Mesh Reduction - Triangle Strips

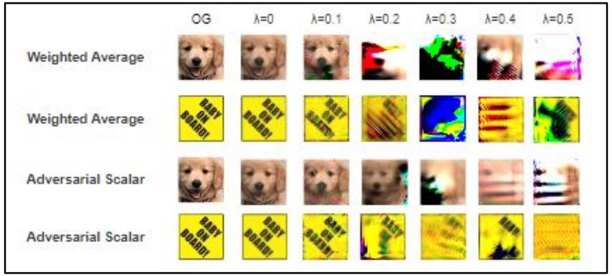
By Elijah Berberette & Hunter Price

Test Questions

1. What is a Frame and Frame Rate and why should you care about it?
2. Why would one use Triangle Strips?
3. What kind of data is typically tied to a face/triangle in a mesh?

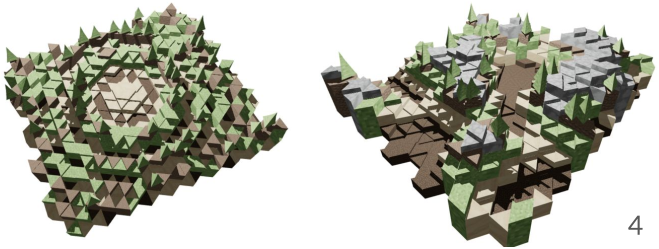
Elijah Berberette

- Degree:
  - Computer Science Masters (maybe PhD)
  - Concentration: Intelligent Systems and Machine Learning
  - Advisor: Dr. Jens Gregor
- Interests
  - Machine Learning/Deep Learning
  - Software Engineering
- Goals
  - Publish one DL paper by the end of the year
  - Become an ML engineer
- Hometown: Lobelville, TN



Hunter Price

- Degree:
  - Computer Science Masters (maybe PhD)
  - Concentration: Scientific Visualization
  - Advisor: Dr. Jian Huang
- Interests
  - 3D Visualization
  - Machine Learning/Deep Learning
- Goals
  - Discover how to intertwine HCI, Visualization, and DL to model and share intelligence.
- Hometown: Kelso, Washington

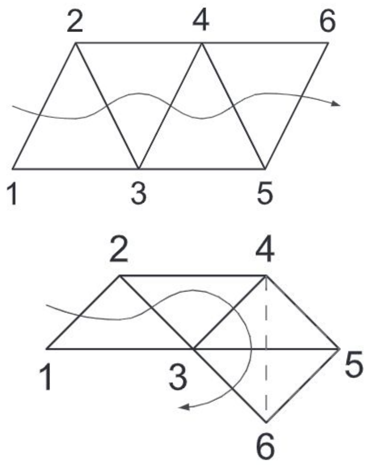


Outline

- Overview of Graphics and the Graphics Pipeline
- History Of Graphics & Mesh Reduction
- The Basic Triangle Strip Generation Algorithm
- Applications of Mesh Reduction & Triangle Strip Generation
- Live Demo & Collected Data
- Open Issues In Mesh Reduction
- References & Discussion
- Test Questions Revisited

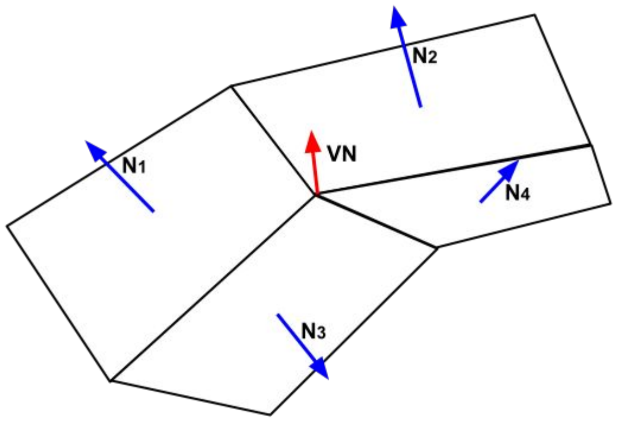
Overview

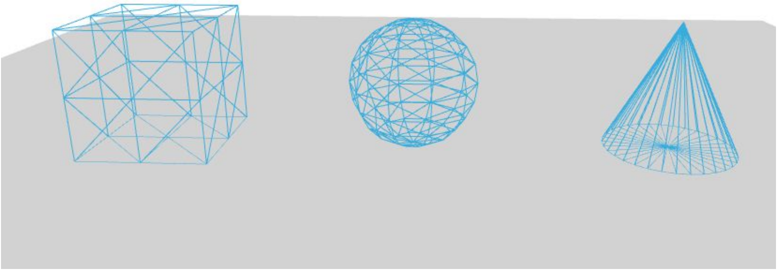
- Computer Graphics
  - All about how to model light transport
- Rendering
  - Generating images from some internal model
- Mesh
  - Collection of vertex and face data
- Framebuffer/Frame
  - A $h \times w \times 3$ buffer holding pixel data to be displayed on the screen
- Triangle Strip (Sequential Tristrip)
  - “a sequence of n+2 vertices that represent n triangles” [Vaněček]



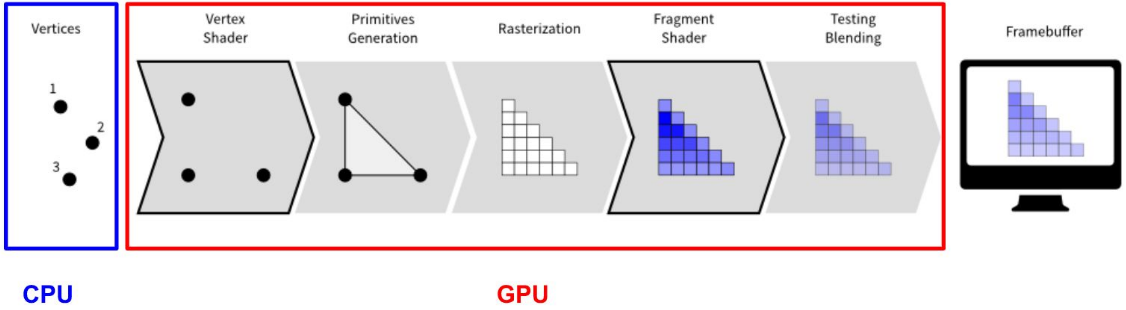
What is a triangular mesh?

- Collection of:
  - Vertices - v: <x, y, z>
  - Vertex Normals - vn: <x, y, z>
  - Texture Coordinates - vt: <u, v>
  - Face Specifications - f: < v/vn/vt, v/vn/vt, v/vn/vt >
- Can create complex geometries out of simple shapes





Rendering A Frame



- For each triangle we want to render we need:
  - 3 vertex coordinates
  - 3 vertex normals
  - (~0-6)*3 vertex texture coordinates
- Pass data from CPU to GPU
- GPU does computation and returns a frame.
- Generally want to do this ~30 times a second
- Issue?
  - PCIe 3.0 x16 Bandwidth is 16 GB/s
  - PCIe 4.0 x16 Bandwidth is 32 GB/s

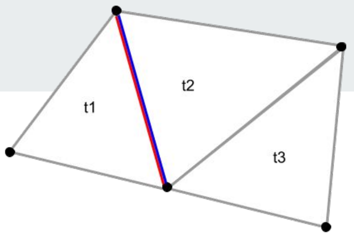
```
void draw_triangle(
  float[] v1, float[] v2, float[] v3
  float[] n1, float[] n2, float[] n3
) {
  // draw a triangle
  glBegin(GL_TRIANGLES);
    // vertex 1
    glNormal3f(n1[0], n1[1], n1[2]);
    glVertex3f(v1[0], v1[1], v1[2]);
    // vertex 2
    glNormal3f(n2[0], n2[1], n2[2]);
    glVertex3f(v2[0], v2[1], v2[2]);
    // vertex 3
    glNormal3f(n3[0], n3[1], n3[2]);
    glVertex3f(v3[0], v3[1], v3[2]);
  glEnd();
}
```

History

- Graphics
    - First computer graphics system developed at MIT 1963 named Sketchpad.
    - PHIGS - one of the first ISO attempts at a standard graphics library in the 80’s.
    - Silicon Graphics Inc (SGI) flourished in the late 80’s - 90’s with a proprietary API named IRIS GL.
    - SGI open sourced their GL as an open standard named OpenGL.
    - WebGL released in 2011.
- Tri Strip Generation Algorithms
    - SGI - SGI Algorithm (1990)
    - STRIPE - Optimizing triangle strips for fast rendering (1996)
    - Tunneling - Tunneling for Triangle Strips in Continuous Level-of-Detail Meshes (2001)

Triangle Strip Generation - High Level Overview

- NP-Complete Problem
  - Analogous to the Hamiltonian path problem
- Typically done only once on a mesh
- Algorithm Time Complexity: $O(n + n*s)$
  - n: number of triangles
  - s: number of strips
- General Steps
  1. Read Triangular Mesh & Create Adjacency Graph
  2. Run Graph Through TriStrip Generation Algorithm



Preprocessing Data - Adjacency Graph

```
struct Triangle {
  int vRef[3];
  int aTri[3];
}
```

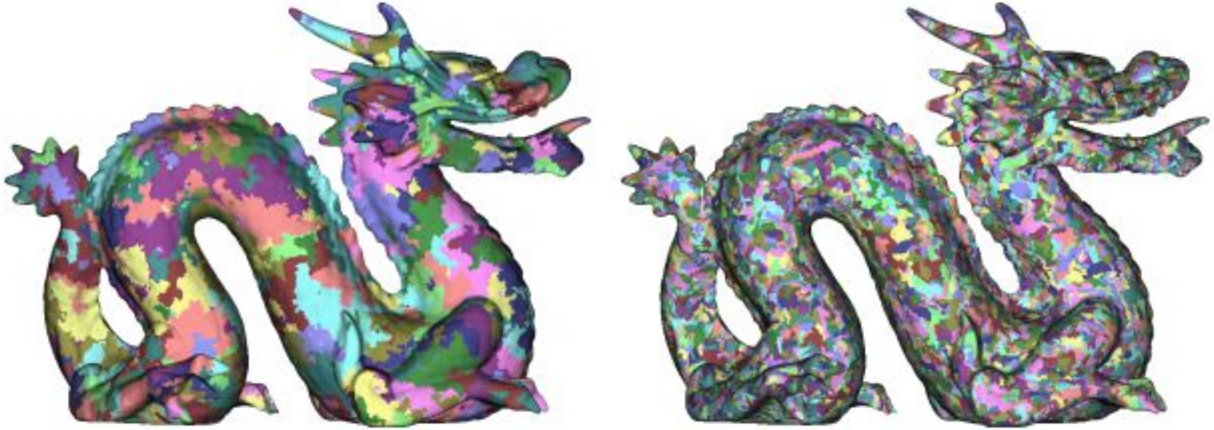
```
struct Edge {
  int ref0;
  int ref1;
  int faceNb;
};
```

- Steps:
  1. Create a list of **Triangle** Objects for each face in mesh.
     a. fill the ***vRef*** list with indices into vertex array, and file the ***aTri*** list with default values.
  2. Create empty list to hold **Edge** Objects.
  3. Iterate through each triangle create 3 **Edge** objects connecting the vertices and add them to the list of edges.
     a. ensure ***ref0*** has the smaller index between ***ref0*** and ***ref1***
  4. Sort the edge list 3 times in order of ***faceNb***, ***ref0***, ***ref1***
  5. Iterate through list of edges if adjacent edges share both ***ref0*** and ***ref1***, connect the two faces they represent.

SGI-based Algorithm [Vaněček]

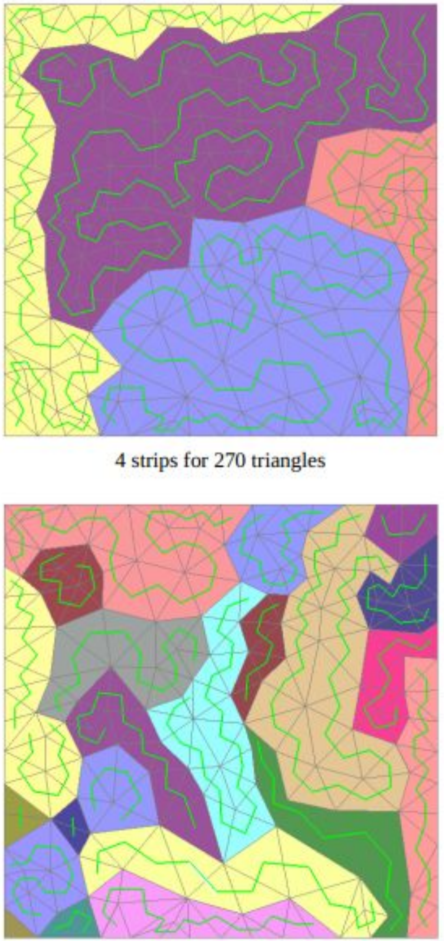
- Steps:
    1. If there are no more triangles in the triangulation then exit
    2. Find the triangle ***t*** with the least number of neighbors
    3. Start a new strip
    4. Insert the triangle ***t*** to the strip and remove it from the triangulation
    5. If there is no neighboring triangle ***t*** then go to step 1.
    6. Choose a new triangle **t**, neighboring to triangle t, with the least number of neighbors. If there is more than one triangle ***t'*** with the same least number of neighbors, look one level ahead, if there is a tie again, choose ***t'*** arbitrarily.
    7. ***t*** <- ***t'***. Go to step 4.

Tunneling for Triangle Strips in Continuous Level–of–Detail Meshes [2001]



Tunneling algorithm – 1,798 strips

SGI algorithm – 17,653 strips



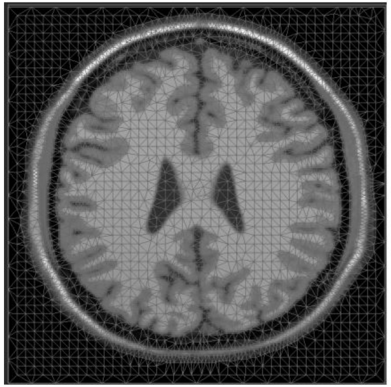
4 strips for 270 triangles

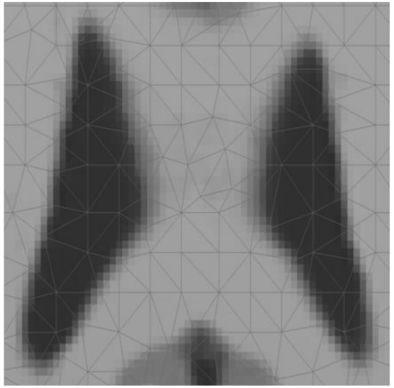
22 strips for 270 triangles

Figure 1: Good and bad stripifications

Applications

- Scientific Visualization (Medical Imaging)
- Video Games (Unity & Unreal Engine)
- CAD
- Revit





Live Demo

- https://cosc581-project.github.io/

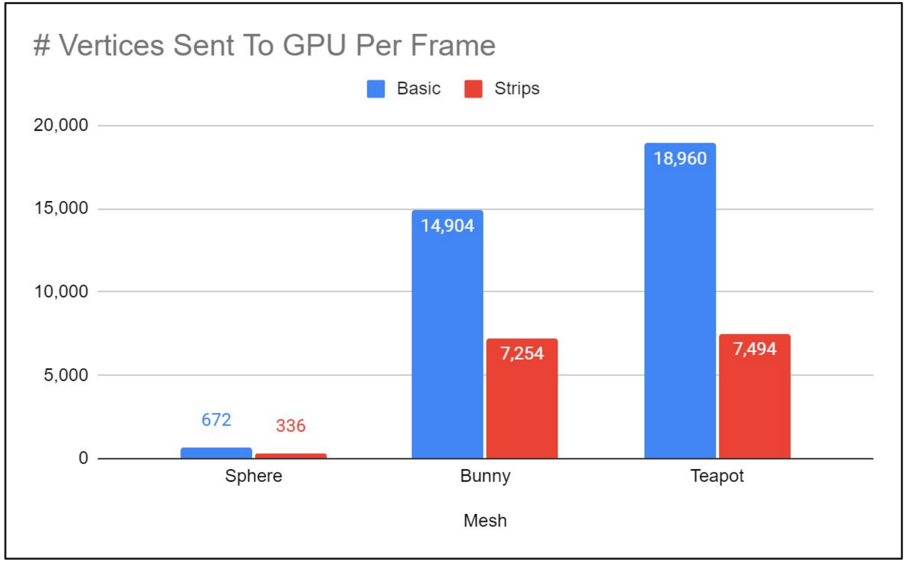
Basic vs Tristrip Comparison

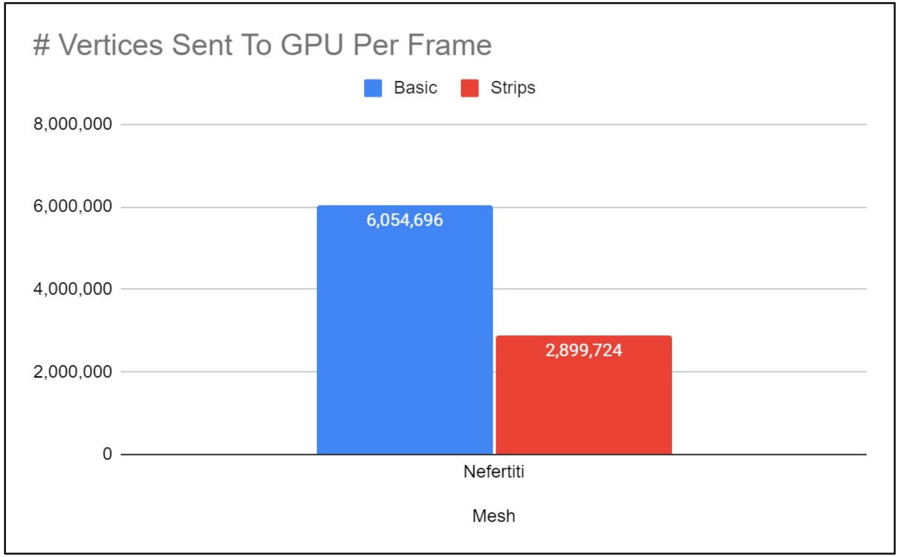
Mesh	# Vertices	# Faces	# Strips	# Strip Vertices
Sphere	114	224	56	336
Bunny	2,403	4,968	1,143	7,254
Teapot	3,644	6,320	587	7,494
Nefertiti	1,009,118	2,018,232	440,746	2,899,724

Basic vs Tristrip Comparison - Per Frame (30 fps)

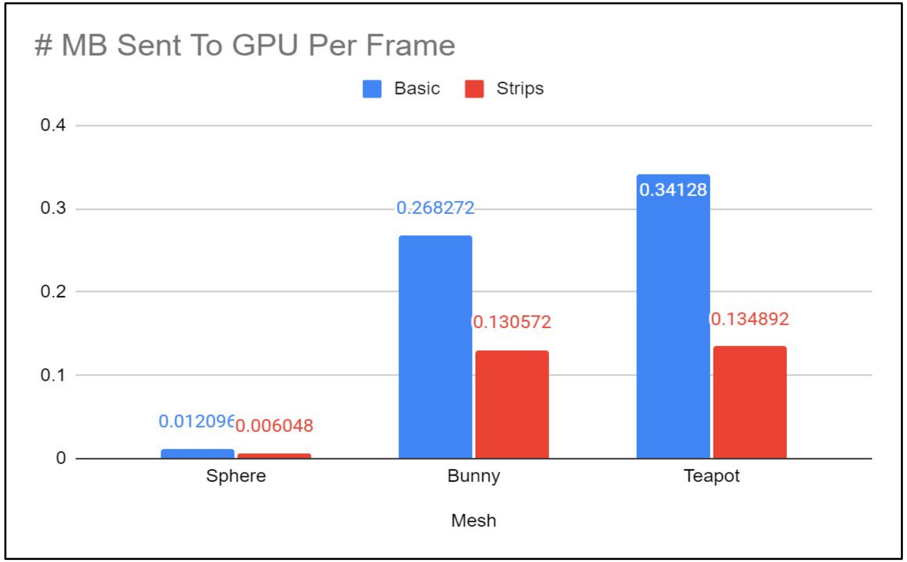
Mesh	# Vertices/Frame	MB/Frame	# Vertices/Frame	MB/Frame
Sphere	672	0.012	336	0.006
Bunny	14,904	0.27	7,254	0.131
Teapot	18,960	0.34	7,494	0.135
Nefertiti	6,054,696	108.98	2,899,724	52.2

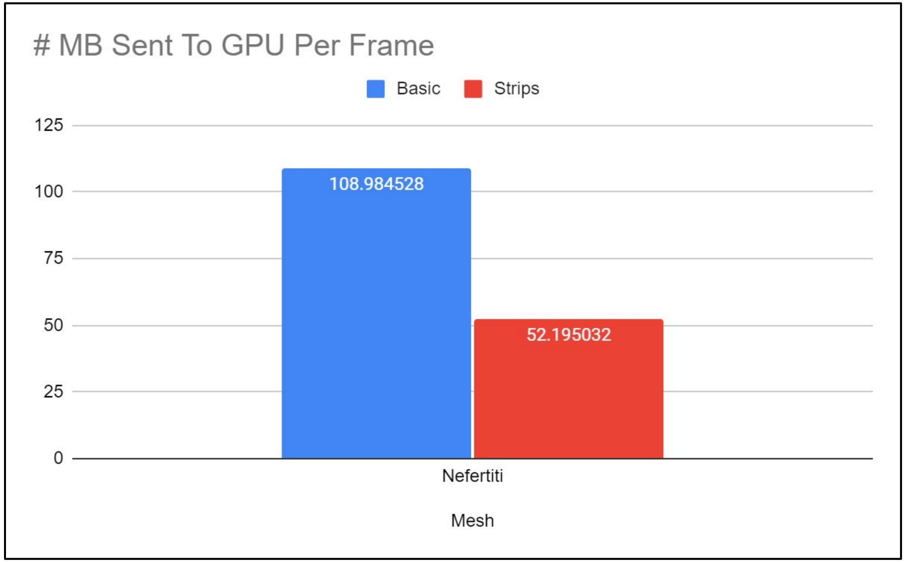
Vertices sent to the GPU



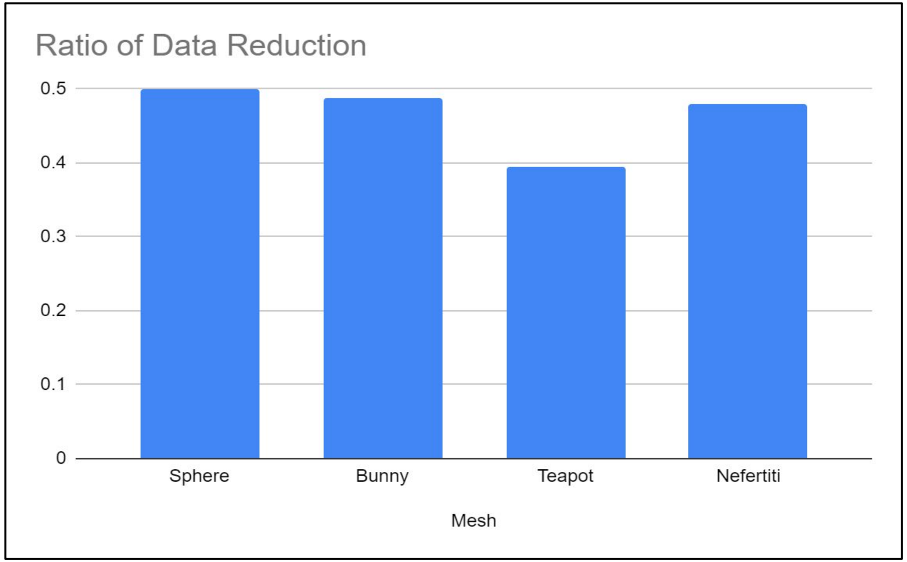


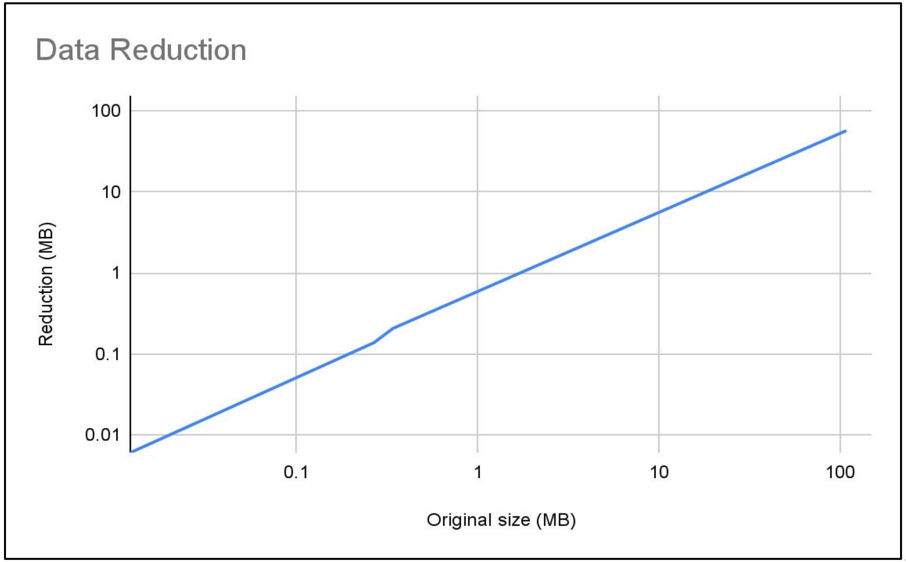
MB sent to the GPU





Data Reduction





Open Issues

- Triangle Strip Generation itself is a generally solved problem
    - It is “good enough”
- Making 3D Visualization More Available
    - How do people without access to expensive GPUs use 3D visualization?
- Data Has Increased In Size
    - How do we visualize exascale data?
- Dynamic Mesh Reduction and Level Of Detail (LOD)
    - How do we render “nice” looking graphics without maxing out the hardware?

References

Akeley, K., Haeberli, P., Burns, D.; tomesh.c. C Program on SGI Developer's Toolbox CD, 1990

Evans, Francine, Steven Skiena, and Amitabh Varshney. "Optimizing triangle strips for fast rendering." Proceedings of Seventh Annual IEEE Visualization'96. IEEE, 1996.

Stewart, A. James. "Tunneling for triangle strips in continuous level-of-detail meshes." Graphics interface. Vol. 2001. 2001.

Vaněček, Petr, and Ivana Kolingerová. "Comparison of triangle strips algorithms." Computers & Graphics 31.1 (2007): 100-118.

Schroeder, Will; Martin, Ken; Lorensen, Bill (2006), The Visualization Toolkit (4th ed.), Kitware, ISBN 978-1-930934-19-1

Discussion Slide

Test Questions (Revisited)

1. What is a Frame and Frame Rate and why should you care about it?
2. Why would one use Triangle Strips?
3. What kind of data is typically tied to a face/triangle in a mesh?

Extra slides

Nefertiti Mesh

Optimizing Triangle Strips for Fast Rendering [1996]

“The cost columns show the total number of vertices required to represent the dataset in a generalized triangle strip representation under the OpenGL cost model (we are counting each vertex and swap that needs to be sent to the renderer).”

Evans, Francine, Steven Skiena, and Amitabh Varshney. "Optimizing triangle strips for fast rendering." Proceedings of Seventh Annual IEEE Visualization'96. IEEE, 1996.

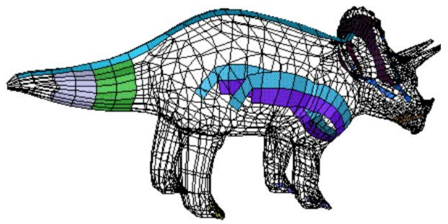


Figure 8: The six largest patches in a triceratops model.

Data File	Num Verts	Num Tris	Cost SGI	Cost Ours	Savings
plane	1508	2992	4005	3509	12%
skyscraper	2022	3692	5621	4616	18%
triceratops	2832	5660	8267	6911	16%
power lines	4091	8966	12147	10621	13%
porsche	5247	10425	14227	12367	11%
honda	7106	13594	16599	15075	9%
bell ranger	7105	14168	19941	16456	17%
dodge	8477	16646	20561	18515	10%
general	11361	22262	31652	27702	12%

Table 1: Comparison of triangle strip algorithms.

Data Sizes

- This can be scrapped and added as a note to numbers and data
- vec*n*: a vector of single-precision floating-point numbers
    - 24bits * n
- per face
    - 3 vertices (3 vec3s)
    - 3 vertex normal vectors (3 vec3s)
    - 0-8 textures (0-8 vec2s * 3)

End

Throw away slide - SGI pseudo code

Algorithm 1. Pseudo-code of the basic *SGI* algorithm.

```
procedure SGIStrip
  while there is any node in the graph do
    start a new strip
    choose the lowest degree node

    add the node to the current strip
    remove the node from the graph
    update the graph
    while there exists a neighbor of the
    current node do
      node = SGIHeuristic (node)

      add the node to the current strip
      remove the node from the graph
      update the graph
    end while
  end while
end procedure
```

Algorithm 2. Pseudo-code of the *SGI* heuristic.

```
function SGIHeuristic(node)
  choose the lowest degree neighbor of the node
  if more nodes with the same lowest degree
  exist then
    look one step ahead
  return chosen neighbor
end function
```