Applied Mesh Analysis: Automating Distant-City LODs in 'Marvel's Spider-Man 2'

Scott Kircher (Principal Engine Programmer, Graphics, Insomniac Games)

Location: Room 2006, West Hall

Date: Thursday, March 21

Time: 10:00 am - 11:00 am

Pass Type: All Access Pass, Core Pass - Get your pass now!

Topic: Programming

Format: Session

The New York City of Marvel's Spider-Man 2 is roughly twice as large as that of earlier titles in the series. At the same time, new quality and performance constraints required the creation of additional distant city LODs. The manual distant-LOD geometry pipeline of Marvel's Spider-Man and Marvel's Spider-Man: Miles Morales was no longer viable. This lecture describes how Insomniac Games transitioned to automated distant-LOD geometry generation. In particular, it covers in-depth the applications of fundamental mesh analysis concepts (dual meshes, the Euler Characteristic, curvature estimation, etc...) to several difficult problems (beyond standard reduction and remeshing) that arise from the unique constraints regarding distant city LODs present in Marvel's Spider-Man 2.

Takeaway

Attendees will gain a better understanding of how basic mesh analysis can be used to solve a variety of LOD-related problems, outside of the standard reduction and remeshing algorithms typically provided by middleware. In addition, attendees will gain insight into how the beautiful city vistas in Marvel's Spider-Man 2 were generated via automation.

About

90% · Metacritic

93% liked this video game

Google users

Marvel's Spider-Man 2 is a 2023 action-adventure game developed by Insomniac Games and published by Sony Interactive Entertainment. Wikipedia

Initial release date: October 20, 2023

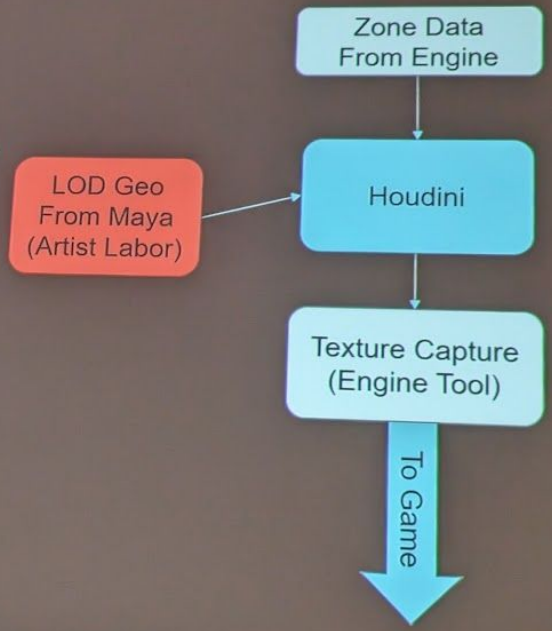
Mode: Single-player video game

Awards: Annie Award for Outstanding Achievement for Character Animation in a Video Game

Nominations: The Game Award for Game of the Year, MORE

Previous Distant-city Pipeline

- *Marvel's Spider-Man* (PS4)
- *Marvel's Spider-Man: Miles Morales* (PS4/PS5)
- *Marvel's Spider-Man Remastered* (PS5)
- Single distant-city LOD
  - Used beyond ~145 meters
  - Used for in-game map
  - Used for raytracing BVH on PS5
- Houdini driven, manual update pipeline
- Geometry constructed by hand
- Atlas/proxy textures captured by custom tool

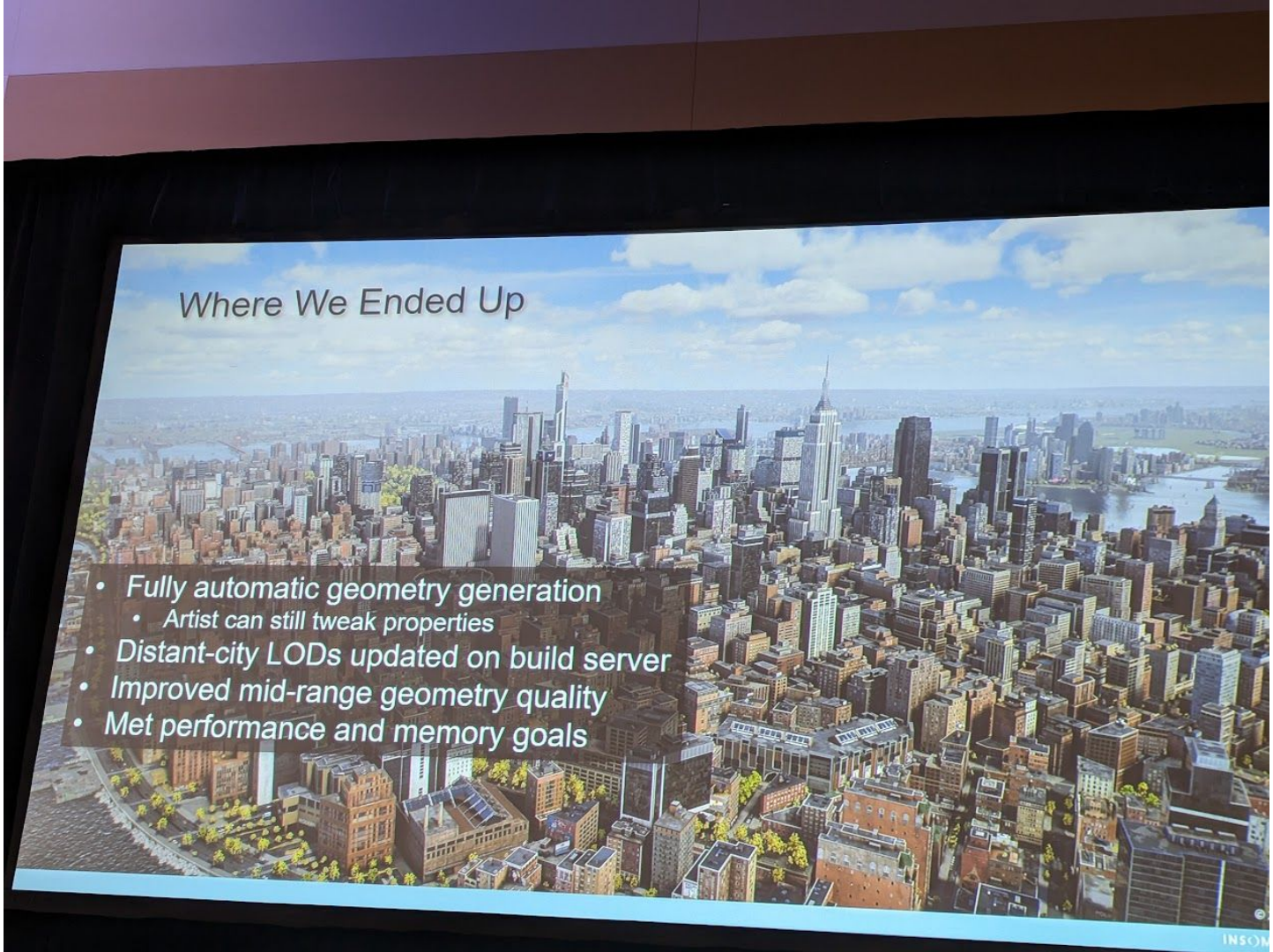


Goals for New Pipeline

- Automate!
  - Reduce manual labor
  - No more hand-made geometry
  - Auto-update LODs on build server
- Improve quality!
  - Two main LODs
    - High-density, mid-range
    - Low-density, far-range
  - Two raytracing BVH LODs
    - Medium-density, mid-range
    - Low-density, far-range

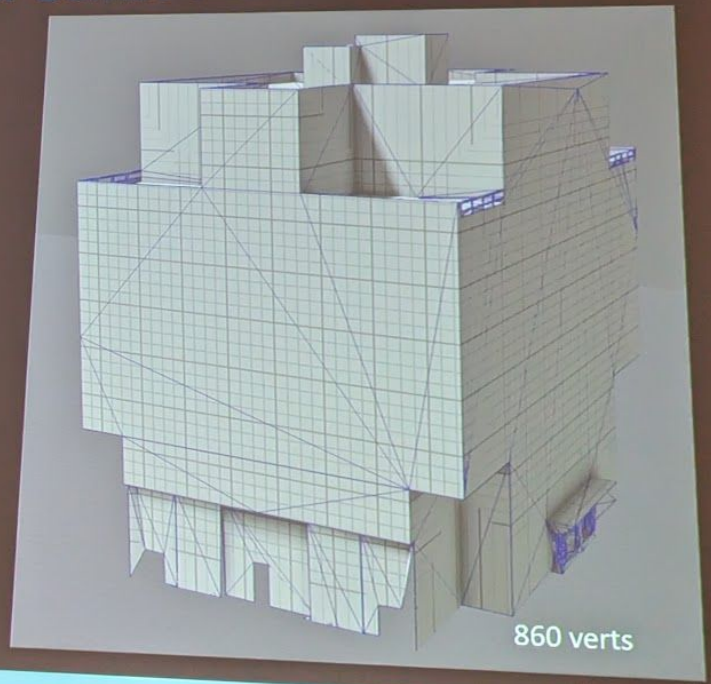
Significant Engineering Constraints

- Game almost done
  - Roughly one-half year of production remaining
  - Building construction (mostly) finished
- 16-bit index format
  - Max of 65535 vertices per building
  - Fit into existing systems & save memory
  - Try not to re-engineer a lot of unfamiliar code
- Memory & Disk Space at a Premium
  - Vertex format: 4-bytes per vertex
    - 10-bit compressed Positions
    - No vertex Normals or UVs
  - Normal maps to be used sparingly
    - Save memory
    - Required for curved surfaces, avoid otherwise


Where We Ended Up
• Fully automatic geometry generation
• Artist can still tweak properties
• Distant-city LODs updated on build server
• Improved mid-range geometry quality
• Met performance and memory goals

Remeshing + Reduction Super Combo Platter

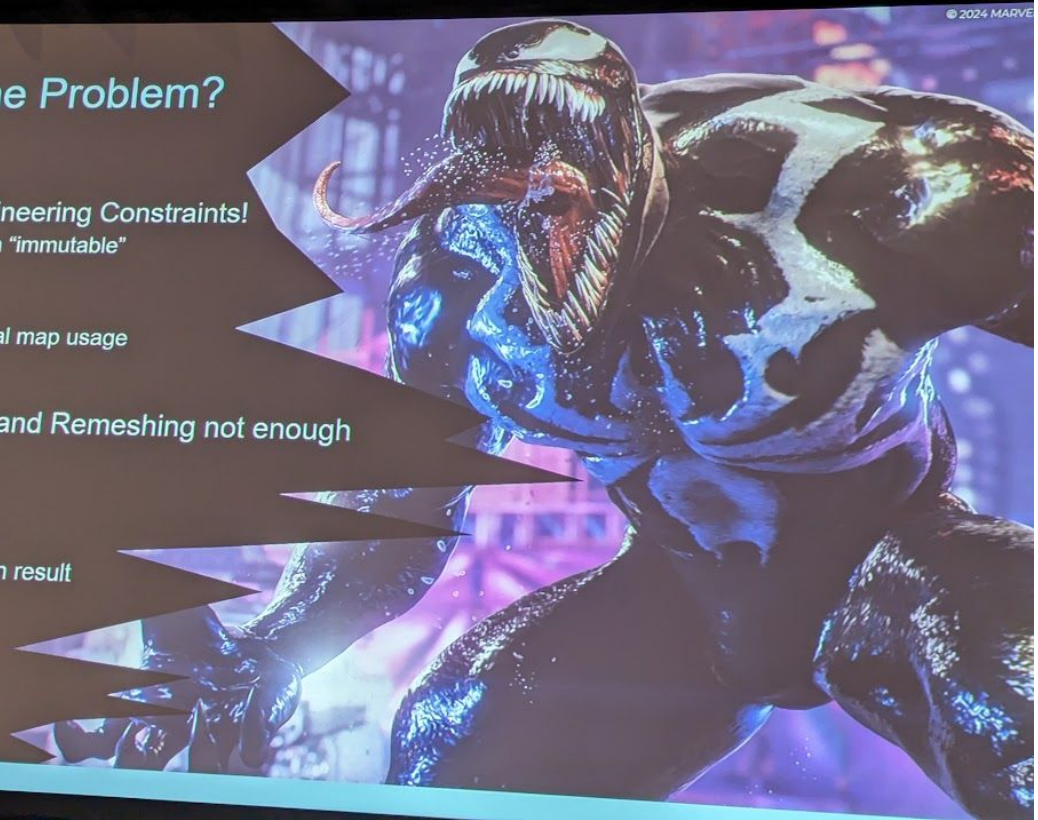
- Given our building construction
  - Reduction works well for higher-density LOD
  - Remeshing works well for very low density LOD
  - What about in between?
- First distant-city BVH LOD
  - Balance quality and memory footprint
  - Want “medium” density
  - Reduction tears apart juxtaposed pieces
  - Remeshing result not quite right
    - Too much deviation
    - OR too many output vertices
- Solution
  - Remesh to overly conservative target
  - Then reduce to desired density



So, What's the Problem?

- Remember our Engineering Constraints!
  - Building construction "immutable"
  - 16-bit indices
  - 4-byte vertices
  - Try to minimize normal map usage

- Simplygon Reduction and Remeshing not enough
- Three major problems
  - Lack of normals
  - Remeshed "inner shell"
  - Cracks in mesh reduction result

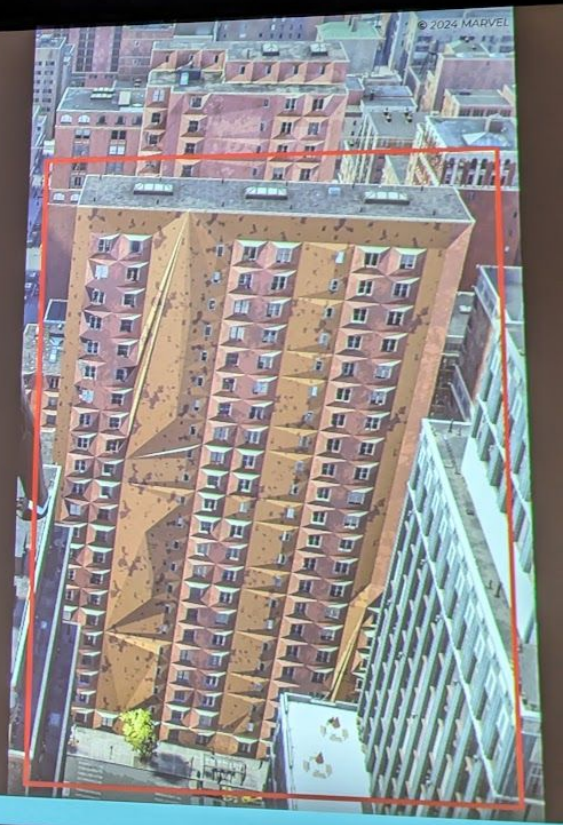


Problem #1: Lack of Normals

- Position-only vertex format
  - No vertex normals
- (Mostly) no normal maps

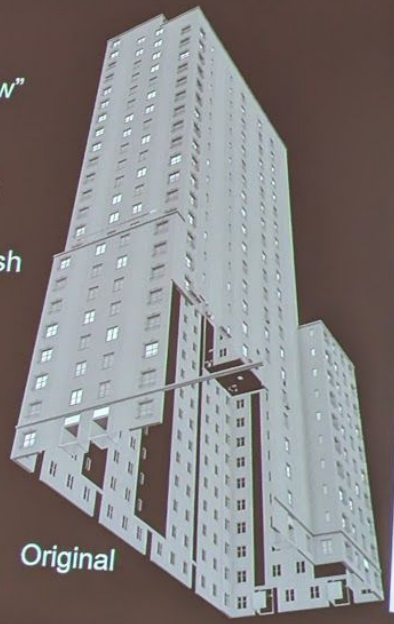
- Normals computed at runtime, per triangle
  - Old school flat-shading!

- Automatic reduction/remeshing perturbs surface
  - Formerly flat regions may not be flat anymore!
  - Faceted look
  - "Triangulation artifacts"

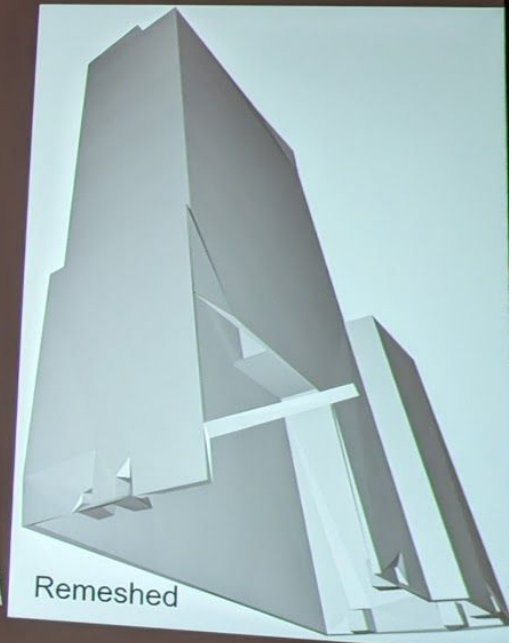


Problem #2: Remeshed “Inner Shell”

- Building construction often “hollow”
  - Generally unclosed
  - Not actual open edges
  - Backsides on some architectural tiles
- Remeshing produces a closed mesh
  - Surface wraps “inside” building
  - This is the “inner shell”



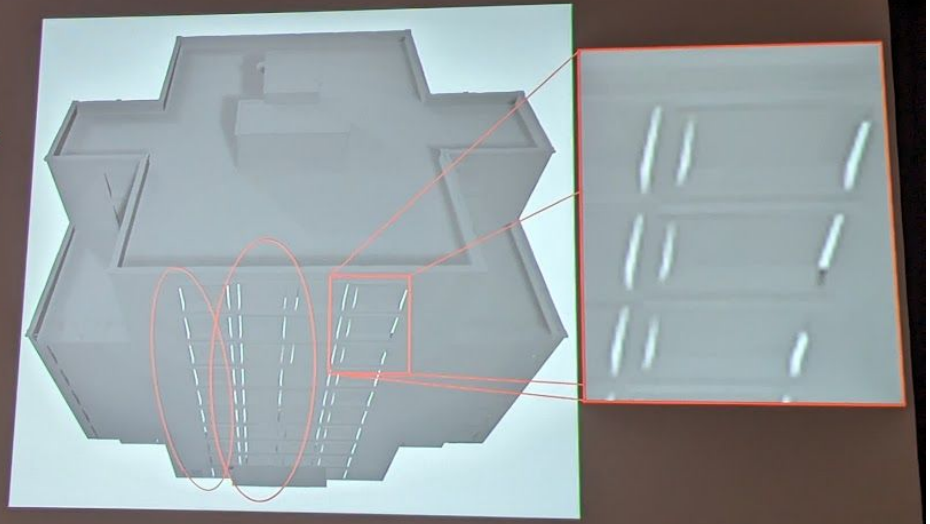
Original



Remeshed

Problem #3: Cracks in Mesh Reduction Result

- Juxtaposed architectural tiles
- Tiles may also have internal gaps
  - Disjoined/interpenetrating geometry
- Welding can't fix everything
- Reduction can move vertices
  - Opening visual gaps



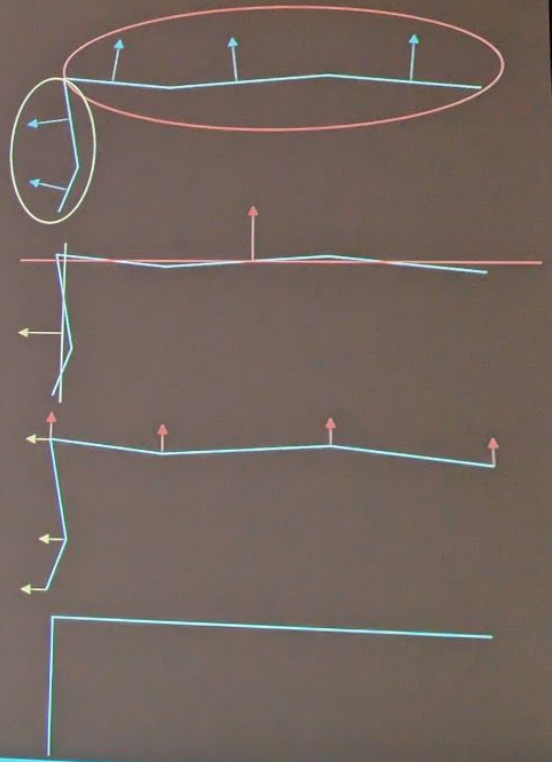
Flattening Algorithm Basic Overview

1. Cluster faces with similar normals

2. Compute target plane for each face cluster

3. Assign target planes to shared vertices
   - Up to three (linearly independent) planes can be satisfied per vertex

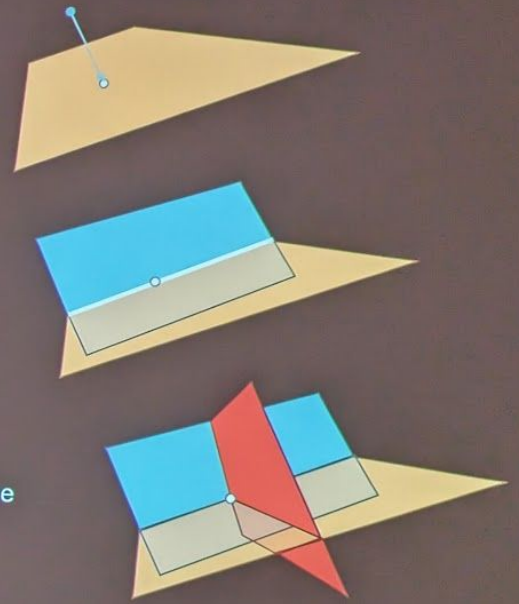
Compute new vertex positions satisfying all target planes



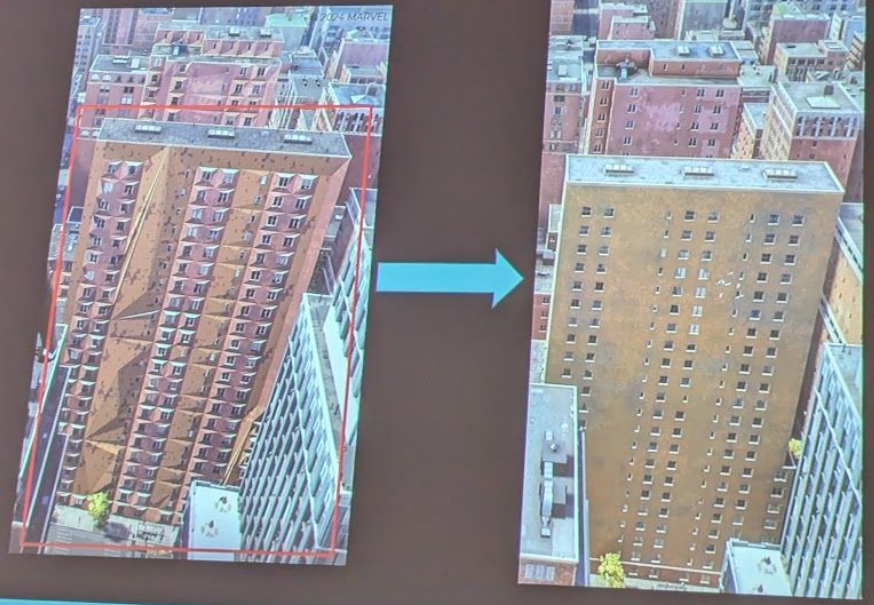
Step 4: Compute Vertex Positions Satisfying all Planes

Three cases:

1. Single plane
   - Project vertex onto plane
2. Two planes
   - Compute line of intersection between planes
   - Find closest point on line to original vertex position
   - Move vertex to that closest point
3. Three planes
   - Compute line of intersection between two of the planes
   - Compute point of intersection between that line and third plane
   - Move vertex to that intersection point



Flattening Results

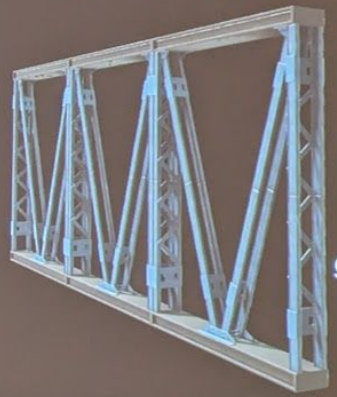


Detecting Backstop-Unsuitable Structures

- Observation: Bridges/Support-structures have lots of "holes"
- Can use *topological genus* to auto-detect
  - Buildings have relatively low genus
  - Bridges have very high genus
- Want to ignore tiny cracks and non-manifoldness
  - Simplygon Remeshing gives us manifold with no "small" holes
  - Compute genus of remeshed result!
- Also use a surface area threshold
  - Buildings tend to have large surface area
  - Statues and other random objects have lower surface area

$$\chi = V - E + F$$

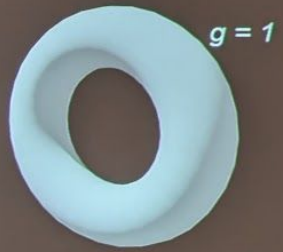
$$g = \frac{2 - \chi}{2}$$



g = "a lot"

A Little More About Topology: Genus

- *Genus* is the number of "holes" in the shape
  - *Not* holes made by boundary edges
  - Measure of topological "complexity"

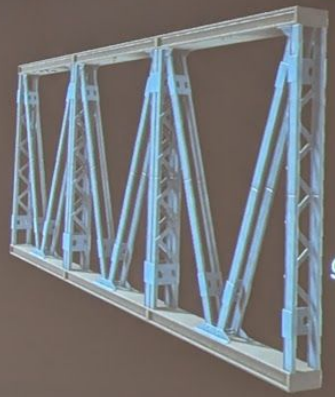


- Euler characteristic

$$\chi = V - E + F$$

- Can be used to compute genus for *manifold* triangular meshes

$$g = \frac{2 - \chi}{2}$$



g = "a lot"