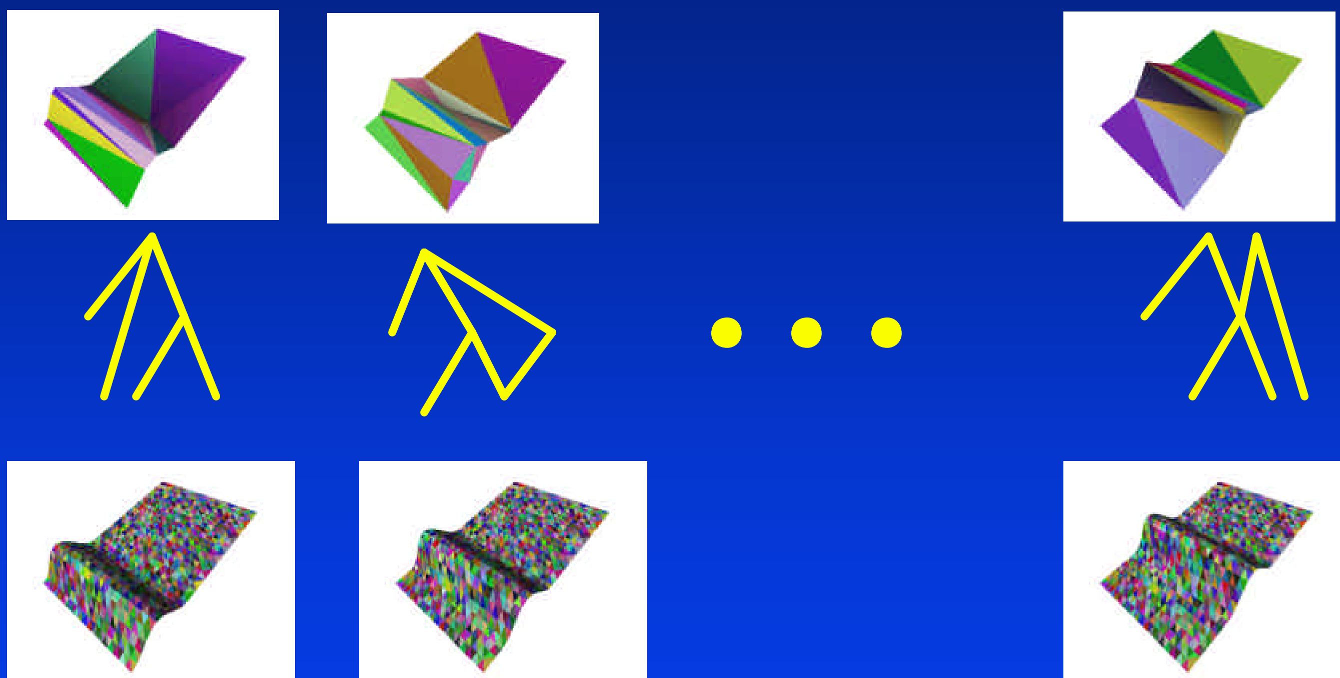
*…Multi-Resolution???*

# Key Observation 1

***Node positions and/or attributes must be updated over time! This suggests an O(n) process for each time-step***

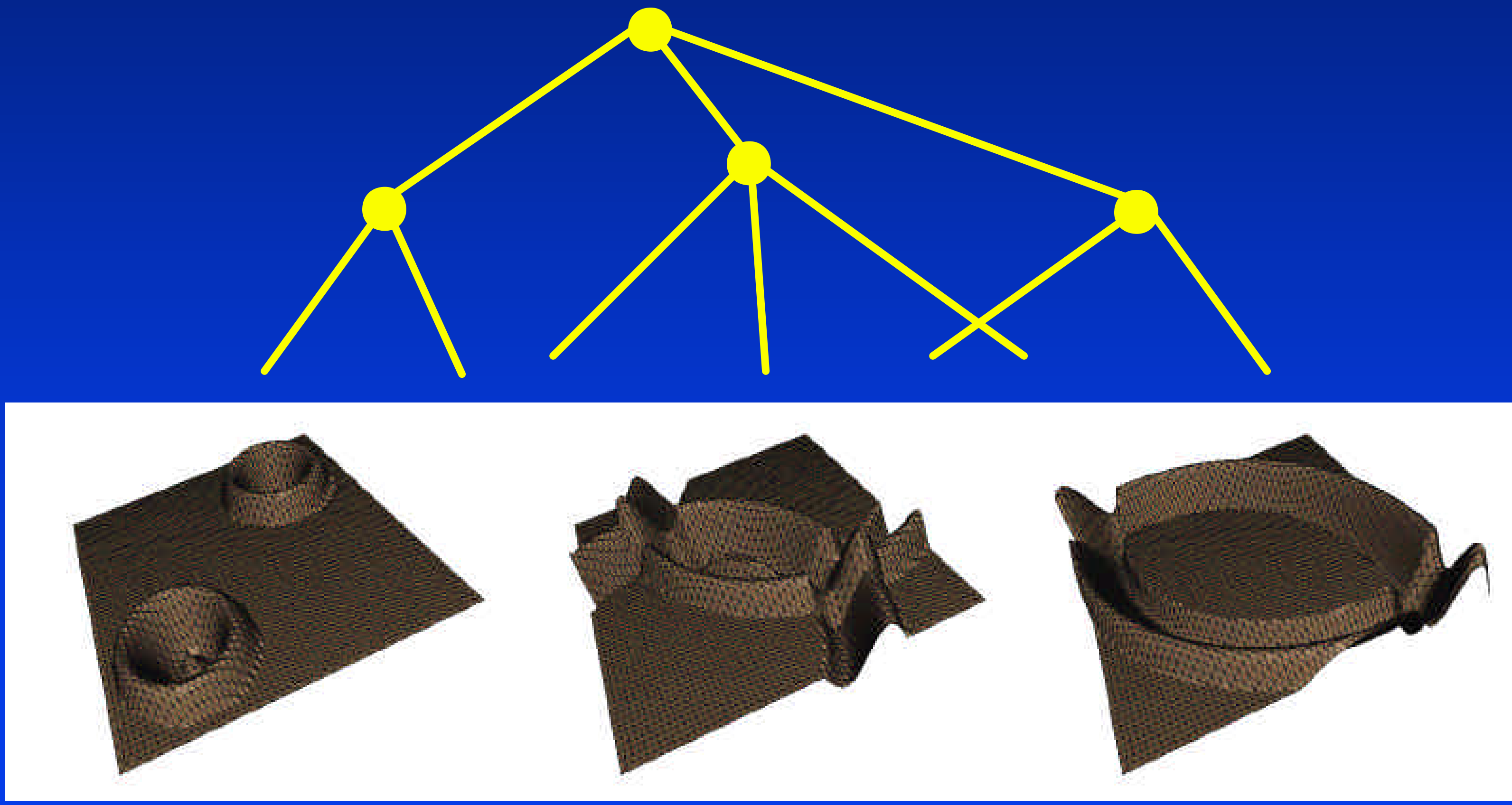
➔Multi-resolution utilization suggests O(number of nodes in LOD)

# Dynamic Multi-Resolution?

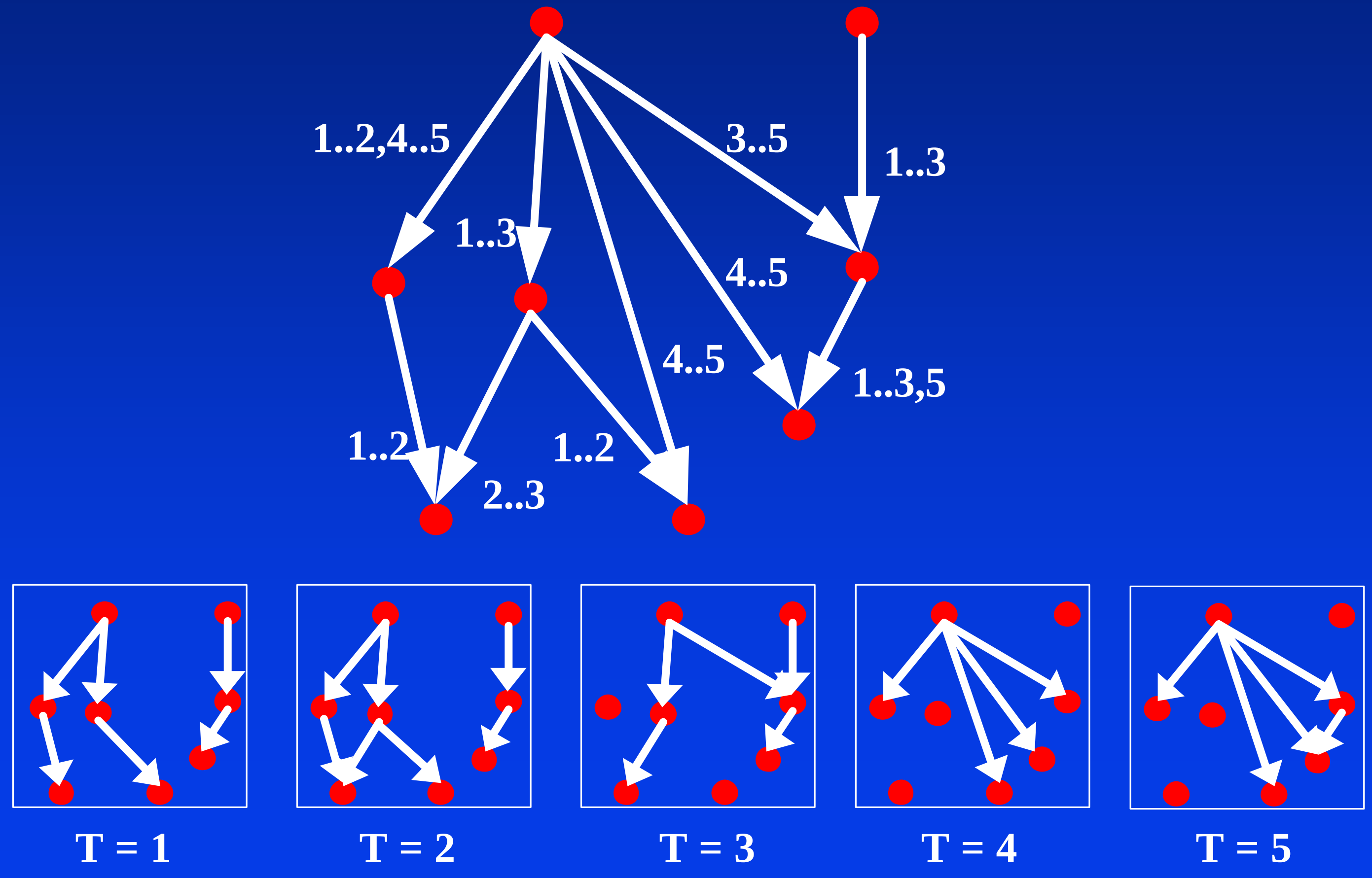


$T_0$ $T_1$ . . . $T_n$

# One Multi Resolution Structure: TDAG



# TDAG: Encoding Multiple DAGS



# TDAG: Merging Queues

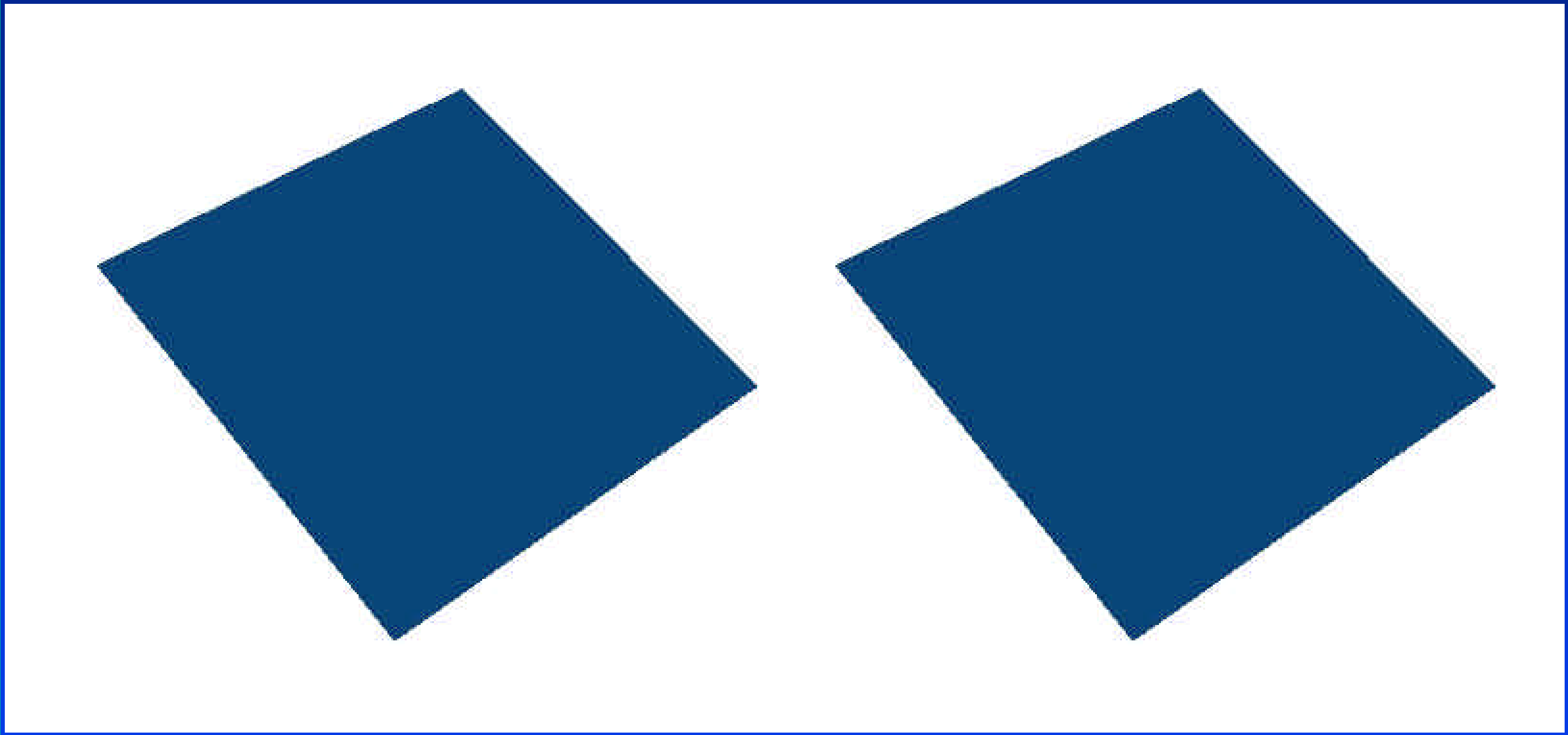
*The structure of each DAG is governed by the order of decimation operation queue.*

*Instead of merging DAGS, we "merge" the queues that govern the decimation order.*

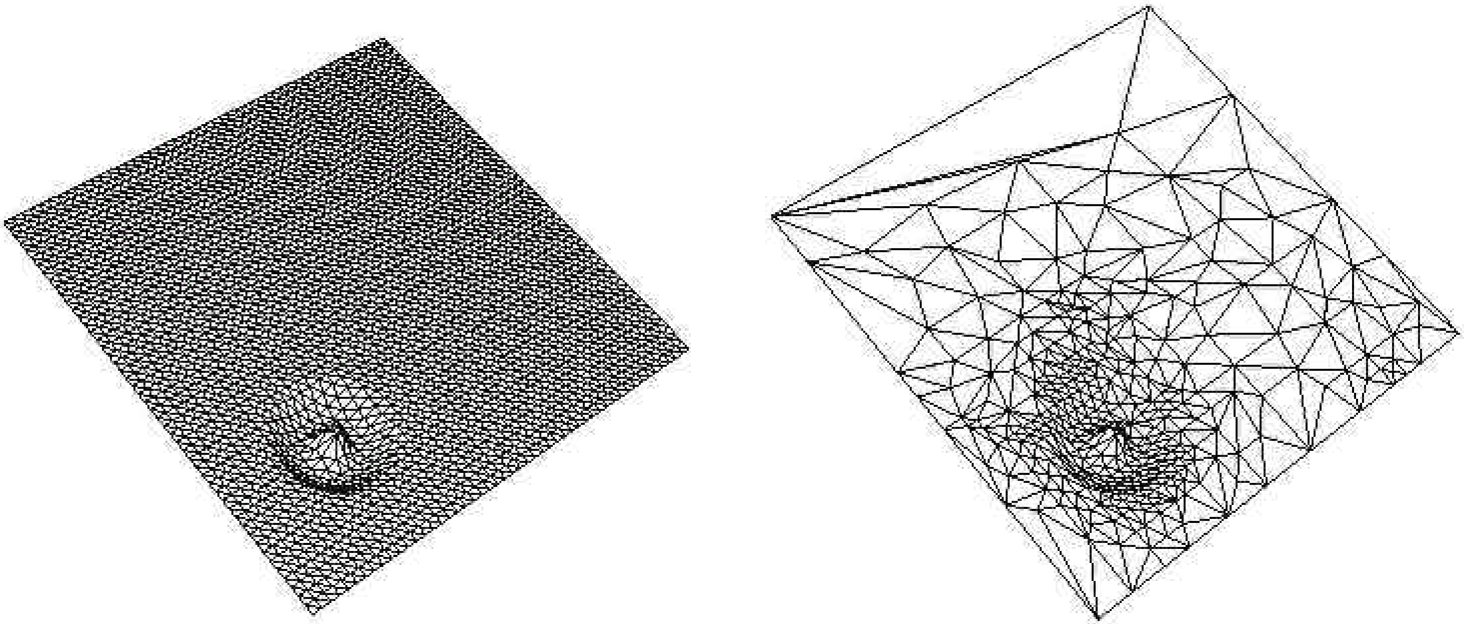
*More details in the paper…*

# Queries: 2D Cuts (Time x Resolution)

# Full Resolution Vs. Multi Resolution



# Full Vs. Multi Resolution (Meshes)

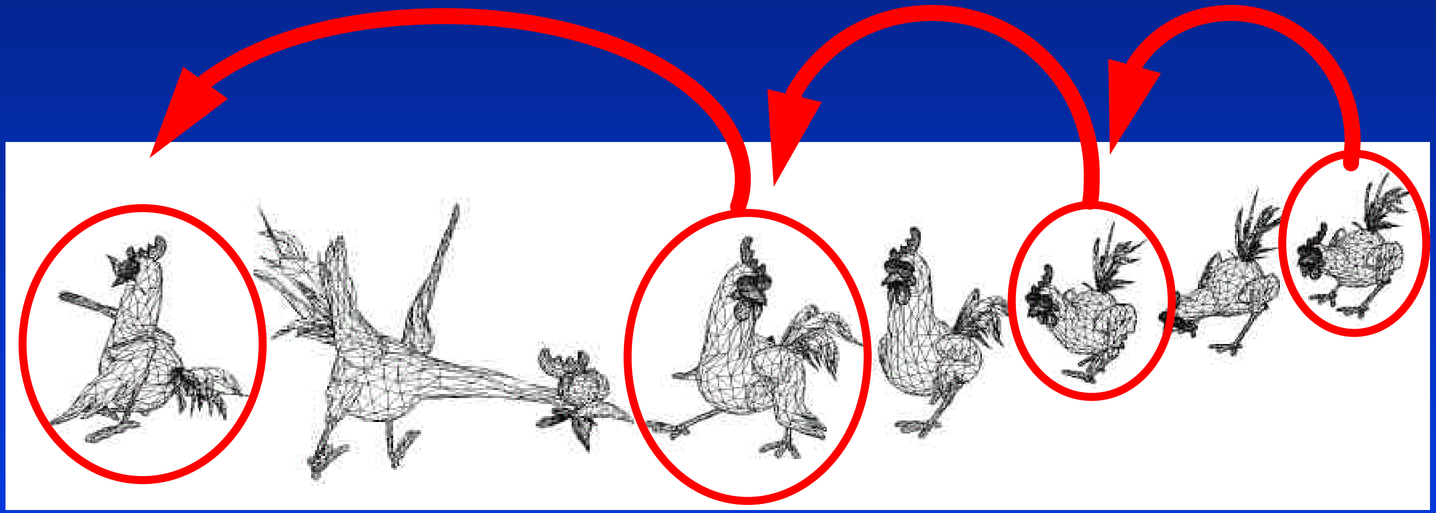


# Not Enough!

*The spatial dimension was used for creating approximations.*

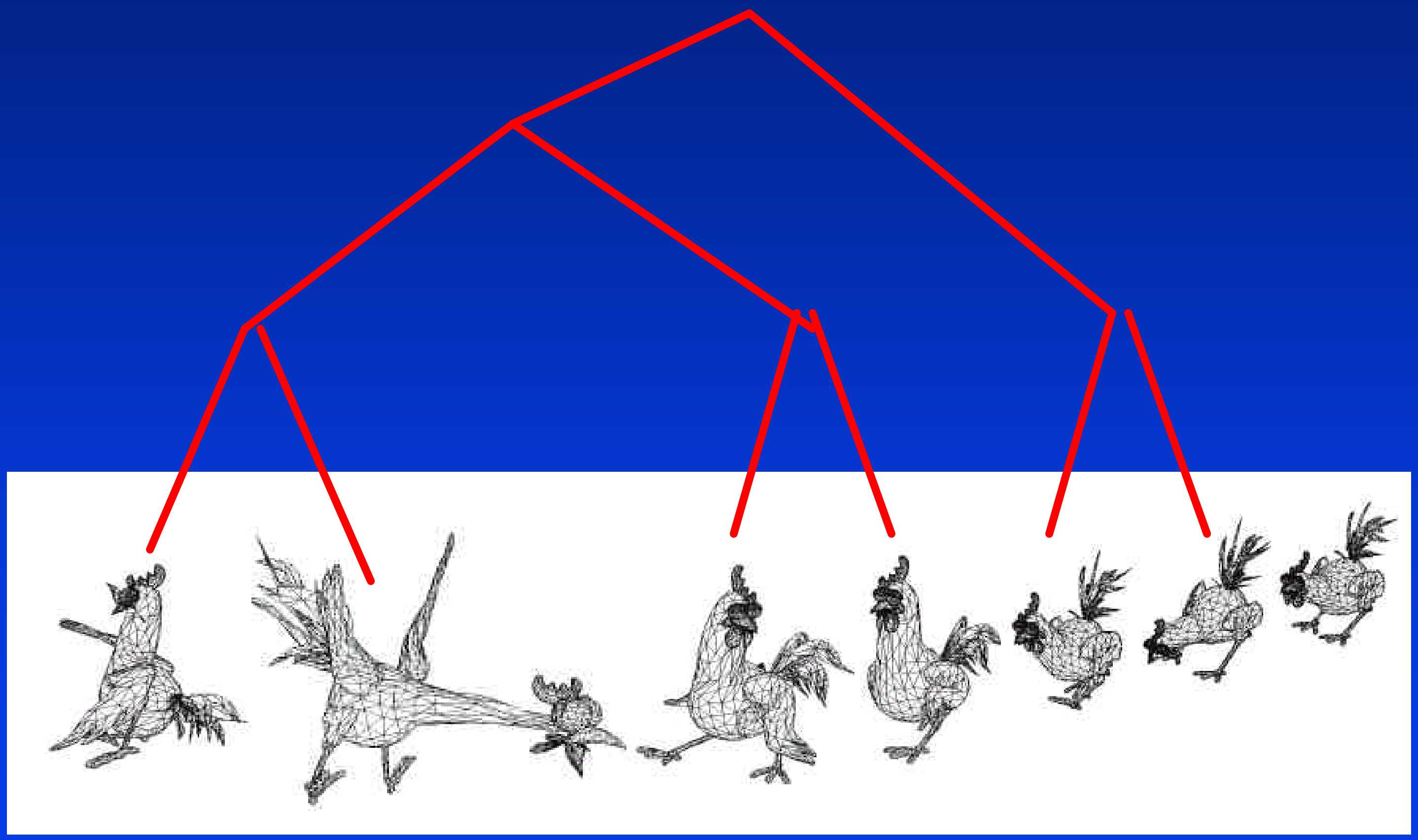
*We have dynamic meshes what about the temporal dimension?*

# Temporal Multi Resolution?

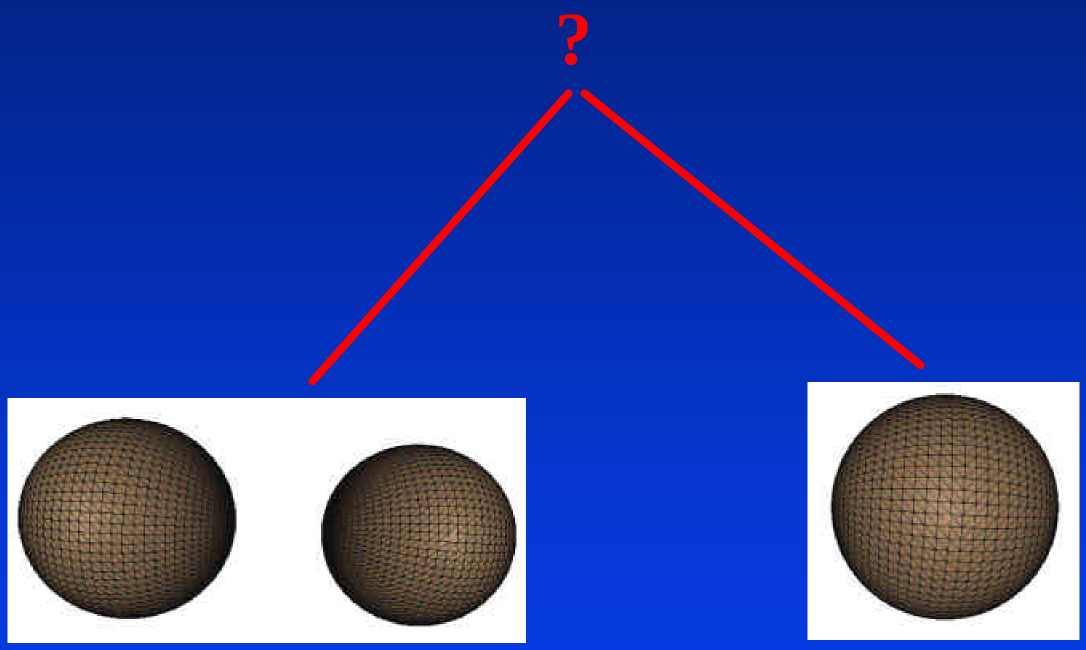


Skipping Frames

# Averaging



# Averaging

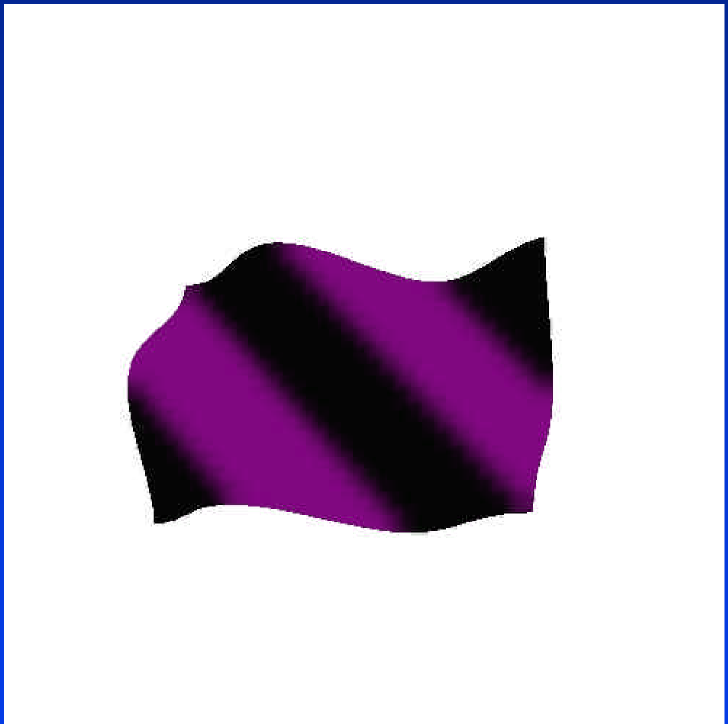


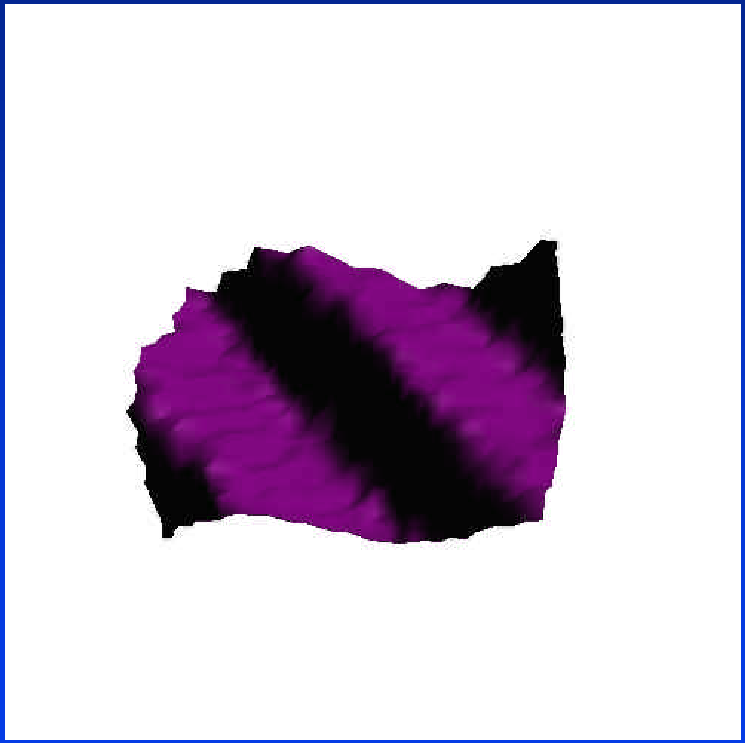
# Key Observation 2

***Some dynamic movements are perceptually more important than others!***

➔ Change over time can be separated into low frequency and high frequency for better encoding and utilized for faster update

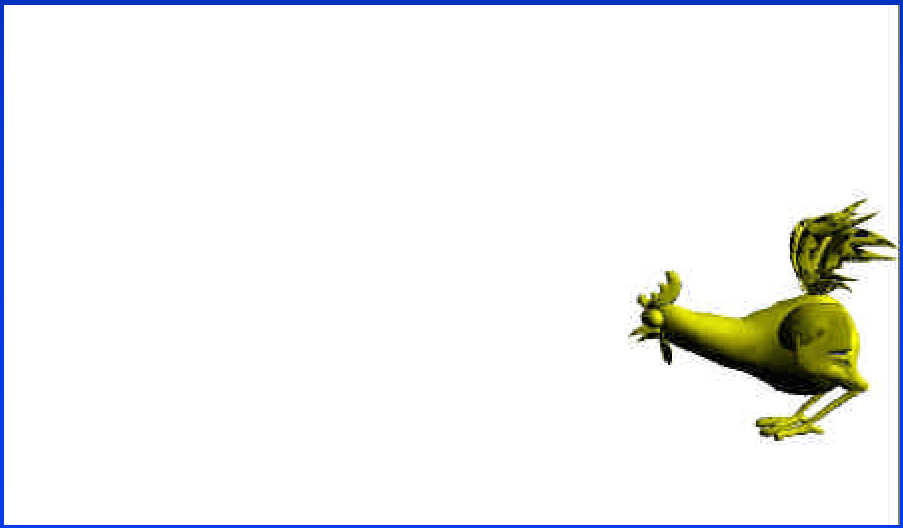
# Low & High Frequency Changes





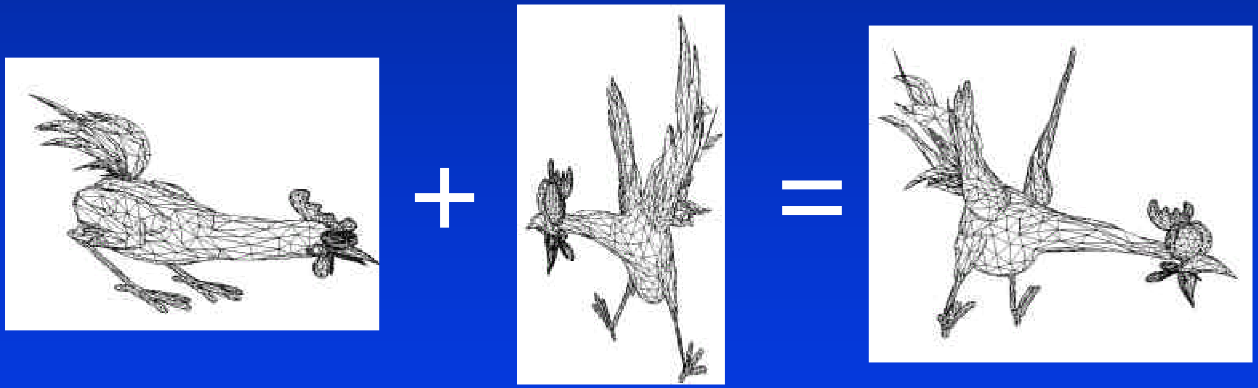
# Example: Chicken Sequence

***Examine both "low frequency" changes (walk, stretch, rotate) and "high frequency" (wiggle, flap etc.)***



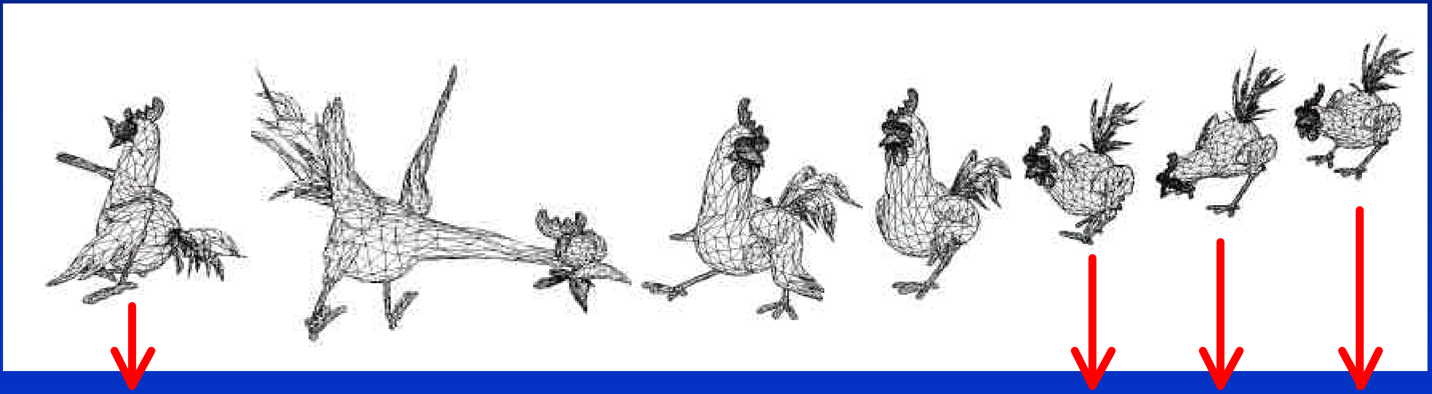
# Separating Frequencies

***Goal: take out*** $A_t$ ***- the Affine transformations from the sequence***



$$A_t * R_t = V_t$$

# Define Vertices Vector ( 4 x n Matrix)



| | | | | |
|---|---|---|---|---|
| $V_{k0}$ | | $V_{20}$ | $V_{10}$ | $V_{00}$ |
| $V_{k1}$ | | $V_{21}$ | $V_{11}$ | $V_{01}$ |
| $V_{k2}$ | . . . | $V_{22}$ | $V_{12}$ | $V_{02}$ |
| . | | . | . | . |
| . | | . | . | . |
| $V_{kn}$ | | $V_{2n}$ | $V_{1n}$ | $V_{0n}$ |

# Extracting Affine Maps

*Choose for example $V_0$, and for all t search $A_t$ (4x4 matrix) such that:*

$$A_t V_0 = V_t$$

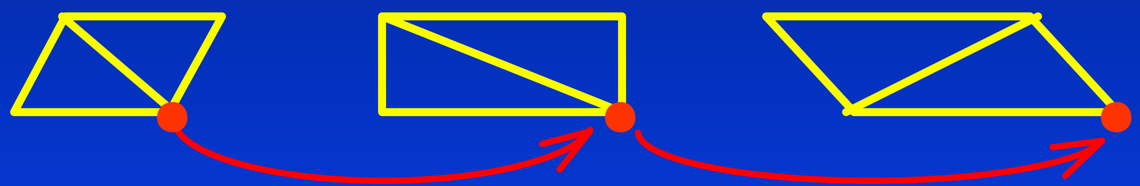
*Solve:*

$$A_t = V_t V_0^T (V_0 V_0^T)^{-1}$$

*Define Residual Mesh* $R_t = A_t^{-1} V_t$

# Topology & Connectivity Change

*Correspondence between nodes of meshes in different time steps.*



*Define* $\mathbf{V}_t$ *as a sample subset of common vertices*

# Why $V_0$ ?

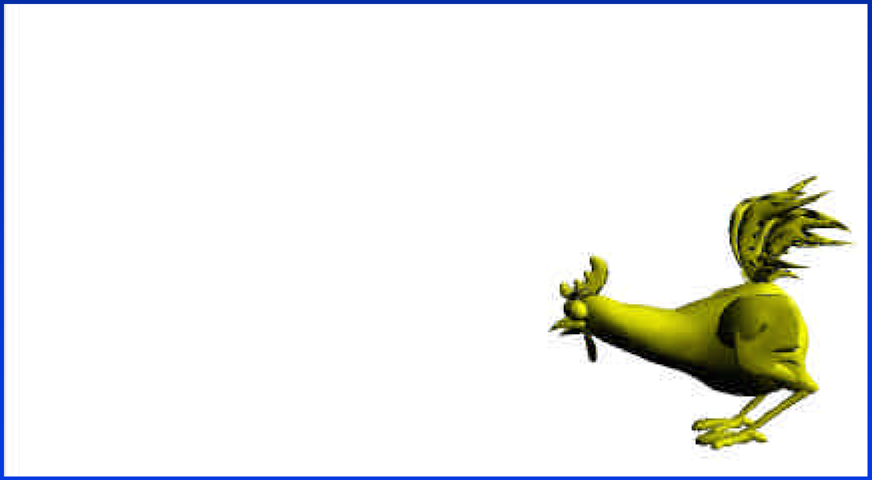
$$A_t V_0 = V_t$$

*Possibility for an on-line algorithm.*

*In long sequences – not much difference which mesh to choose as the base mesh.*

# Low Frequency Information

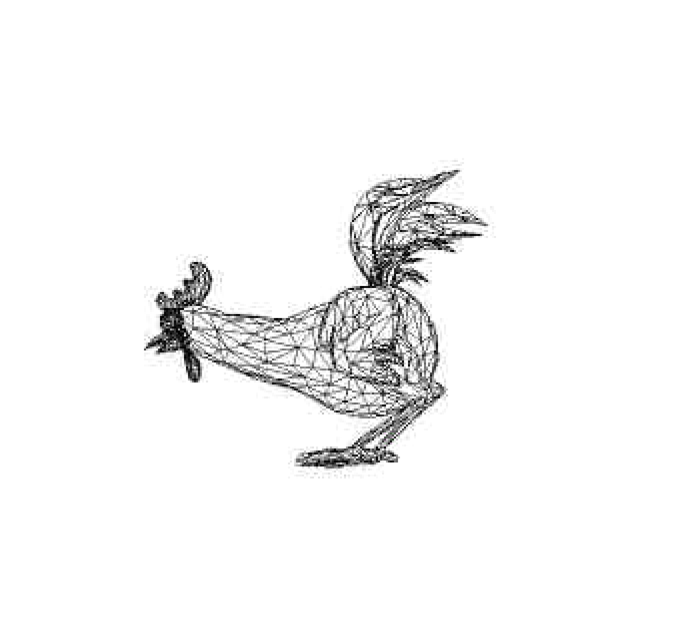
***Applying the affine matrices to the first mesh yields the following sequence:***



$A_t * V_0$
Sequence

***Note: very fast and not so bad as first temporal approximation!***

# High Frequency Information



*The residual meshes are encoded in a TDAG structure!*

$R_0, R_1, R_2, \ldots R_k$ → **TDAG**

# Recovering $M_{t+1}$ From $M_t$

1. *Update the multi-resolution cut in the TDAG according to the tolerance*
2. *Update attributes and position of the active vertices*
3. *Applying* $A_{t+1}$ *to position and scale the mesh and arrive at the approximation for* $M_{t+1}$

# The New Dynamic Mesh Representation

*Affine matrices + TDAG*

*All Attribute changes of vertices (including positions) are stored in the TDAG implicitly and used only if the vertex is active.*

# Model Utilization

***What to do when the time or space limitations for creating the next frame are too small for the full process?***

# Reversing the Order

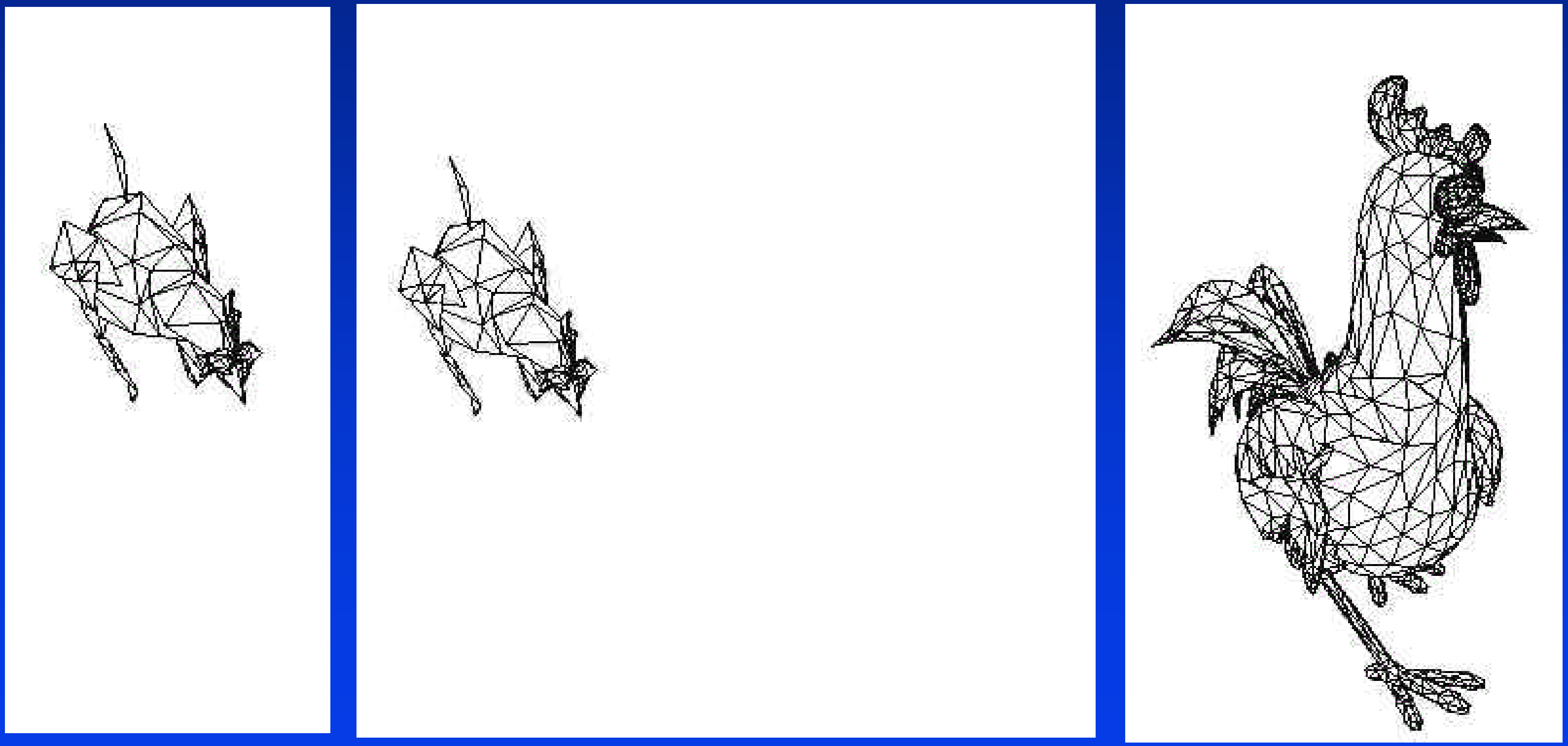
*First approximation: just apply* $A_{t+1}$ *to the current mesh.*

*Second approximation: update attributes and positions of current mesh without updating the cut.*

*Third approximation: update the mutli-resolution cut.*

# Low Resolution Version (~25% in size)

# Adaptive Change of Resolution



# Summary

*Utilizing both spatial multi-resolution and temporal frequency separation to create LOD for dynamic meshes.*

*By reversing the order of update operations we gain very fast approximations which are perceptually better.*

# Future Directions

- ***More complex encoding of movements than Affine trans.***
- ***Working with iso-surfaces - node correspondence problem!***
- ***Multi thread rendering (similar to double buffers).***
- ***Utilizing compression.***

# Call for Meshes ☺

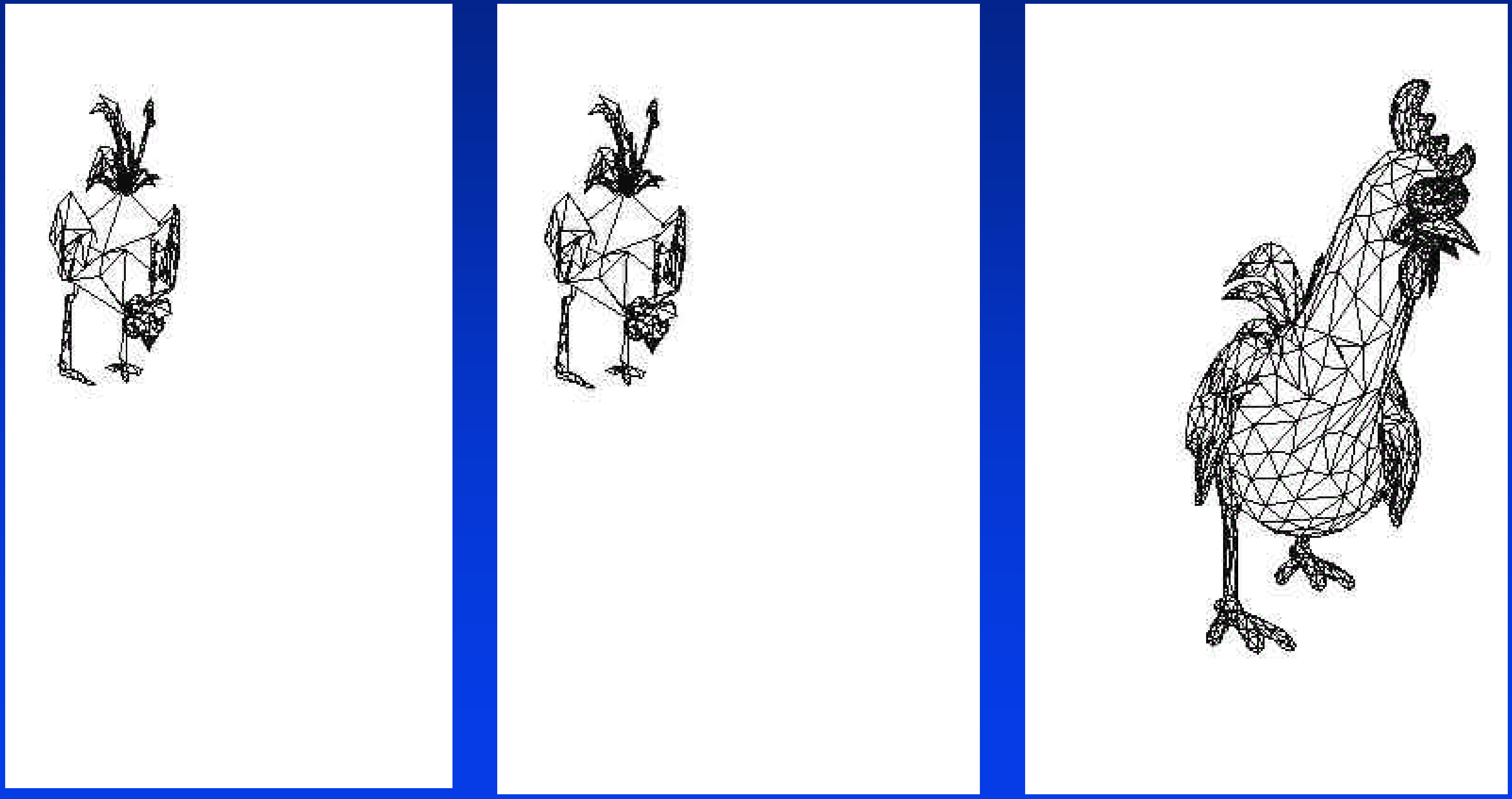
*We need dynamic mesh sequences!*

*Please contact –*

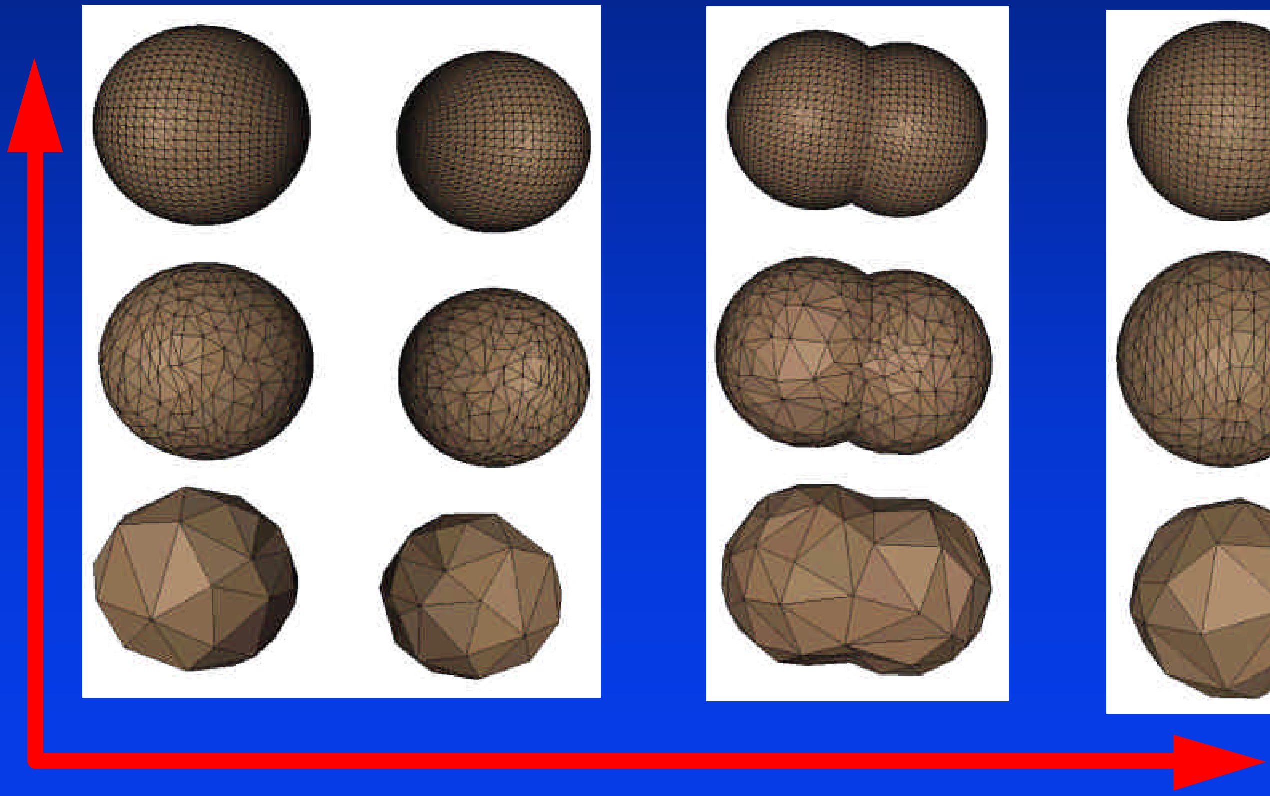
*arik@idc.ac.il*

*Thank you!*

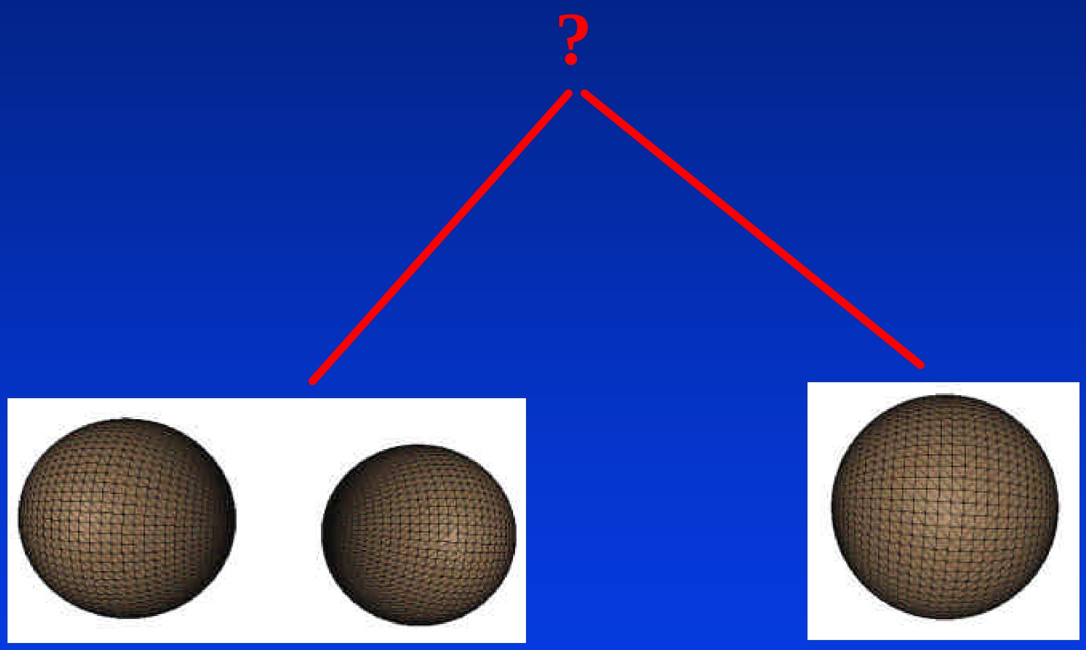
# Change of Resolution



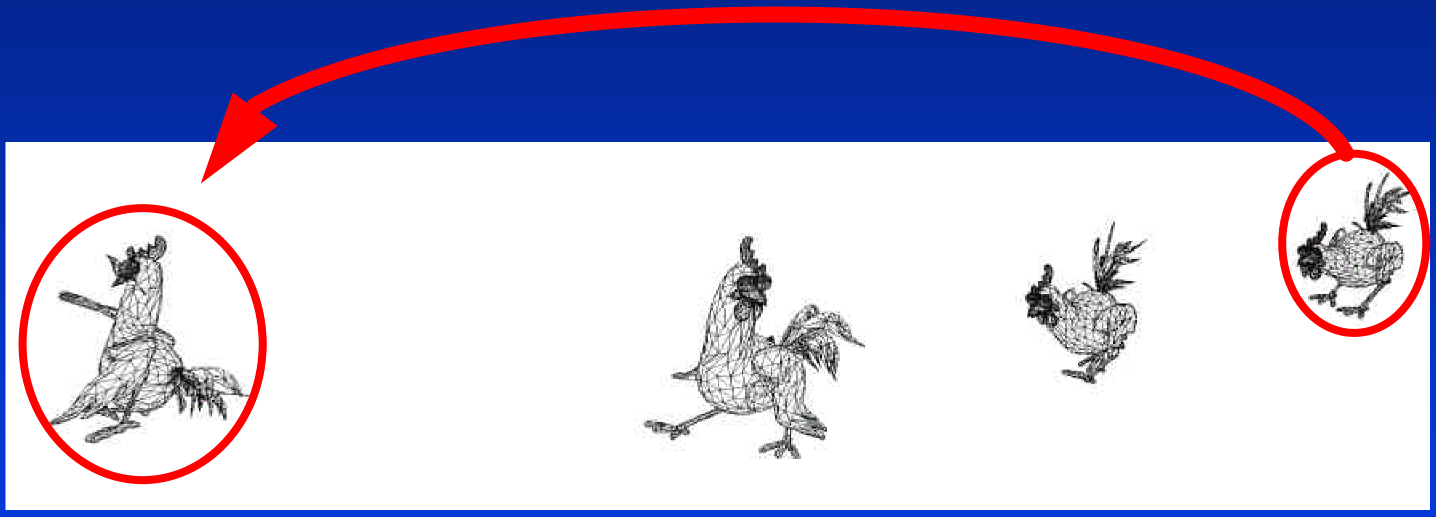
# Time x Resolution



# Averaging



# Frame Skipping



# Frame Skipping

# Time Tags

***All fields are augmented with time stamps***

- attributes
- position
- graph edges (decimation dependencies)

***Single "live" value: array***

***Multiple "live" values: interval tree, range tree etc.***

# Traversal

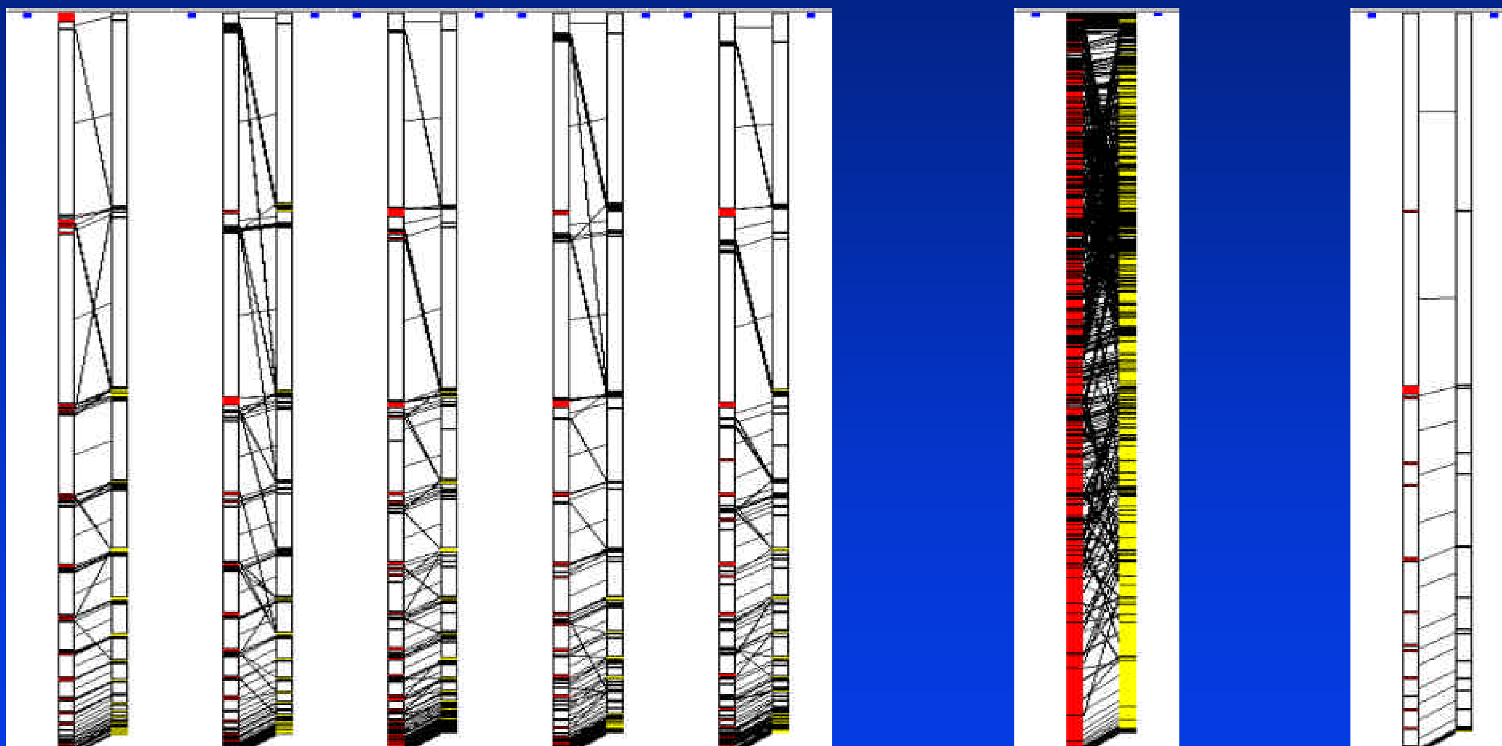
***A mesh is created by top-down traversal for each time step following "live" links and using "live" values.***

***This means all connectivity and topology changes are encoded seamlessly!***

# On-line Construction

- ***Bottom up decimation for time step t uses a priority queue and creates a DAG for time t.***
- ***Note that the DAG for time t and t+1 are the same iff the two decimation orders are the same.***
- ***Instead of merging two DAGs, the decimation of time t+1 uses an enhanced priority taking into account the order in the previous time t, creating similar DAGs for all time-steps.***

# The queues



1-2 2-3 3-4 4-5 5-6 Independent Fully Conform

# Dynamic Multi Resolution Extraction