Real-Time Hair Simulation and Rendering on the GPU

Sarah Tariq

Louis Bavoil

SIGGRAPH2008

NVIDIA

Results

- 166 simulated strands
- 0.99 Million triangles

- Stationary: 64 fps
- Moving: 41 fps

- 8800GTX, 1920x1200,
- 8XMSAA

SIGGRAPH2008

Results

- 166 simulated strands
- 2.1Million triangles
- Stationary: 24fps
- Moving: 17.5fps
- 8800GTX, 1280x1024,
- 2x SSAA, 8XMSAA

SIGGRAPH2008

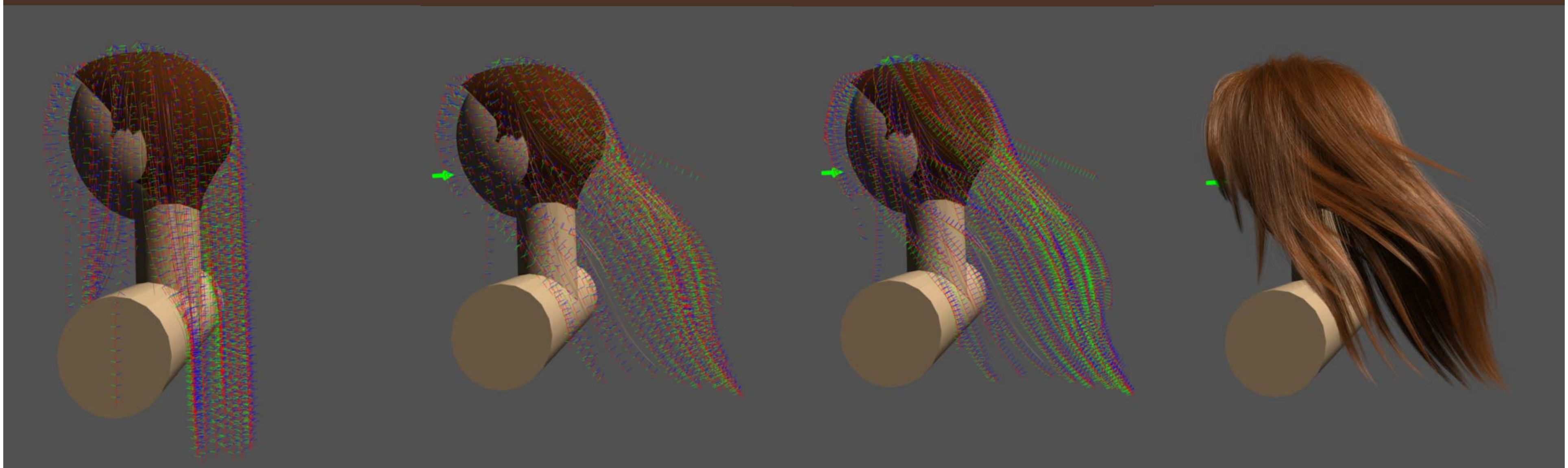
Main Contributions

- A system that runs entirely on the GPU
  - Simulation on GPU
  - Tessellation and interpolation on GPU
- Robust inter hair forces
- Detection and avoidance of interpolant hair collisions
- D3D11 tessellation based implementation

Previous work

- A number of people have done and are doing impressive things in realistic real time hair:
  - Dual Scattering Approximation for Fast Multiple Scattering in Hair. Zinke and Yuksel
  - A practical self-shadowing algorithm for interactive hair animation. Bertails et al.
  - Algorithms for Hardware Accelerated Hair Rendering. Tae-Yong Kim
  - Real-Time Approximate Sorting for Self Shadowing and Transparency in Hair Rendering. Sintron and Assarsson
  - Deep Opacity Maps. Yuksel and Keyser

Process



Import Guide Hair	Simulate Guide Hair	Tessellate and Interpolate Guide Hair	Render Final Hair

Simulation

SIGGRAPH2008

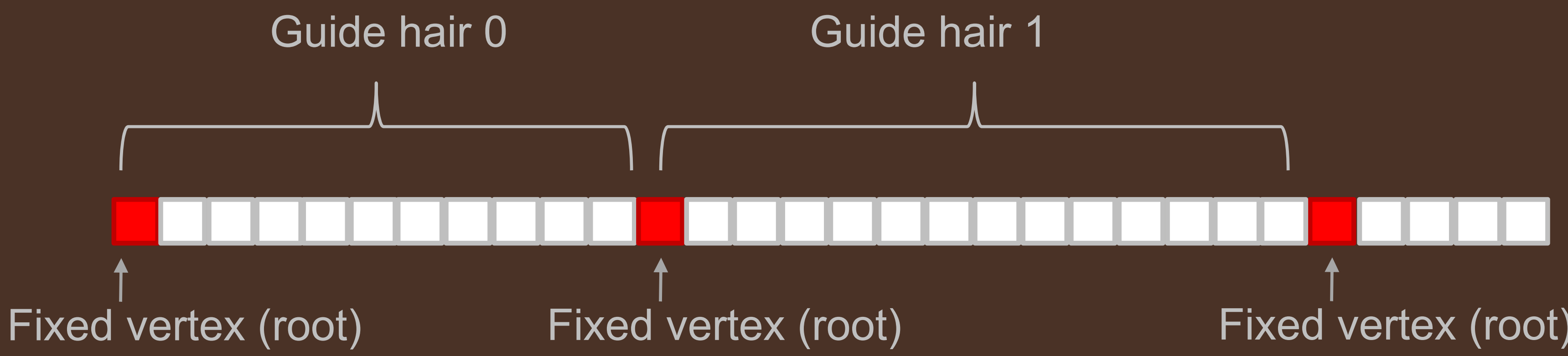
NVIDIA

Simulation

- Hair simulated as a particle constraint system
  - Hair vertices are simulated as particles
  - Links between hair vertices are treated as Distance constraints
    - these constraints maintain hair length
  - Angular forces at each hair vertex maintain hair shape
    - We have 2D angular forces (ignoring the twist dimension)
  - Collision constraints keep hair particles outside obstacles

Representation

- All the guide hair vertices are appended together to form one Vertex Buffer (VB)

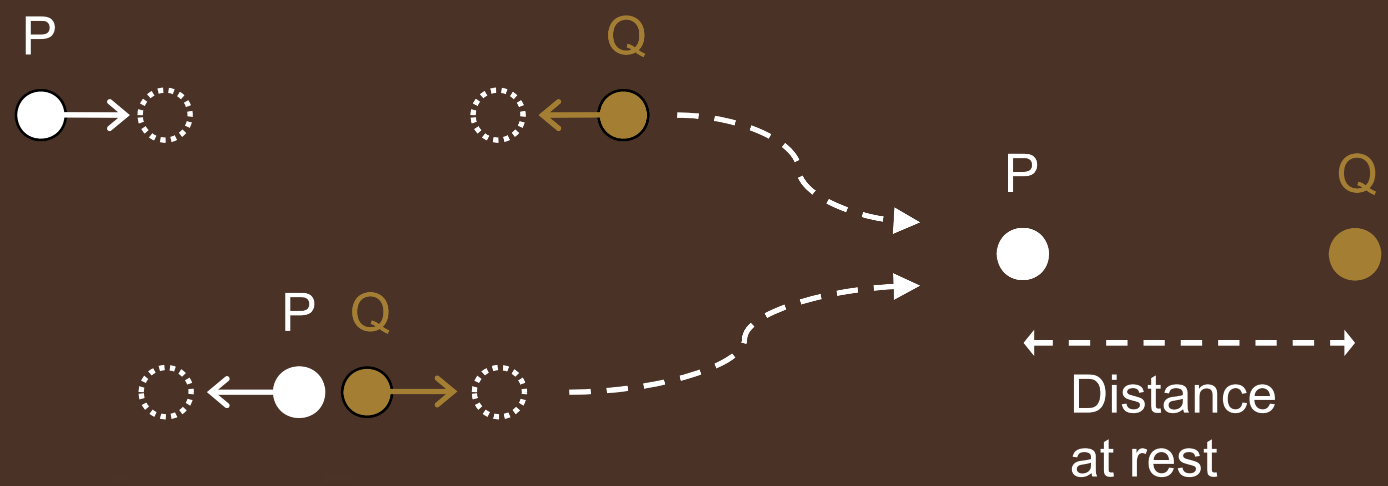


Dynamics on the GPU

- Simulation is done using the Vertex Shader and Stream Output
  - write directly from the Vertex Shader to another Vertex Buffer, skipping rasterization
- Ping-Pong between two guide hair VBs
  - Bind VB1, run a vertex shader on it, Stream Out the vertices to VB2
  - Bind VB2, run next vertex shader on it, Stream Out to VB2
  - Continue

Example: Distance Constraints

- A distance constraint DC(P, Q) between two particles P and Q is enforced by moving them away or towards each other:

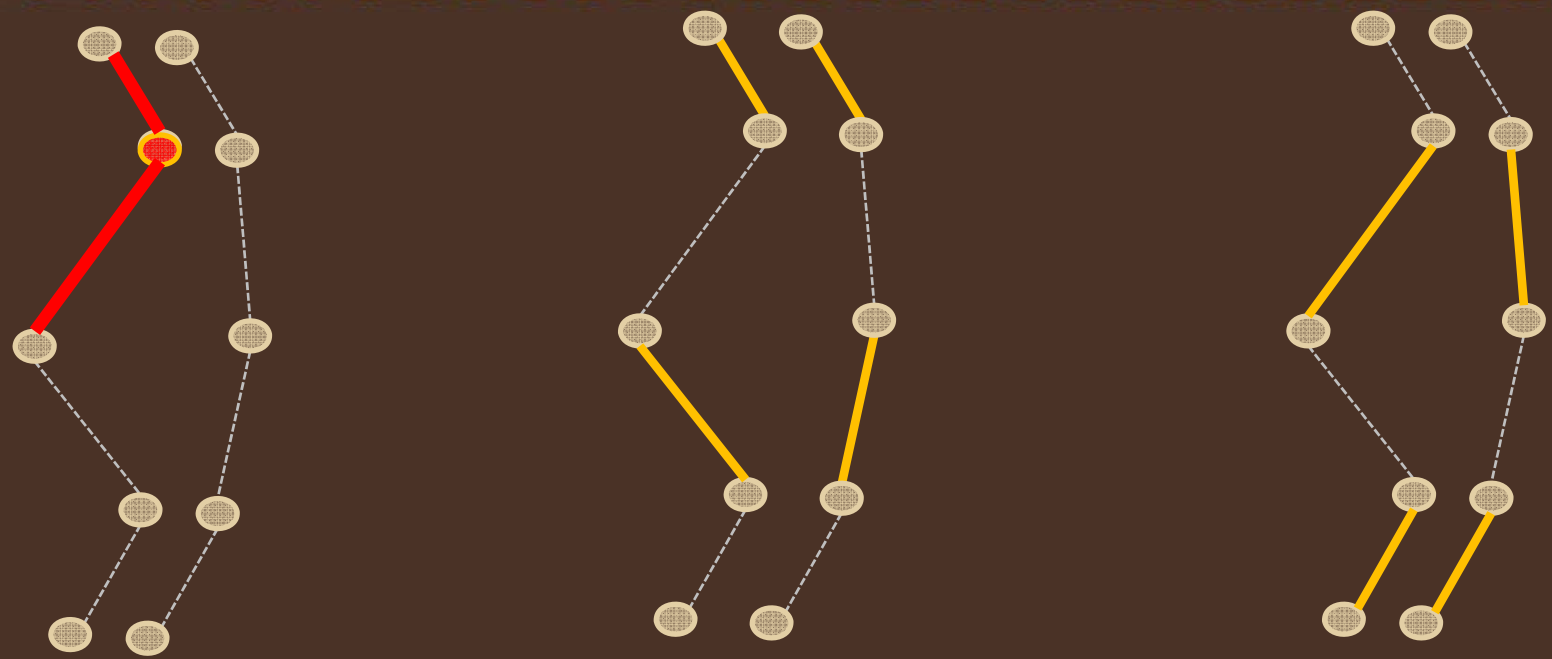


Example: Distance Constraints

- To satisfy a distance constraint we need to move the positions of two vertices

- Use Geometry Shader
  – Input two vertices
  – Output the modified positions of these vertices after satisfying their distance constraint

Example: Distance Constraints



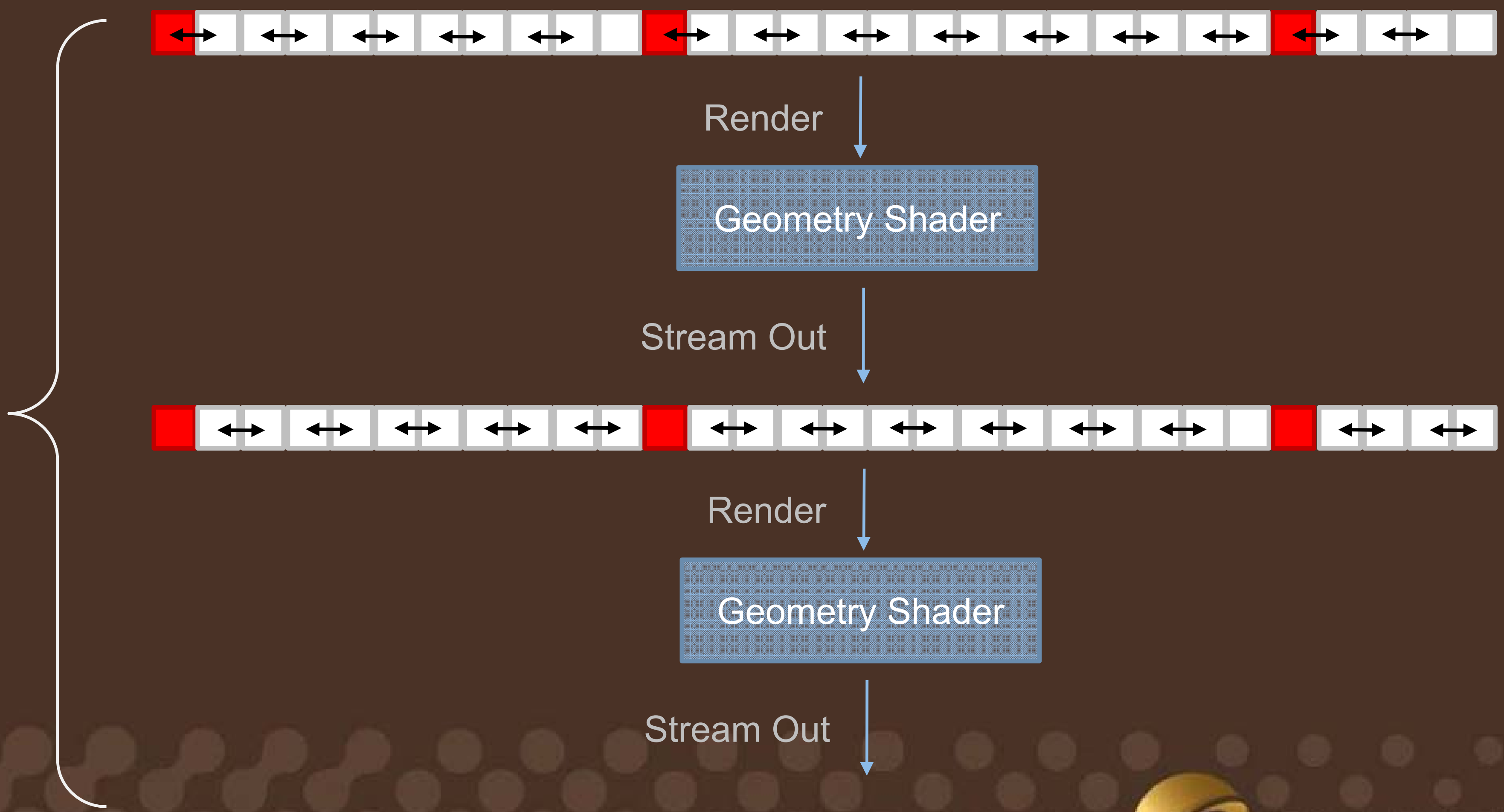
Since particles are subject to multiple constraints we cannot satisfy them all in parallel

First batch of constraints

Second batch of constraints

Iterate

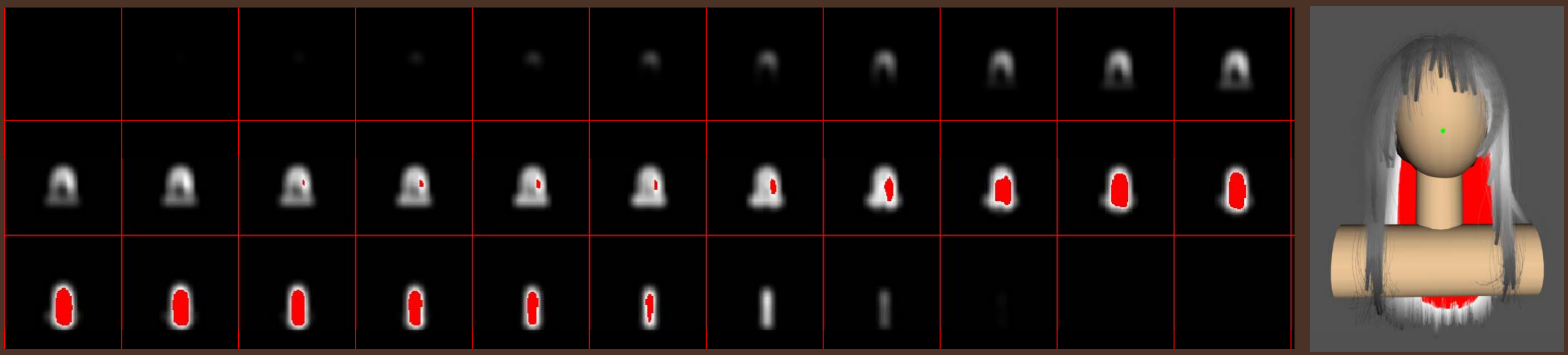
Example: Distance Constraints



Example: Inter-hair collisions

- Inter-hair collisions dealt with in grid based framework
- Hair strands and obstacles (like head/body) voxelized into a low res grid
- Hair vertices pushed out of high density areas

Example: Inter-hair collisions



- Force is applied in the direction of the negative gradient of the density
  - – blur the voxelized density, then for each vertex falling in a high density area find the gradient of the density field at that point
- This approach tries to achieve volume preserving quality of inter-hair collisions

Inter-Hair Collision Forces

Visualizing density

Before

After

Final Rendering

Before

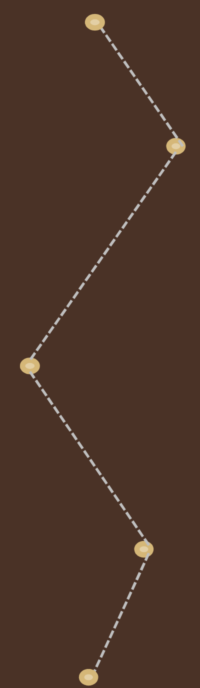
After

Wind Forces

- Wind forces simulated using semi-lagrangian fluid simulation on a coarse grid
- Voxelized hair and mesh also added to grid as obstacles to wind

Tessellation and Interpolation

Tessellation

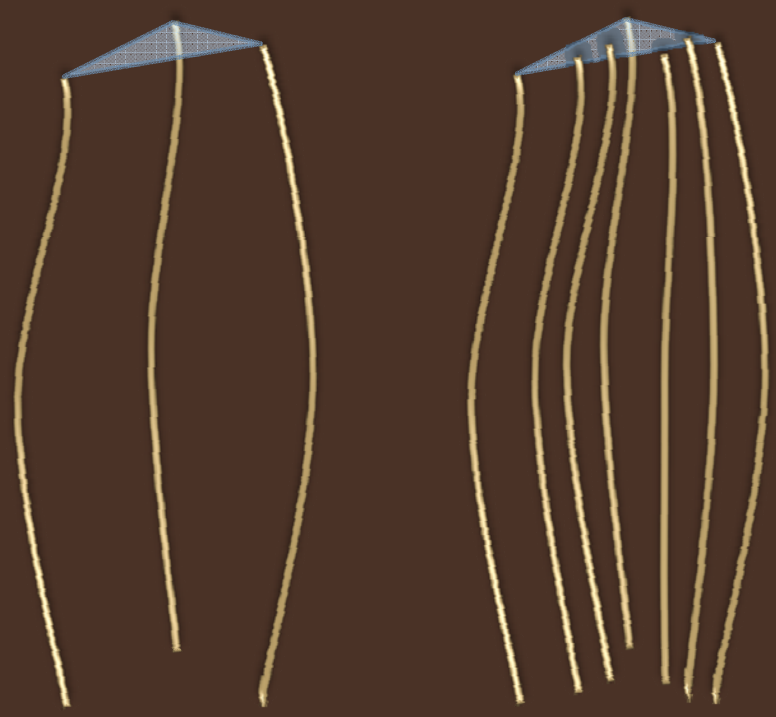


Simulated Vertices

Smoothly Tessellated Hair

SIGGRAPH2008

Interpolation

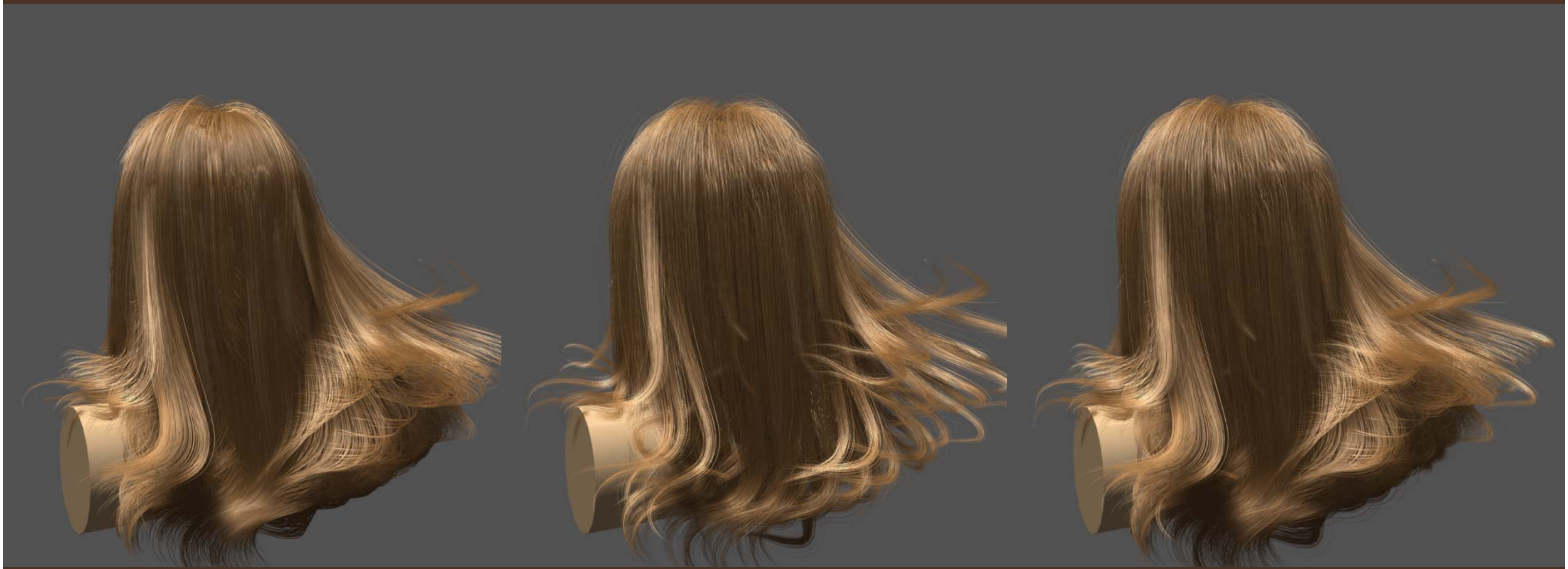


Multi Strand Interpolation

Clump Based Interpolation

SIGGRAPH2008

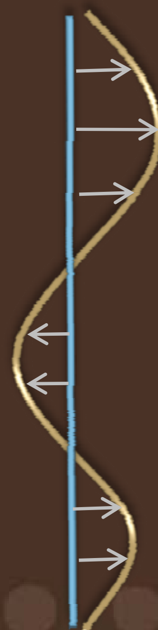
Interpolation



Multi strand Interpolation | Clump Based Interpolation | Combination

Curly Hair

- Encode additional curl offsets into constant buffers
- These offsets are added to the clump offsets
- Can either be created procedurally, or artists can create example curls and the offsets can be derived from those



Modulate density and thickness across scalp

- Green: Local density of hair
  - For e.g. Clump based hair has higher density near the front of the scalp
- Red: Local thickness of hair
  - For e.g. Clump based hair has less thickness of hair near the front of the scalp

Multi Strand

Clump Based

Process

- Create a tessellated dummy hair and render it N times, where N is the number of final hairs
- In the VS, load from Buffers storing simulated strand attributes
  - Constant attributes like strand texcoords, length, width etc
  - Variable attributes like vertex positions, coordinate frames etc

Process

- Stream out the data after each stage to minimize re-computation
  – Tessellate the simulated strands and Stream out
  – Interpolate the tessellated strands and Stream out
  – Render final hair for shading to shadow map
  – Render final hair for rendering
- Each stage uses data computed and streamed out from previous stage

Additional Optimization details

- To get a nicer look you use Alpha to Coverage, however
  - using it disables earlyZ if you are also writing and reading depth
  - To get earlyZ do a depth pre pass before, and during the final rendering just test depth without writing it
- Don't use the GS for creating hair strands
  - Can use the GS for expanding the lines to triangles but performance gain depends on pipeline load

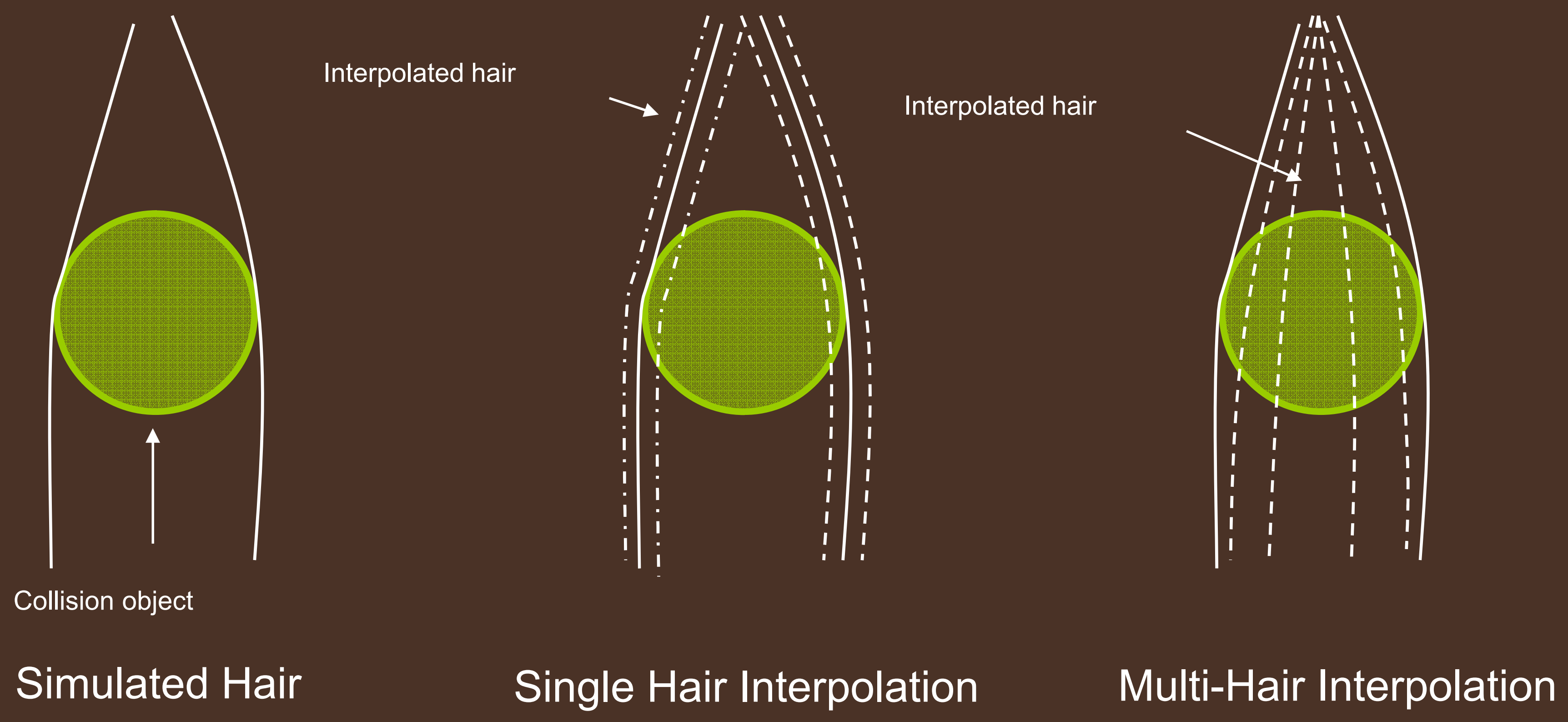
Avoiding interpolated hair collisions

- Interpolating between multiple guide strands can lead to some hair going through collision obstacles

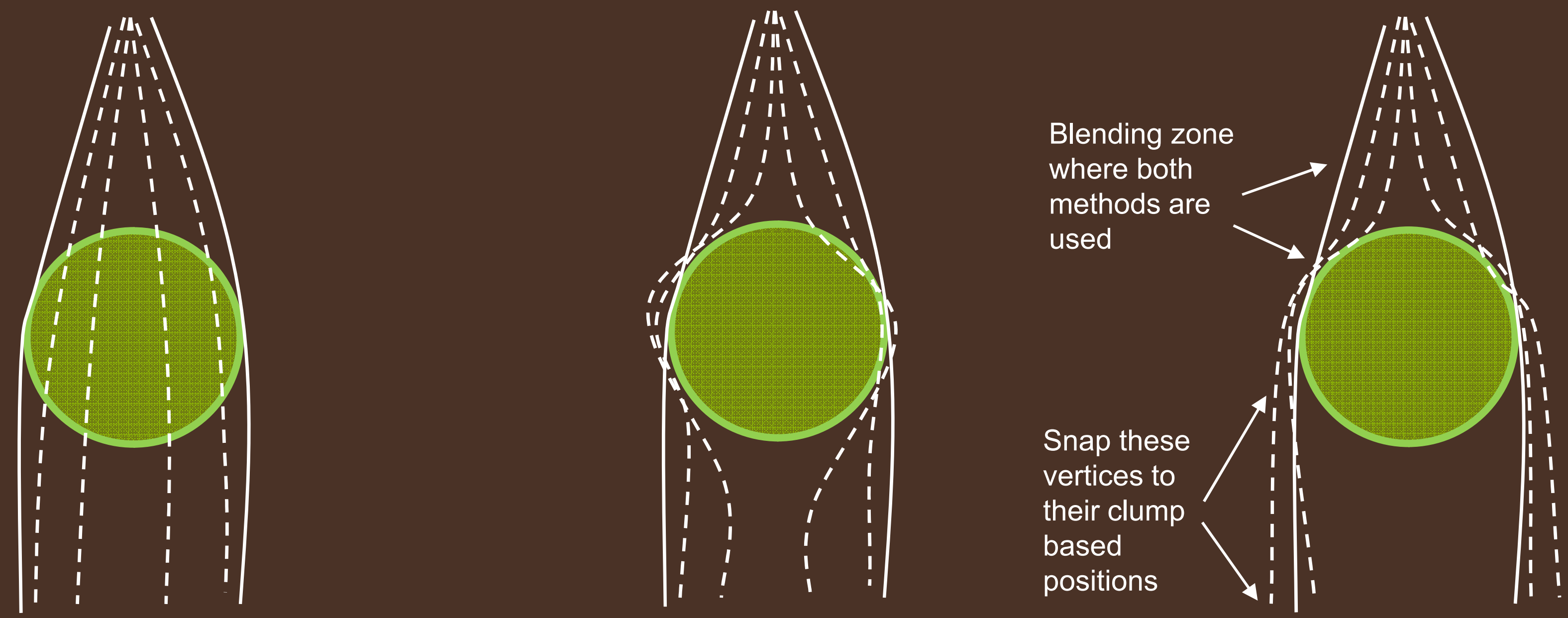
Interpolated hair intersect collision volumes

Fixing collisions without doing extra simulation

Avoiding interpolated hair collisions



Avoiding interpolant collisions

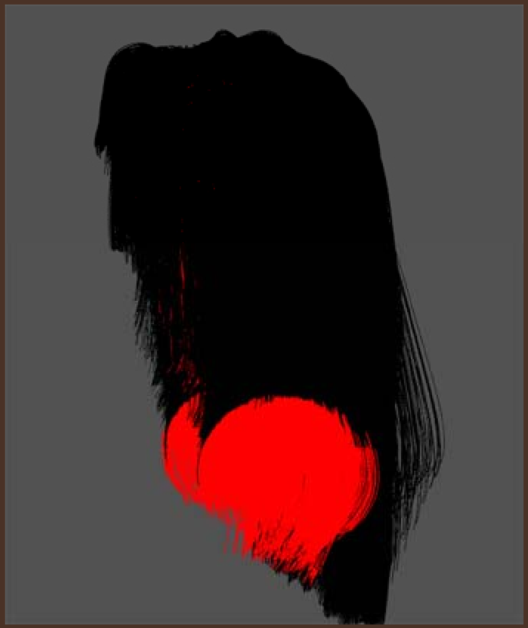


No Collision Avoidance

Modifying only penetrating vertices

Our Method

Avoiding interpolated hair collisions

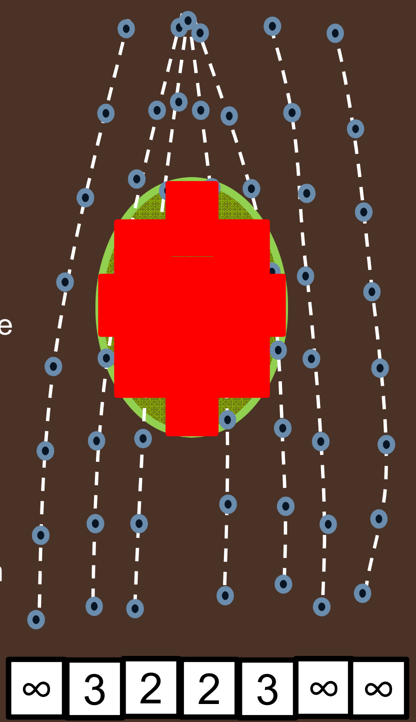

SIGGRAPH2008

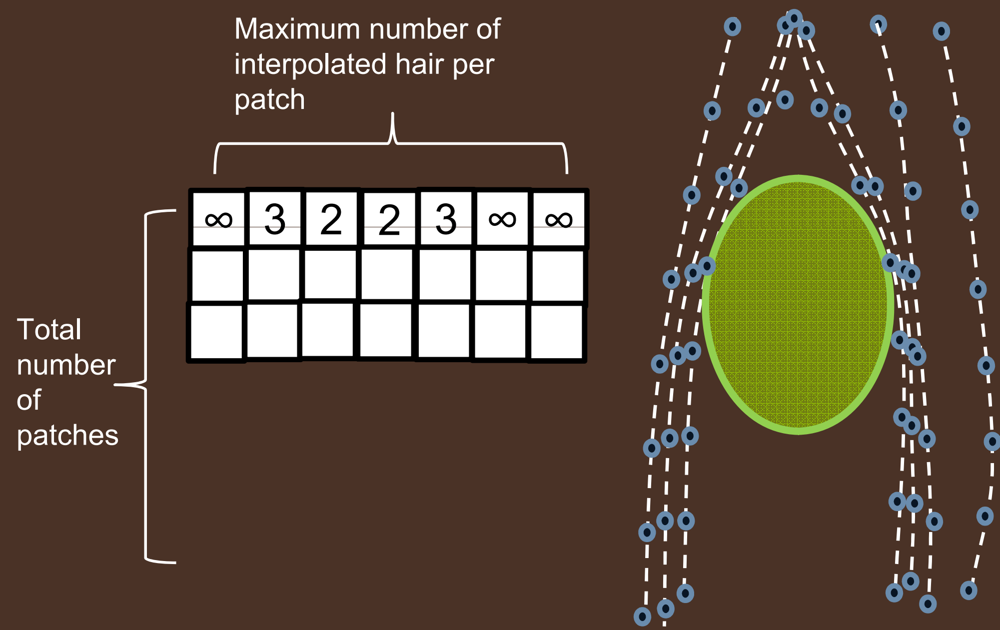
Avoiding interpolated hair collisions

Render each interpolated hair strand to one pixel

Output the vertex ID if the vertex is colliding, else a large number

Use Minimum blending

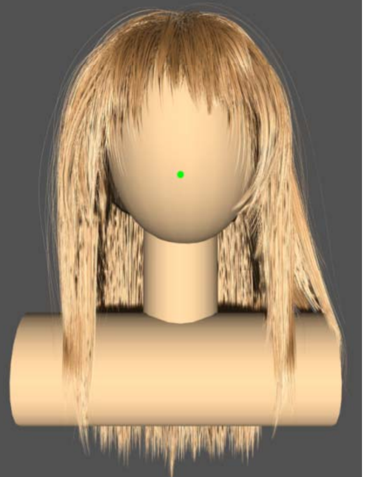




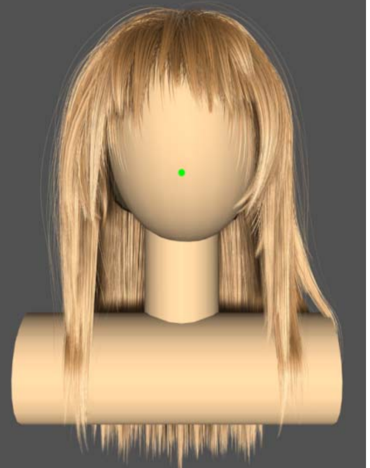
Rendering

Shadows

- Material Model: Opaque hair
- Essential Requirements
  - No flickering, smooth shadows
  - Soft Shadows
- Do PCF with multiple taps in VS
  - Help reduce temporal/spatial aliasing
  - Calculate shadows in VS and interpolate across hair length to further reduce aliasing



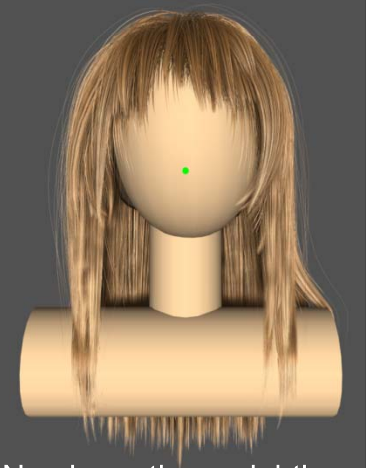
1 tap

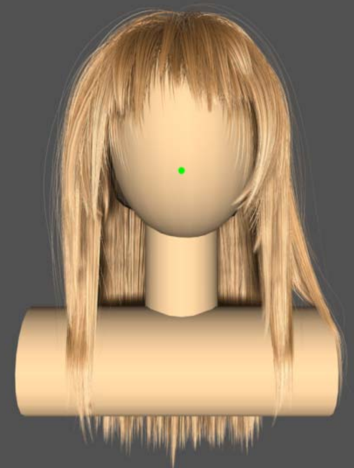


36 taps

Shadows

- Material Model: Translucent Hair
- [Yuksel and Keyser 08], [Kim and Neuman 01], [Lokovic and Veach 01]
- We do absorption weighted PCF
  - Similar to [Halen 06]
  - Weigh the PCF sample by the distance to occluder


No absorption weighting

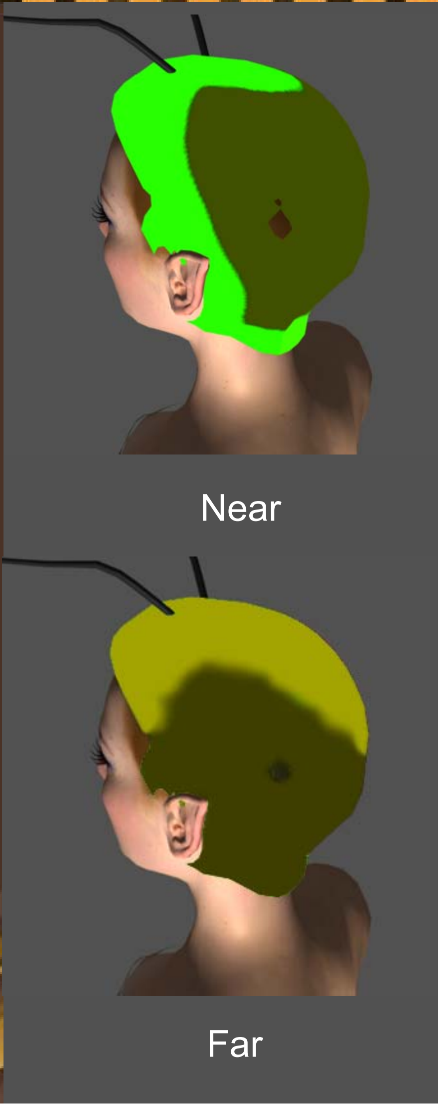

With absorption weighting

LOD

- Can use the size of a projected segment to decide on the LOD
  - Low LOD levels would use less number of lines, less segments per line, and thicker lines

LOD

- Can also use artist defined LOD
  - Artists can create different looks for different LOD of the hairstyle.
    - For example a lower LOD can have large strips just on the top of the head along the part of the hair.
  - These LODs can then be transcribed into textures
  - The Hull Shader will lerp between appropriate LOD textures to decide on the line density, and line thickness



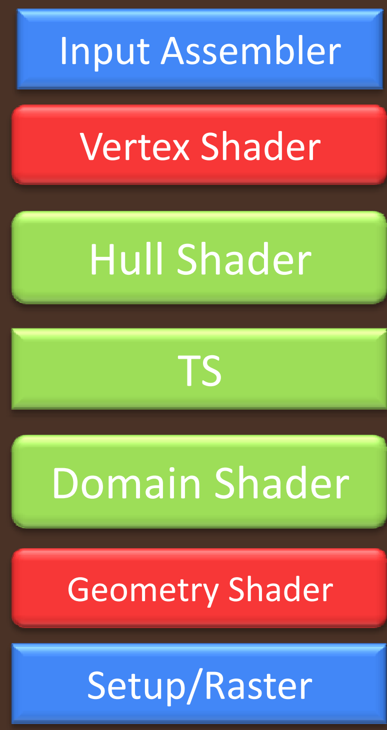
Using Dx11 Tessellation Engine

SIGGRAPH2008

NVIDIA

Tessellation Pipeline

- Direct3D11 extends Direct3D10 with support for programmable tessellation
- Two new shader stages:
  - Hull Shader (HS)
  - Domain Shader (DS)
- One fixed function stage:
  - Tessellator (TS)



ISO Lines

- Input an arbitrary patch
- For each patch output a number of lines with many segments per line
  - The number of lines output per patch and the number of segments per line are user controlled and can be different per patch
  - The positions of the vertices of the line segments are shader evaluated
- Render as lines, or expand in the GS

Interpolating and Tessellating hair

- With Tessellation engine we can create tessellated and interpolated hair on the fly
- Benefits:
  - Easy and intuitive
  - More programmable
    - Can create geometry only where needed
    - Reduce detail where not needed
  - Continuous LOD

Pipeline

Input

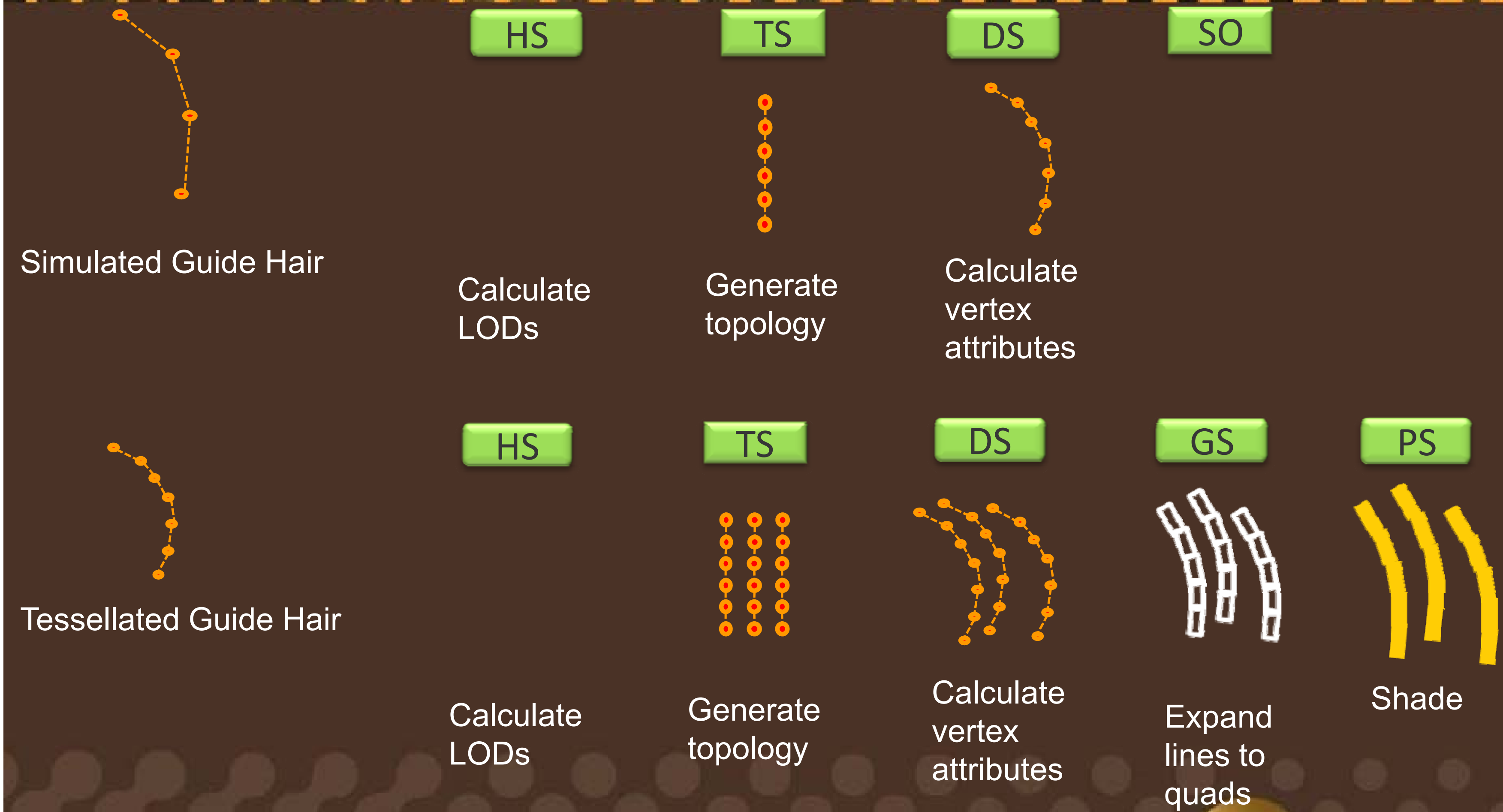
Simulated Guide Hair

HS	TS	DS	GS	PS
Calculate LODs	Generate topology	Calculate vertex attributes	Expand lines to quads	Shade

Output

Tessellated, Interpolated, Rendered Hair

Alternative Pipeline



Thank you!