Unsupervised Conversion of 3D Models for Interactive Metaverses

Jeff Terrace, Ewen Cheslack-Postava,
Philip Levis and Michael J. Freedman

What is a Virtual World?

- Three-dimensional, online environment
- Users can communicate, shop, socialize, collaborate, and learn.

Virtual World Types

Static

- Fixed art
- Artist-generated environment
- Predictable
- Restricted user ability

Dynamic

- New art can be inserted
- User-generated environment
- Unpredictable
- Open, free ability

Virtual World Examples

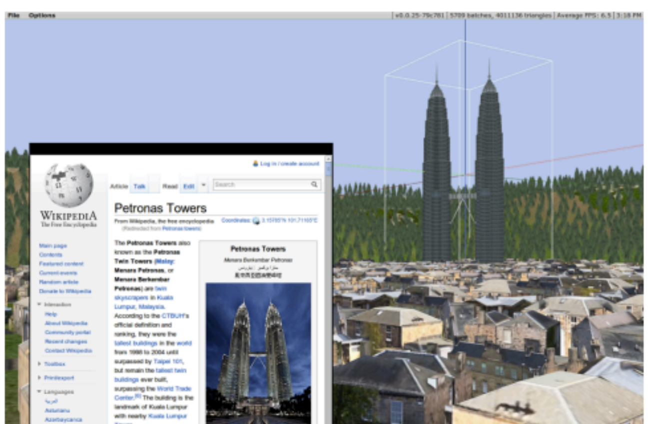
- World of Warcraft
  - Online game
  - 10 million players

- Second Life
  - Virtual world
  - Explore, socialize, trade

- EvE Online, Habbo Hotel, etc.

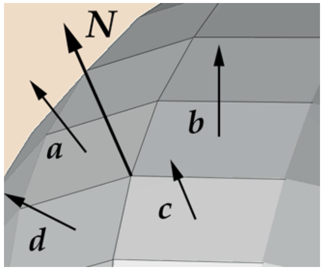
Sirikata

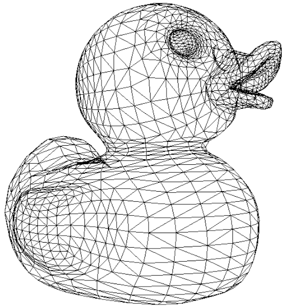
- Platform for seamless, scalable, and federated metaverses

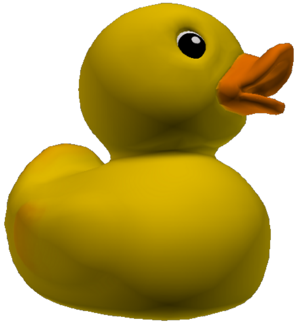


3D Content

- Mesh Representation
  - Vertex coordinates
  - Normal vectors
  - Polygon indexes
  - Textures
  - Texture coordinates





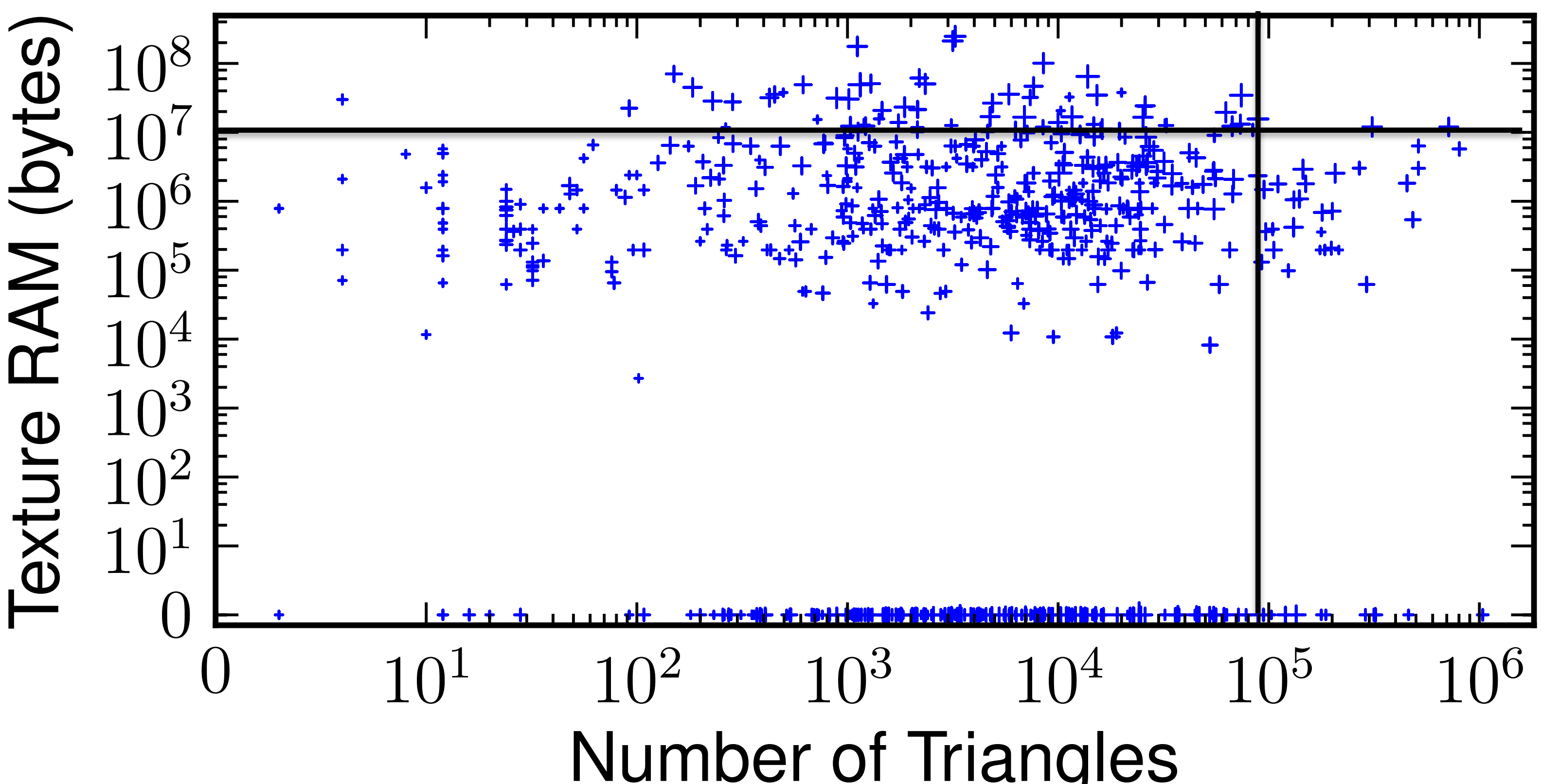


Importing Content

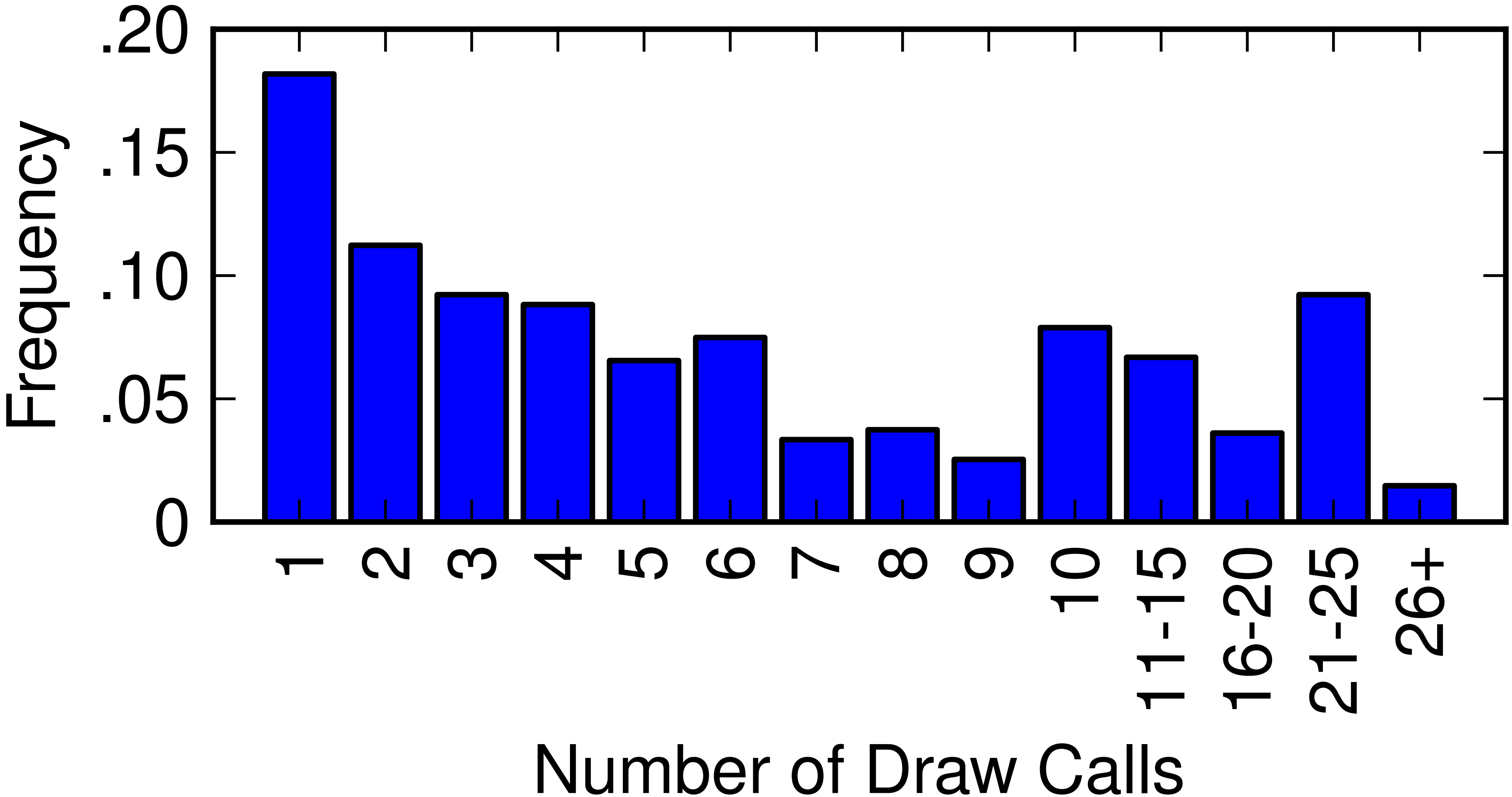
- GPU limits for interactive frame rates
  - **triangles** (millions)
  - **texture RAM** (256MB – 2GB)
  - **batches / draw calls** (thousands)
- Static worlds
  - Artist works closely with developers
  - Pre-processed
- Dynamic worlds
  - Arbitrary, user-generated content

Gathering Content

- Summer 2011
- 15 students at Stanford and Princeton
- Uploaded 3D models to website



Draw Call Distribution



Possible Solutions

- Enforce limits on triangles, textures, and draw calls
  – Decreases usability
  – Reduces available content

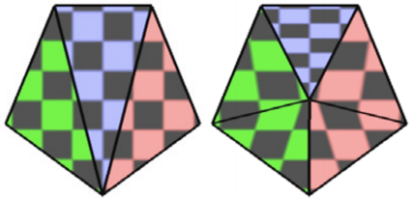
- We can do better!
  – Automatically condition the content into efficient format

Conditioning Goals

1. Reduce Draw Calls
   – 1 per object
2. Reducing Texture Space
   – To fit more textures into RAM
3. Simplify Mesh
   – Complex meshes can be drawn at lower resolution
4. Progressive Transmission
   – Display low-resolution first, streaming more detail
   – Great for low-bandwidth links or distant observers

Conditioning

- Mesh Simplification
  - Well studied area
    - *Mesh Optimization* [Hoppe ‘93]
    - *Surface simplification using quadric error metrics* [Garland ‘97]
    - *Appearance preserving simplification* [Cohen ’98]
  - Problems with progressive models



- Retexturing + simplification
  - Existing methods
    - *Texture mapping progressive meshes* [Sander ‘01]
  - Supervised algorithm, small testing set

Conditioning Pipeline

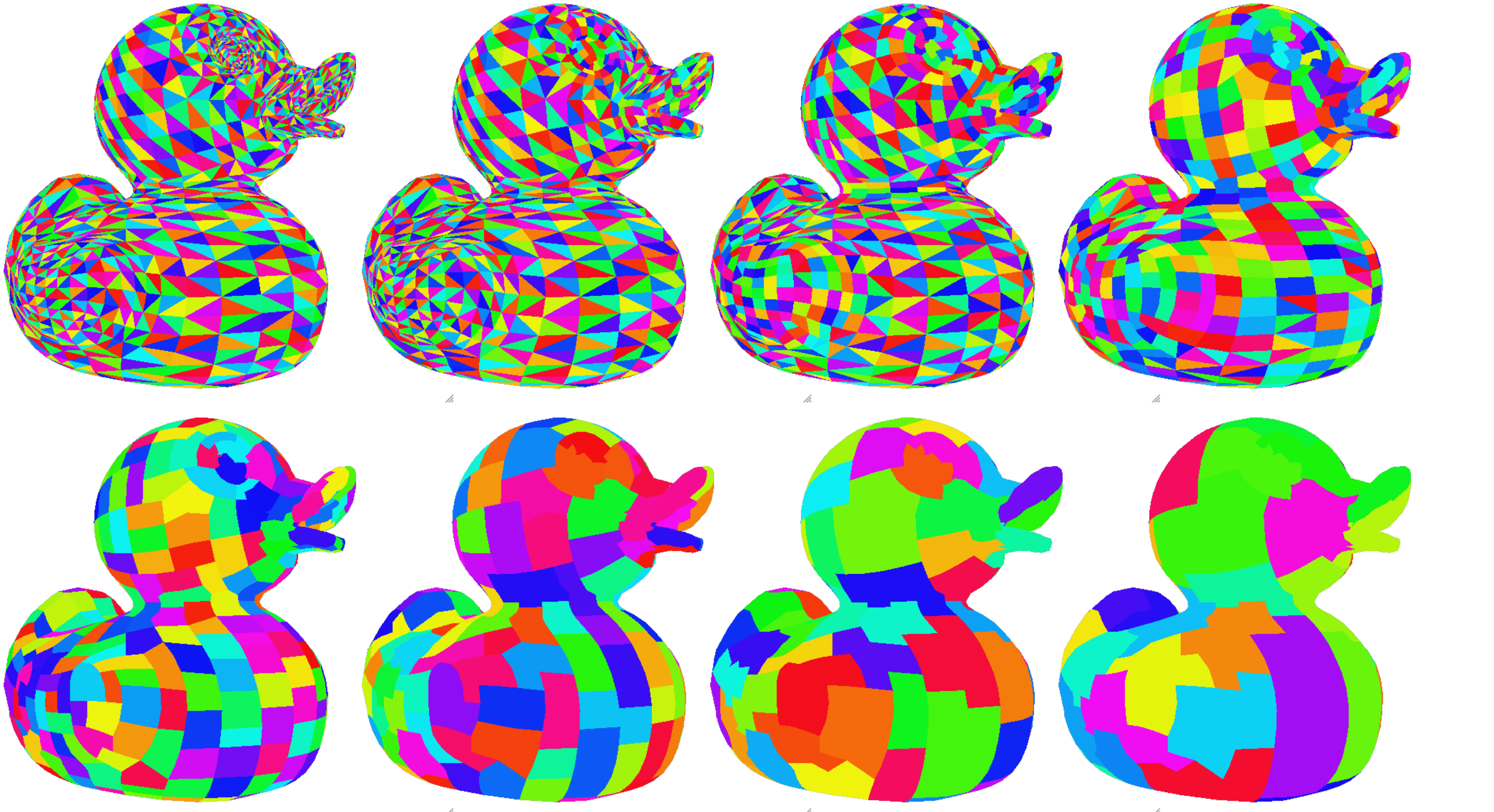
1. Cleaning and normalizing
2. Chart creation
   – contribution: unsupervised
3. Fair allocation of texture space to charts
   – novel technique
4. Mesh simplification
5. Progressive, streamable encoding
   – contribution: efficient format

Cleaning and Normalizing

- All polygons are converted to triangles
- Missing vertex normals are generated
- Extraneous data is deleted
- Complex scene hierarchies and instanced geometry is flattened to a single mesh
- Vertex data is scaled to a uniform size

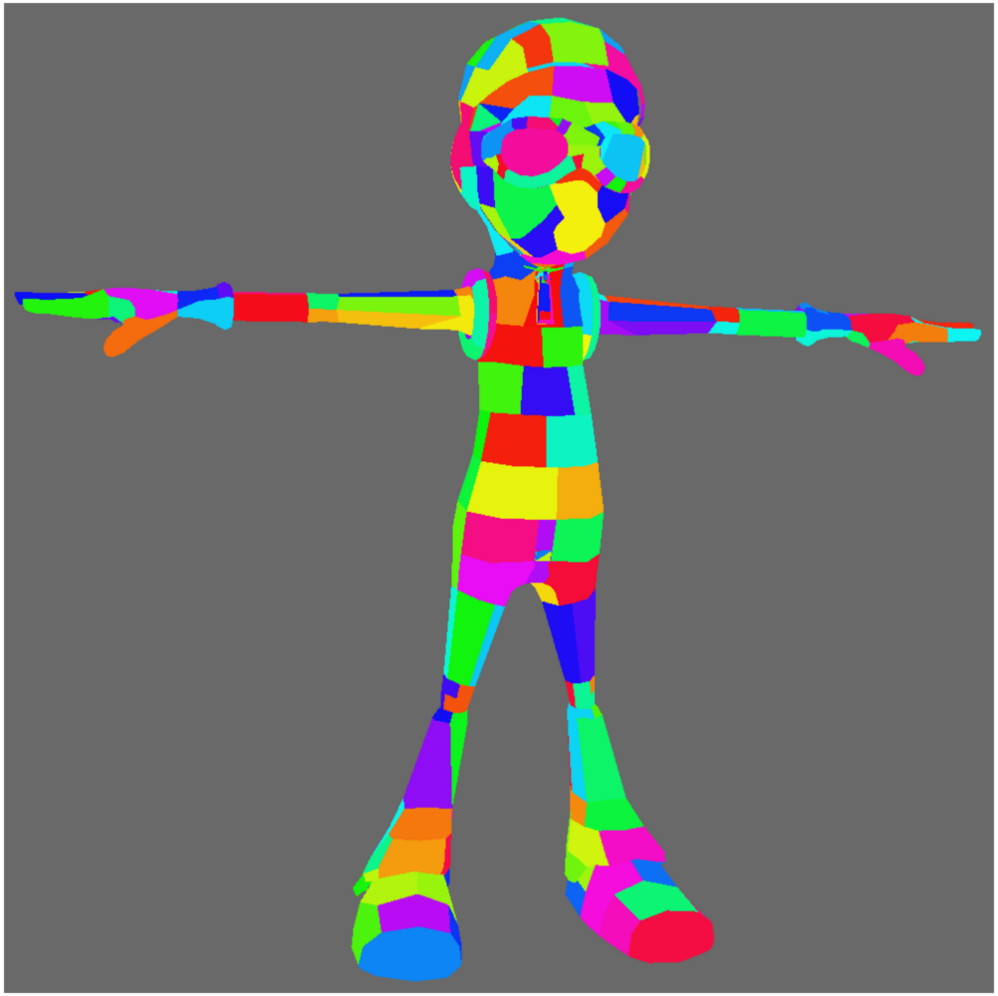
Creating Charts

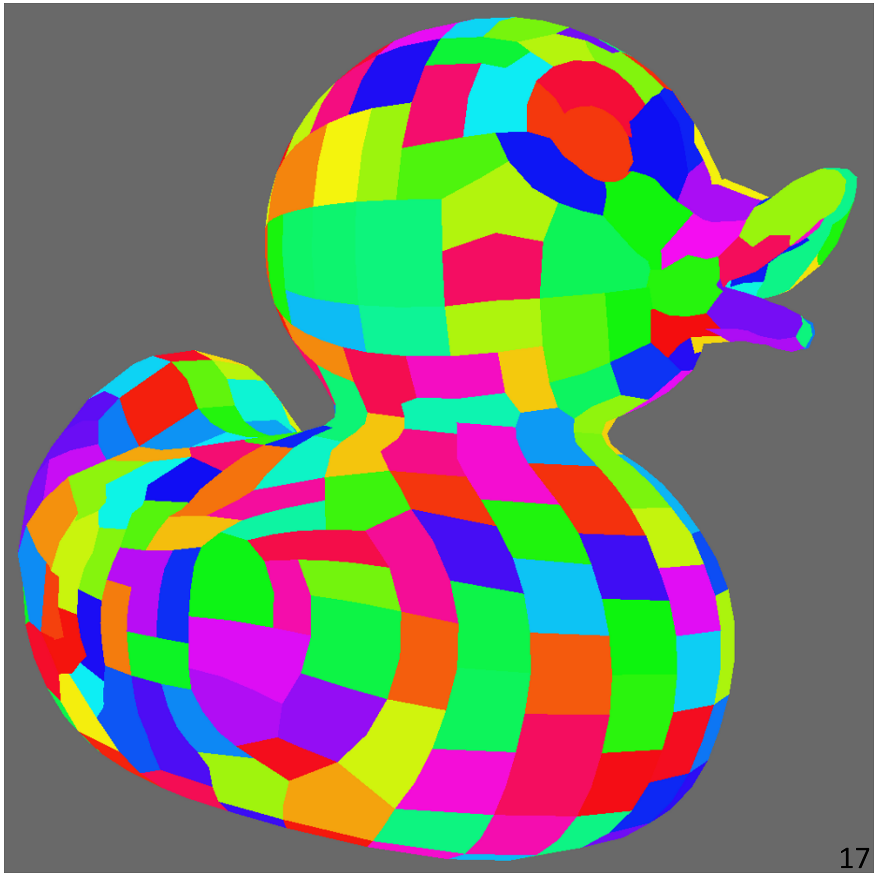
- Retexturing
  - Creates new, single texture from model
- Each triangle could be placed in texture
  - Not great for simplification
- Instead, partition mesh into flat regions
- Starts with a chart for every triangle
- Priority queue of chart merges
  - Ordered by error term incorporating *compactness* and *planarity*



- Flat, disc-like regions
- Compact

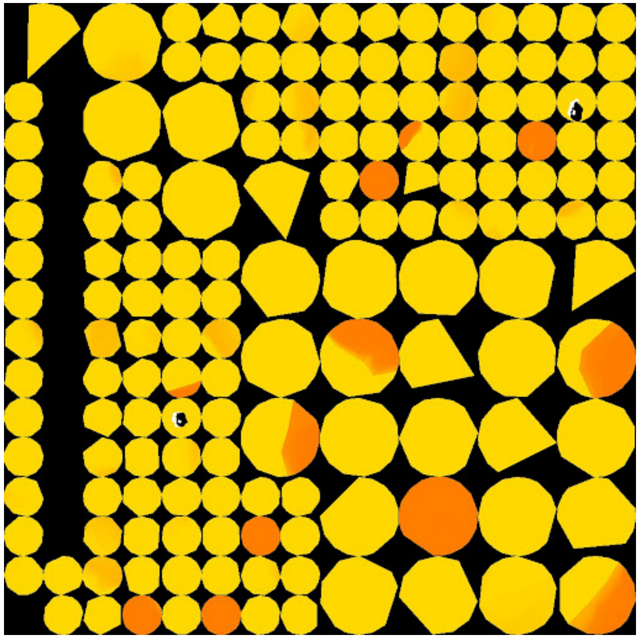
Heuristic Examples





Allocating Texture Space

- Each chart is parameterized from 3D space to 2D texture space
- Each chart is given a size in 2D space



Allocating Texture Space

- Original technique [Sander '01]
  - $L^2(T)$ - root-mean-square stretch
  - $L^\infty(T)$ - maximum stretch
- $L^2(T)$ is used because
  - *"unfortunately there are a few triangles for which the maximum stretch remains high"*
- With our larger set of models, so is L^2!
- A chart with high $L^2(T)$ can allocate too much space, leaving little room

Allocating Texture Space

$$A_c^{''} = \sqrt[3]{(\frac{L_c^2}{\sum L^2})(\frac{A_c}{\sum A})(\frac{A_c'}{\sum A'})} \cdot T$$

- L^2_c - chart's texture stretch
- A_c - chart's surface area in 3D
- A'_c - chart's area in the original texture
- ΣL^2, ΣA, $\Sigma A'$ – sum across all charts

Mesh Simplification

- We use technique based on [Garland ‘97] and [Sander ’01] using quadric error and texture stretch
- See paper for unsupervised stopping heuristic

Ideal Progressive Encoding

1. Simplified base mesh can be downloaded and displayed without downloading the rest
2. Vertex data can be streamed, allowing a client to continuously increase mesh detail
3. The mesh’s texture can be progressively streamed, allowing a client to increase texture detail

File Format

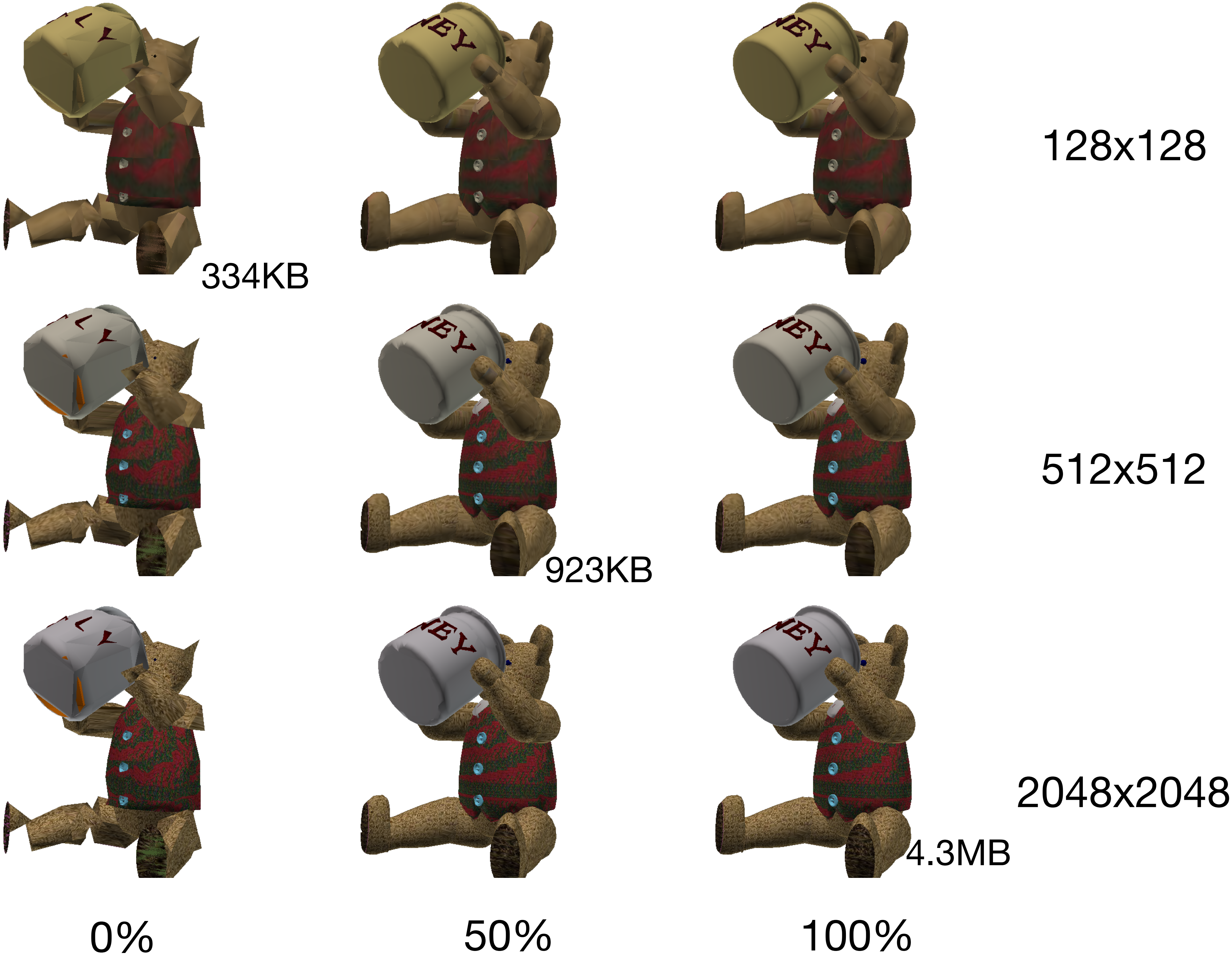
- Existing formats
  – OBJ, STL, PLY, FBX (60 listed on Wikipedia)
- COLLADA
  – Open standards-based XML format (2006)
  – Widely supported: SketchUp, Blender, 3DS Max, Maya, Autodesk, Google Earth
  – pycollada maintainer
- But there are no existing usable progressive formats

Base Mesh & Refinements

- Base mesh encoded as COLLADA
  - backwards compatible, unmodified clients
- Progressive vertex data is a list of refinements: vertex additions, triangle additions, index updates

Progressive Textures

- No suitable progressive image formats
  - JPEG 2000, gif
    - Memory buffer requires O(full resolution) size
  - Microsoft DDS format
    - fixed-point only (like png)
    - not well supported
- Full resolution is resized to multiple LODs
  - 1x1, 2x2, 4x4, … 512x512, 1024x1024, …
  - Also called mipmaps, each encoded as JPEG
  - Concatenated together into TAR file
- Achieves good compression
- Allows client to index into file, e.g. HTTP Range request

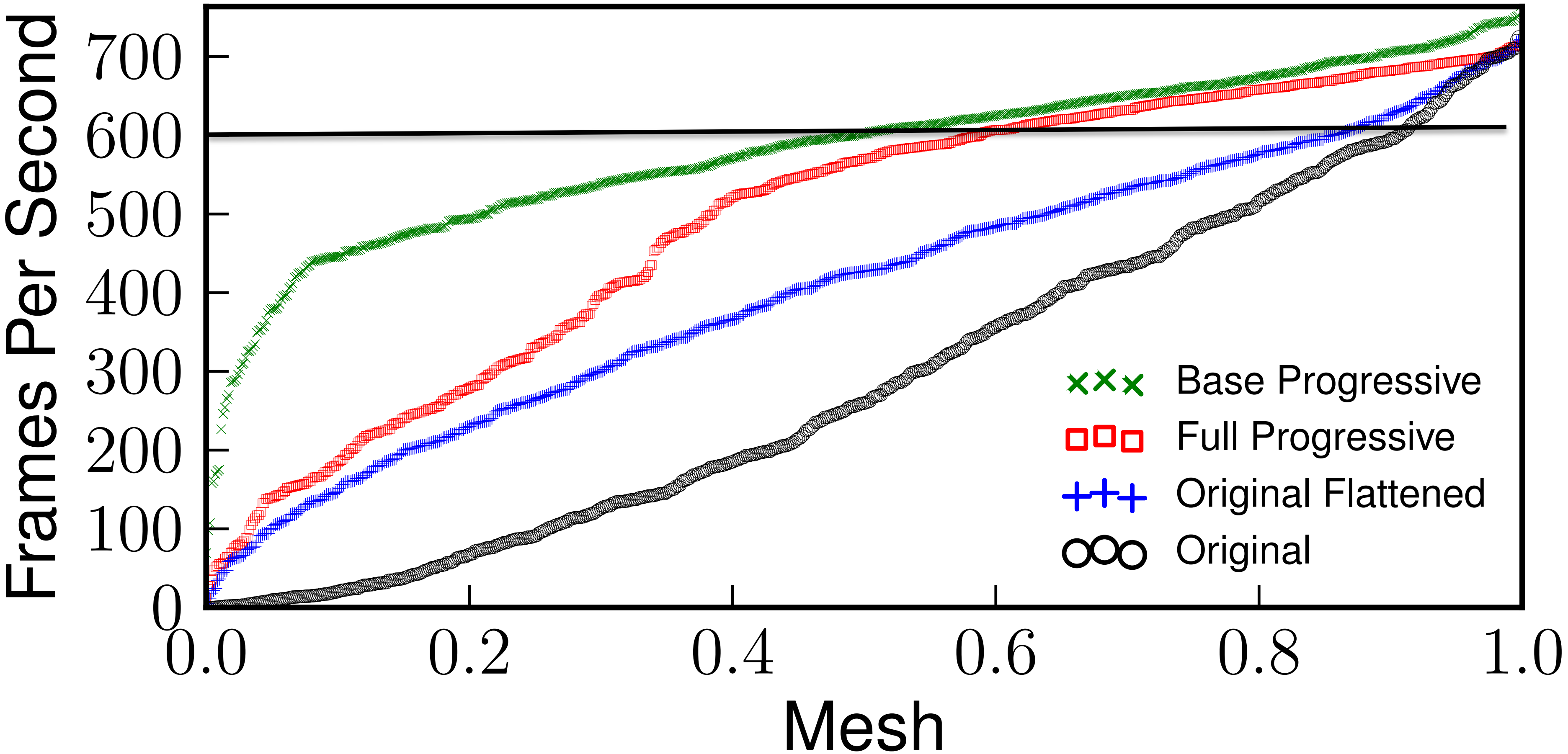


NEY
334KB
923KB
4.3MB
128x128
512x512
2048x2048
0%
50%
100%

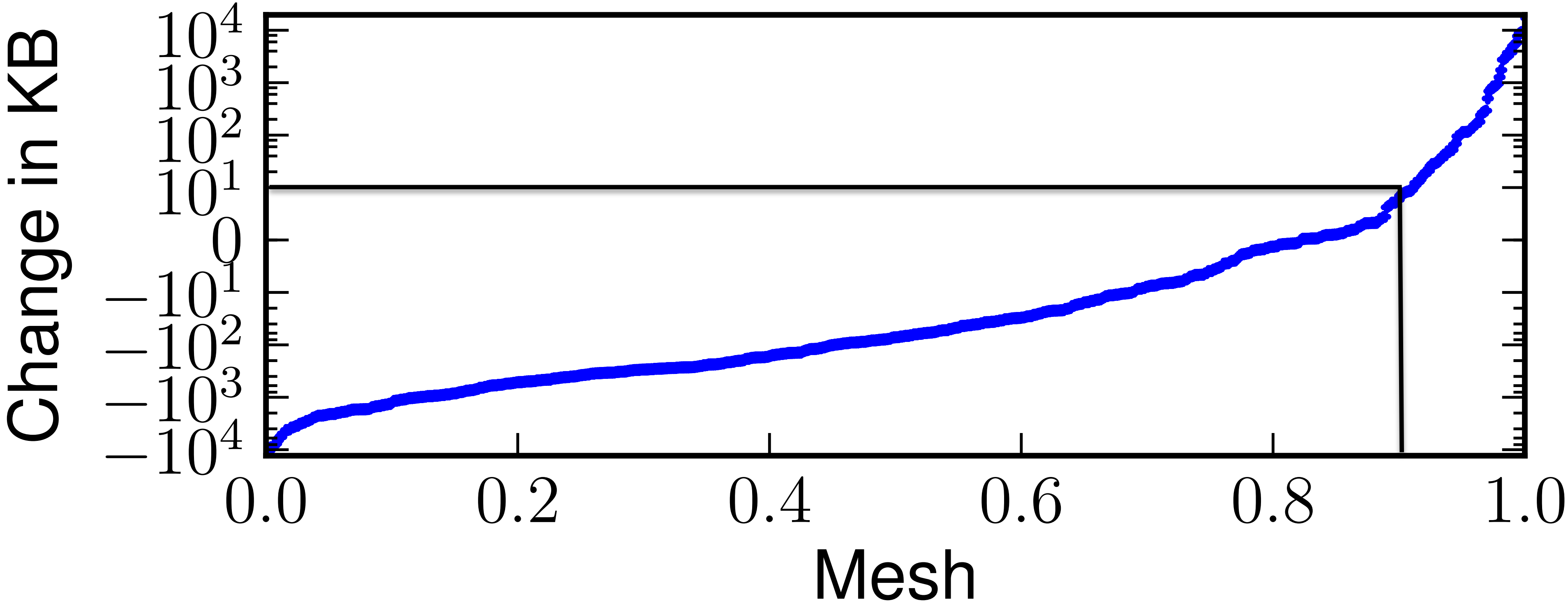
Evaluation

- Render efficiency
  - How much does batching help?
- File Size
  - How does conditioning affect file size?
- Perceptual Error
  - How much does conditioning change how models look?

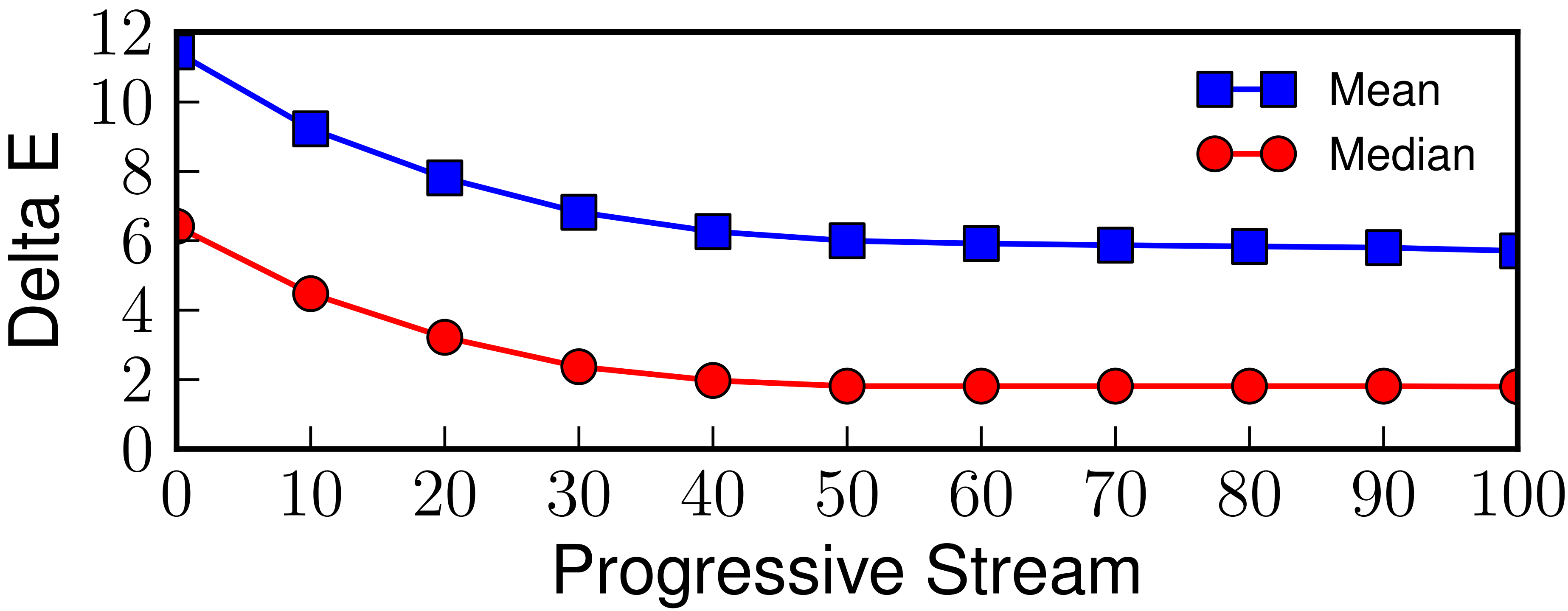
Render Efficiency



File Size – Base Mesh



Perceptual Error



- Delta E < 1 not noticeable by average human
- Delta E of 3-6 are commonly-used tolerances for commercial printing

Conditioning Contributions

- Unsupervised
- Apportioning texture space fairly
- Efficient progressive encoding
- A complete, robust conversion framework

Questions?

open3dhub.com

sirikata.com

@jterrace

jterrace@cs.princeton.edu