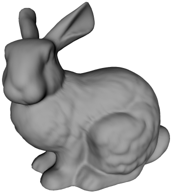
A Flexible C++ Library for Wavelet Transforms of 3-D Polygon Meshes

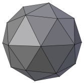
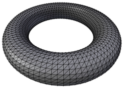
Shengyang Wei

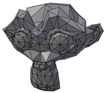
Feburary 4, 2020



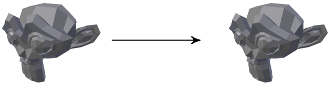
Abstract

- **Polygon meshes**: modelling 3-D objects by joined planar polygons.
    - **Triangle meshes**: all polygons being triangles.

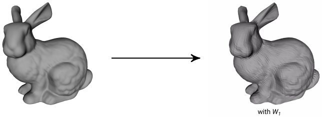


- **Subdivision**: characterizes a smooth surface by a simple mesh.



subdivision

- **Multiresolution analysis and wavelet transforms**: represent a complicated mesh in multiple levels of detail (resolutions).



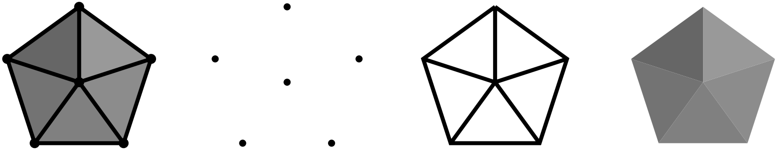
wavelet transform

Table of Contents

Polygon Meshes

Polygon Mesh

- Primitives: vertices, edges, and faces.



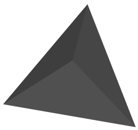
(a) Mesh (b) Vertices (c) Edges (d) Faces

- **Geometry**: positions of vertices.
- **Topology**: adjacency relationships between vertices, edges, and faces.
- **Boundary edge**:has exactly one incident face.
- **Interior edge**: has exactly two incident faces.
- **Boundary vertex**: has exactly two incident boundary edges.
- **Interior vertex**: has incident interior edges only.
- **Closed mesh**: has no boundary edges.

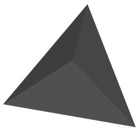
Subdivision

Subdivision

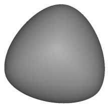
- Subdivision algorithmically inserts vertices, edges, and faces to a simple control mesh to yield a refined one.
- A refined mesh can be obtained by several round of subdivision.



(a) Control mesh.



(b) Repetitive subdivision.

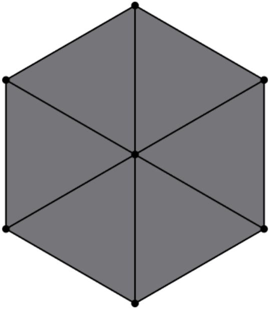


(c) Refined mesh.

- Subdivision is defined by two rules: 1) **topologic refinement rules** and 2) **geometric refinement rules**.

Primal Triangle Quadrisection (PTQ)

- A topologic refinement rule defined on triangle meshes.
- Each triangle is split into four to insert new vertices and edges:
  1. insert a vertex on each edge.
  2. connect the new vertices by edges to split each triangle.

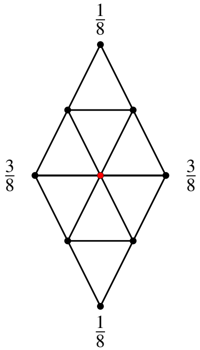


- The obtained mesh is said to have **subdivision connectivity** (PTQ connectivity).

Geometric Refinement Rules

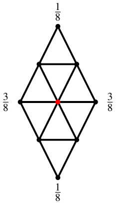
- A geometric refinement rule modifies the positions of new vertices (and probably old vertices).
- The vertices whose positions will be updated are called **target vertices**.
- The vertices that participate in the computation are called **support vertices**.

- A geometric refinement rule is defined by a mask.
- A mask specifies the target vertex and support vertices and their weights.
- The position of a target vertex is updated by weighted sum of vertices on a mask.

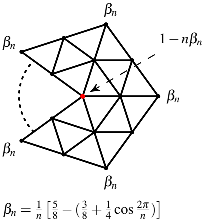


Loop Subdivision

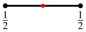
1. Apply PTQ.
2. Apply Mask I to each new interior vertex v_e:
$$v_e = \frac{3}{8}(v_1+v_2) + \frac{1}{8}(v_3+v_4).$$
3. Apply Mask II to each old interior vertex v:
$$v' = (1-n\beta_n)v + \beta_n \sum_{i=1}^{n} v_i.$$
4. Apply Mask III to each new boundary vertex v_e:
$$v_e = \frac{1}{2}(v_1+v_2).$$
5. Apply Mask IV to each old boundary vertex v:
$$v' = \frac{3}{4}v + \frac{1}{2}(v_1+v_2).$$



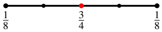
Mask I



Mask II



Mask III

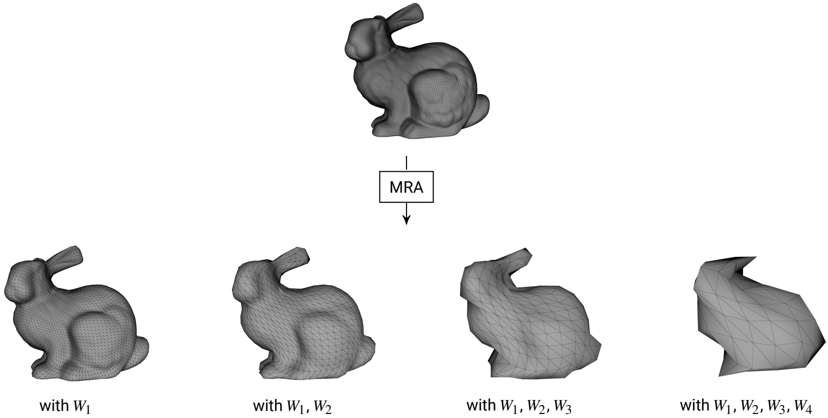


Mask IV

Multiresolution Analysis and Wavelet Transforms

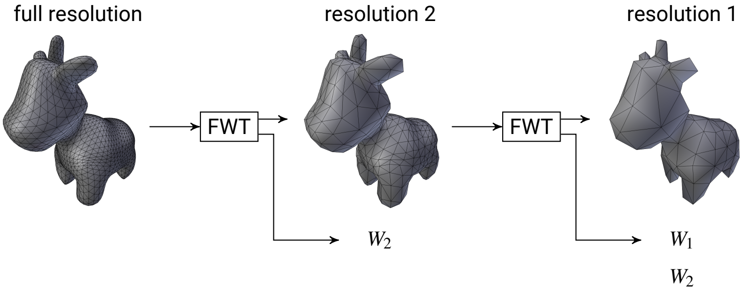
Multiresolution Analysis (MRA)

- MRA represents a complicated mesh in a multiresolution form: a coarse approximation in the lowest resolution and sets of wavelet coefficient that encode information in each higher resolution.



Wavelet Transform I

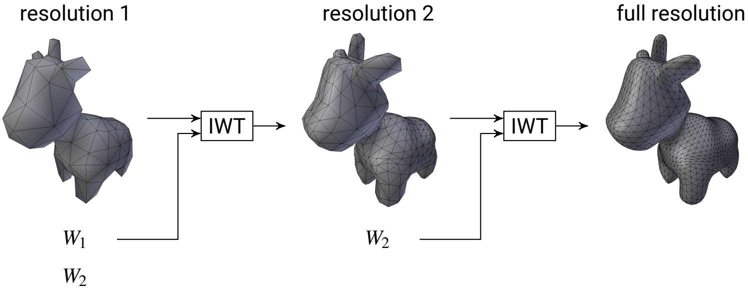
- A multiresolution analysis is associated with a wavelet transform.
- A level of **forward wavelet transform** (FWT) yields a coarse mesh in the next lower resolution and a set of wavelet coefficients.



- A FWT requires subdivision connectivity [1], which guarantees the existence of a multiresolution analysis.

Wavelet Transform II

- A level of **inverse wavelet transform** (IWT) incorporate a set of wavelet coefficients into a coarse mesh to recover the mesh in the next higher resolution.



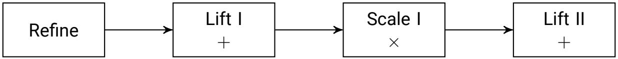
Lifting Scheme and Lifted Wavelet Transforms

Lifting Scheme

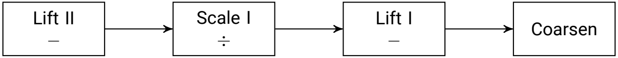
- a framework, proposed by Sweldens [3], for designing, analyzing, and implementing a WT.
- can yield the inverse transform trivially.
- can compute the WT in linear time.
- computation steps:
    1. partition data into disjoint sets.
    2. lifting step: add (subtract) a filtered version of other sets to (from) a set.
    3. scaling step: multiply (divide) a set by a scalar.

Lifted Wavelet Transforms

- IWT (one level):
  1. Refine the mesh by a topologic refinement rule (naturally classify vertices as old and new vertex sets).
  2. Initialize the positions of new vertices with wavelet coefficients.
  3. Perform cascaded lifting steps and scaling steps.
- FWT (one level):
  1. Partition vertices based on subdivision connectivity. For PTQ connectivity, vertices are classified into old and new vertex sets.
  2. Perform reversed lifting steps and scaling steps.
  3. Coarsen mesh (new vertices become wavelet coefficients).



(a) One level IWT.

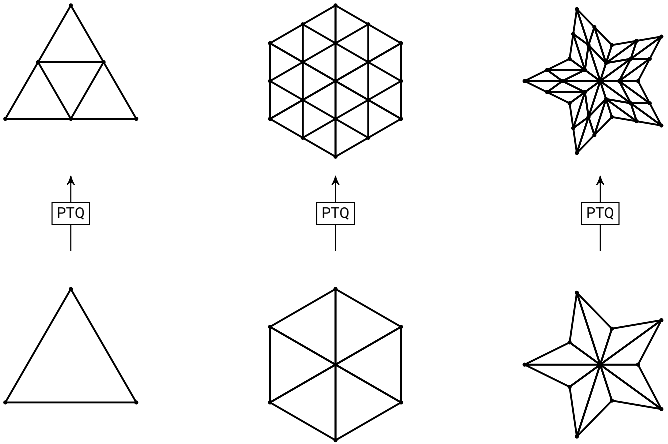


(b) One level FWT.

Vertex Parition and Coarsening I

Topologic examples

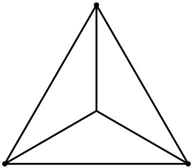
- Partitioning depends on subdivision connectivity.
- Examples with PTQ connectivity:

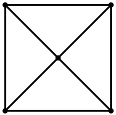


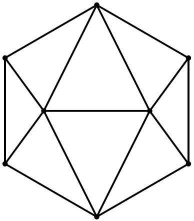
Vertex Parition and Coarsening II

Topologic examples

- Examples without PTQ connectivity:



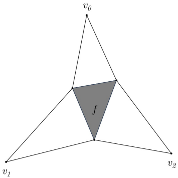


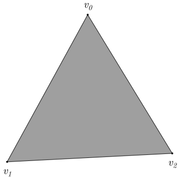


PTQ Connectivity Detection

- Taubin's **covering mesh** method [4] (consider topology only).

1. Construct tiles by reversing PTQ on each triangle.

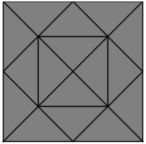




2. Group tiles by connectivity.
3. Check if a group of tiles can yield the topology of the original mesh.

PTQ Connectivity Detection

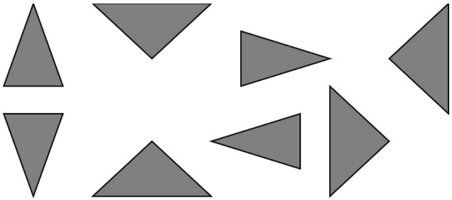
Construct tiles



Tiles:

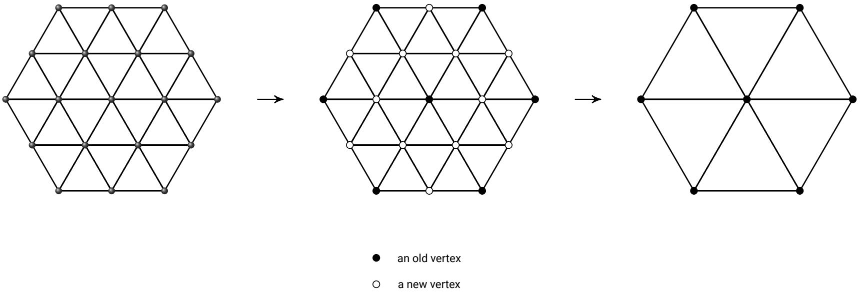
PTQ Connectivity Detection

Filter tiles



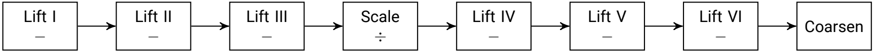
Coarsening

- Identify the vertices introduced by PTQ (new vertices).
- Remove the new vertices and the edges that connect them.

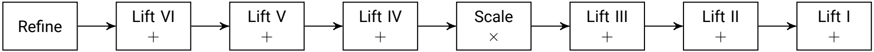


Lifted Loop WT

- defined on triangle meshes with or without boundaries.
- proposed by Bertram [5] and Wang et al. [6].
- consists of six lifting steps and one scaling step.



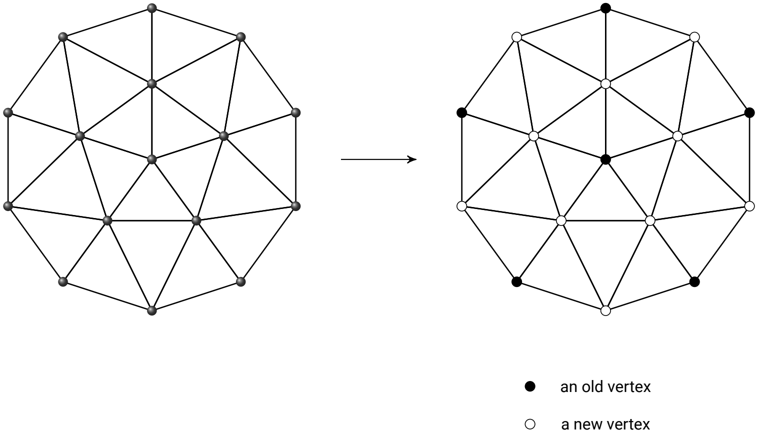
Operations in the FWT



Operations in the IWT

- partition vertices in two groups: new vertices and old vertices.

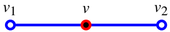
1. Detect PTQ connectivity and classify vertices as new vertices and old vertices.

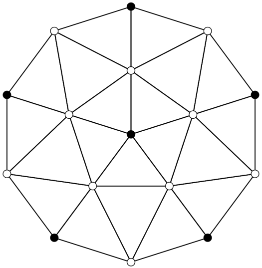


Lifted Loop FWT

2. Lift an old boundary vertex v by new boundary vertices v_1 and v_2:

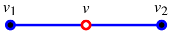
$$v' = v - \frac{1}{4}(v_1 + v_2)$$

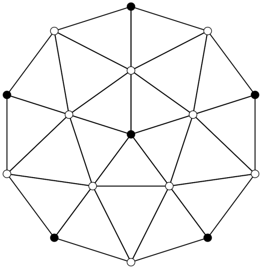




Lifted Loop FWT

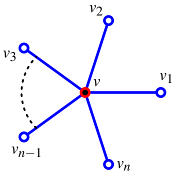
2. Lift an old boundary vertex v by new boundary vertices v_1 and v_2:
$$v' = v - \frac{1}{4}(v_1 + v_2)$$
3. Lift a new boundary vertex v by old boundary vertices v_1 and v_2:
$$v' = v - \frac{1}{2}(v_1 + v_2)$$

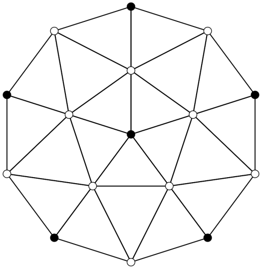




Lifted Loop FWT

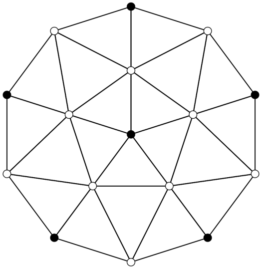
2. Lift an old boundary vertex v by new boundary vertices v_1 and v_2:
$$v' = v - \frac{1}{4}(v_1 + v_2)$$
3. Lift a new boundary vertex v by old boundary vertices v_1 and v_2:
$$v' = v - \frac{1}{2}(v_1 + v_2)$$
4. Lift an old interior vertex v by n new interior vertices $\{v_i\}$:
$$v' = v - \delta_n \sum_{i=1}^{n} v_i$$





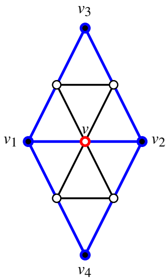
Lifted Loop FWT

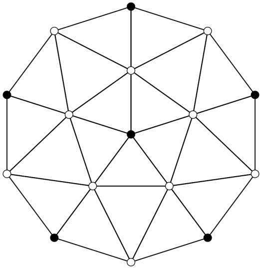
2. Lift an old boundary vertex v by new boundary vertices v_1 and v_2:
$$v' = v - \frac{1}{4}(v_1 + v_2)$$
3. Lift a new boundary vertex v by old boundary vertices v_1 and v_2:
$$v' = v - \frac{1}{2}(v_1 + v_2)$$
4. Lift an old interior vertex v by n new interior vertices $\{v_i\}$:
$$v' = v - \delta_n \sum_{i=1}^{n} v_i$$
5. Scale an old interior vertex v with a valence n by a scalar:
$$v' = \frac{v}{\beta_n}$$



Lifted Loop FWT

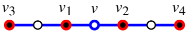
2. Lift an old boundary vertex v by new boundary vertices v_1 and v_2:
$$v' = v - \frac{1}{4}(v_1 + v_2)$$
3. Lift a new boundary vertex v by old boundary vertices v_1 and v_2:
$$v' = v - \frac{1}{2}(v_1 + v_2)$$
4. Lift an old interior vertex v by n new interior vertices $\{v_i\}$:
$$v' = v - \delta_n \sum_{i=1}^{n} v_i$$
5. Scale an old interior vertex v with a valence n by a scalar:
$$v' = \frac{v}{\beta_n}$$
6. Lift a new interior vertex v by old vertices v_1, v_2, v_3, and v_4:
$$v' = v - \left[\frac{3}{8}(v_1 + v_2) + \frac{1}{8}(v_3 + v_4)\right]$$

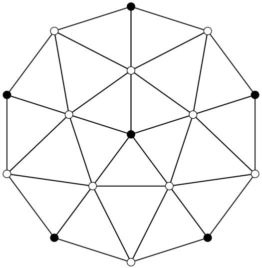




Lifted Loop FWT

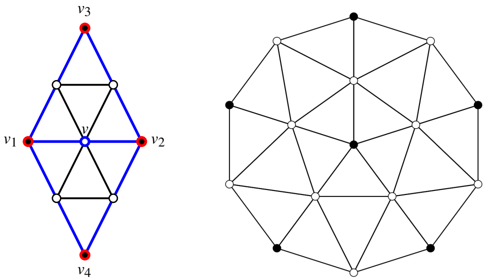
2. Lift an old boundary vertex v by new boundary vertices v_1 and v_2:
$$v' = v - \frac{1}{4}(v_1 + v_2)$$
3. Lift a new boundary vertex v by old boundary vertices v_1 and v_2:
$$v' = v - \frac{1}{2}(v_1 + v_2)$$
4. Lift an old interior vertex v by n new interior vertices $\{v_i\}$:
$$v' = v - \delta_n \sum_{i=1}^{n} v_i$$
5. Scale an old interior vertex v with a valence n by a scalar:
$$v' = \frac{v}{\beta_n}$$
6. Lift a new interior vertex v by old vertices v_1, v_2, v_3, and v_4:
$$v' = v - \left[\frac{3}{8}(v_1 + v_2) + \frac{1}{8}(v_3 + v_4)\right]$$
7. Lift old boundary vertices v_1, v_2, v_3, and v_4 by a new boundary vertex v:
$$v_i' = v_i - \eta_i v \;\; \forall i = 1,2,3,4$$



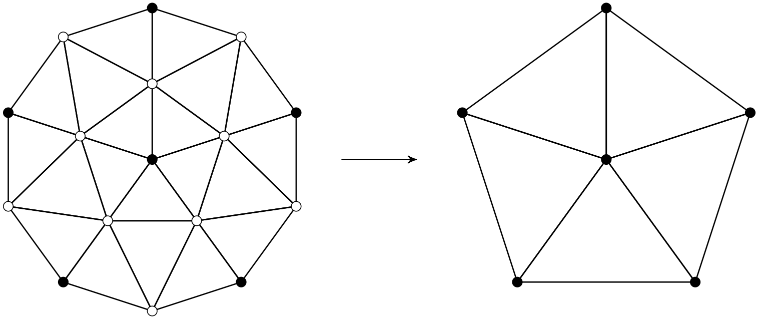


Lifted Loop FWT

2. Lift an old boundary vertex v by new boundary vertices v_1 and v_2:
$$v' = v - \frac{1}{4}(v_1 + v_2)$$
3. Lift a new boundary vertex v by old boundary vertices v_1 and v_2:
$$v' = v - \frac{1}{2}(v_1 + v_2)$$
4. Lift an old interior vertex v by n new interior vertices $\{v_i\}$:
$$v' = v - \delta_n \sum_{i=1}^{n} v_i$$
5. Scale an old interior vertex v with a valence n by a scalar:
$$v' = \frac{v}{\beta_n}$$
6. Lift a new interior vertex v by old vertices v_1, v_2, v_3, and v_4:
$$v' = v - \left[\frac{3}{8}(v_1 + v_2) + \frac{1}{8}(v_3 + v_4)\right]$$
7. Lift old boundary vertices v_1, v_2, v_3, and v_4 by a new boundary vertex v:
$$v_i' = v_i - \eta_i v \ \ \forall i = 1, 2, 3, 4$$
8. Lift old vertices v_1, v_2, v_3, and v_4 by a new interior vertex v:
$$v_i' = v_i - \omega_i v \ \ \forall i = 1, 2, 3, 4$$

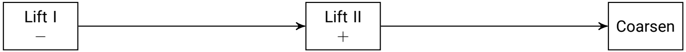


9. Remove the new vertices from the mesh.

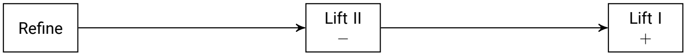


Lifted Butterfly WT

- defined on closed triangle meshes.
- proposed by Sweldens [2].
- consists of two lifting steps.



Operations in the FWT.

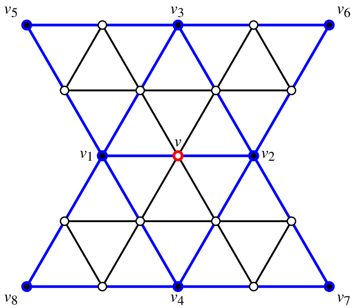


Operations in the IWT.

- partition vertices in two groups: new vertices and old vertices.

1. Lift a new vertex v by old vertices $\{v_i\}_{i=1}^{8}$:

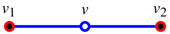
$$v' = v - \left[\frac{1}{2}(v_1 + v_2) + \frac{1}{8}(v_3 + v_4) - \frac{1}{16}(v_5 + v_6 + v_7 + v_8)\right]$$



2. Lift old vertices $\{v_i\}_{i=1}^2$ by a new vertex v:

$$v_i' = v_i + s_i v \quad \forall i = 1,2$$
$$s_i = \frac{4^{L-j} - 1}{2\left[1 + \frac{n}{6}\left(4^{L-j-1} - 1\right)\right]}$$

L is the number of levels, and $j \in \{1,2,\dots,L\}$ is the current level.



Wavelet Transform Toolkit

Wavelet Transform Toolkit

- Wavelet Transform Toolkit:
  - `https://github.com/uvic-aurora/wtt.git`
  - a C++ header-only library for defining and computing lifted wavelet transforms.
  - wavelet-based application programs.
- Library:
  - an application programming interface (API) for defining custom wavelet transforms.
  - built-in functions that implement Loop and Butterfly wavelet transforms, detecting PTQ connectivity, PTQ-based coarsening and refinement, etc.
- Application programs:
  - FWT and IWT computations.
  - wavelet-based compression, approximation, and denoising.

Wavelet Transform Toolkit

- can compute FWT in $O(n \log n)$ time and IWT in $O(n)$ time on a mesh with n vertices.
- can compute FWT and IWT with $O(n)$ memory on a mesh with n vertices.
- Execution time and memory cost(collected on a computer with Core i7-8700k CPU and 32GB RAM):

Name	Vertices	Time (ms)				Memory (MB)	
		Butterfly		Loop		FWT	IWT
		FWT	IWT	FWT	IWT		
`torus`	12288	13.40	8.00	14.40	8.00	13.8	8.3
`bunny`	24578	31.20	14.90	34.00	14.70	27.6	16.6
`bulb`	28162	52.40	23.60	58.00	30.00	31.6	18.9
`dragon`	32000	64.00	30.60	77.10	33.20	35.9	21.4
`torusknot`	40960	90.00	35.80	103.00	45.10	45.9	27.1
`kid`	49154	98.20	45.80	108.20	48.50	55.1	33.2
`horse`	65538	154.41	69.00	161.61	76.70	73.4	43.7
`gargoyle`	65538	156.71	70.00	167.51	74.80	73.4	43.7
`tyra`	98306	241.41	113.00	254.31	120.81	110.1	66.3
`vase`	196610	503.52	242.21	521.02	254.51	220.2	132.5
`armardillo`	262146	701.13	344.22	746.93	363.02	293.6	174.6
`cow`	393218	1080.00	509.12	1130.00	534.72	440.4	265.1
`venus`	1048578	3120.00	1420.00	3250.00	1500.0	1174.4	698.4

Demonstration

Q & A

References

lounsbery, M.
Multiresolution Analysis for Surfaces of Arbitrary Topological Type
University of Washington, WA, USA, 1994

Schröder, P. and Sweldens, W.
Spherical Wavelets: Texture Processing
Rendering Techniques '95, 1995

Sweldens, W.
The Lifting Scheme: A Custom-Design Construction of Biorthogonal Wavelets
Applied and Computational Harmonic Analysis, 1996

Taubin, G.
Detecting and Reconstructing Subdivision Connectivity
The Visual Computer, 2001

Bertram, M.
Biorthogonal Loop-subdivision Wavelets
Computing, 2004

Wang, H. and Tang, K.
Biorthogonal Wavelet Construction for Hybrid Quad/Triangle Meshes
The Visual Computer, 2009