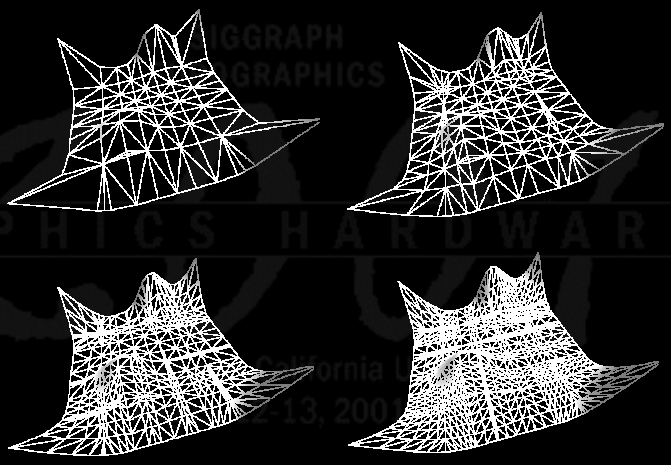
Water Tight Tessellation
Using Forward Differencing

Henry Moreton

NVIDIA

1

Motivation - LOD



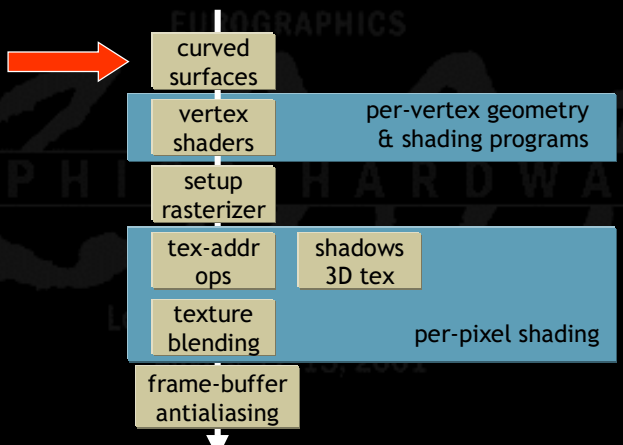
NVIDIA

2

Motivation - animation

3

Pipeline Location

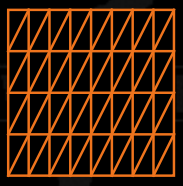


4

Historical Functionality

- Square – tensor product polynomial patches

- Grid – just specify rows and columns



- Integer – number of segments

- Share – use T&L hardware

NVIDIA

5

Requirements

- Independent tessellation factors
- Continuous level of detail
- Performance equaling triangle rendering
- Tensor product and triangular patches

NVIDIA

6

Forward Difference vs. de Casteljau

de Casteljau

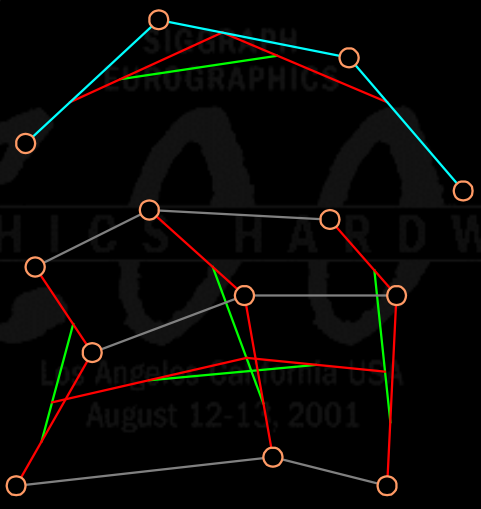
- $O(n^2)$
- precise
- arbitrary parameter values

Forward differencing

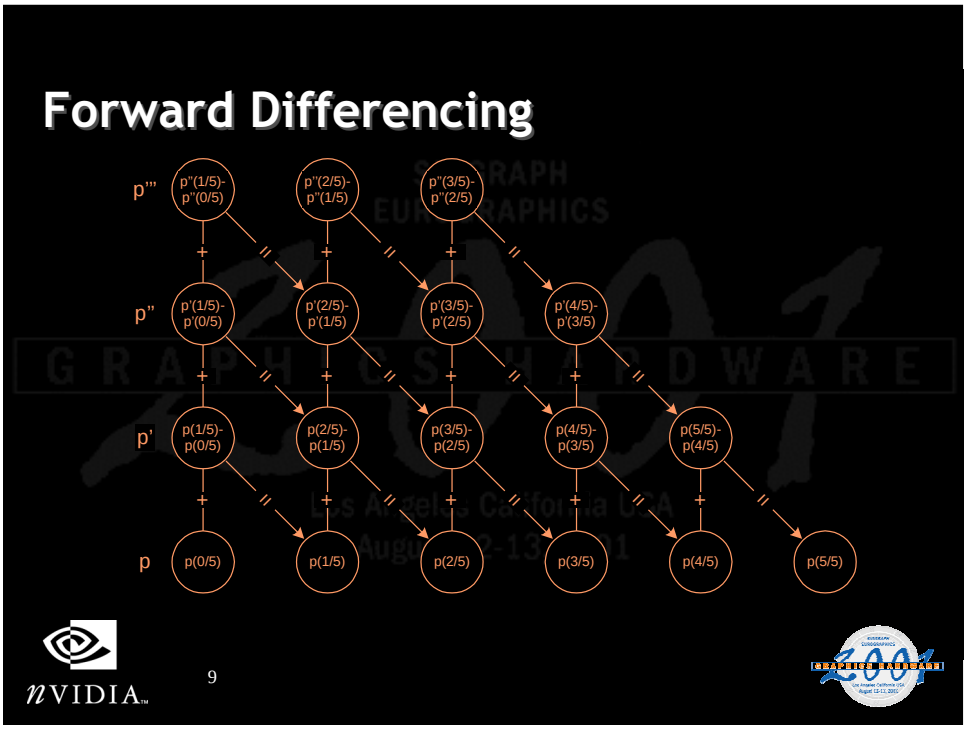
- O(n)
- suffers from roundoff
- equally spaced parameter values

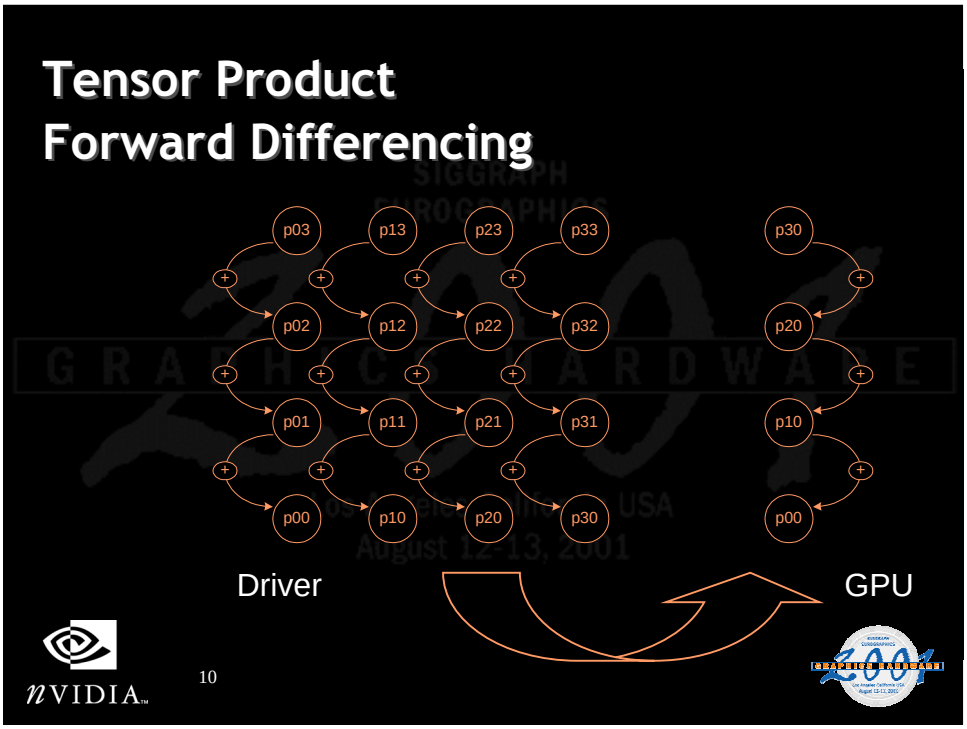
7

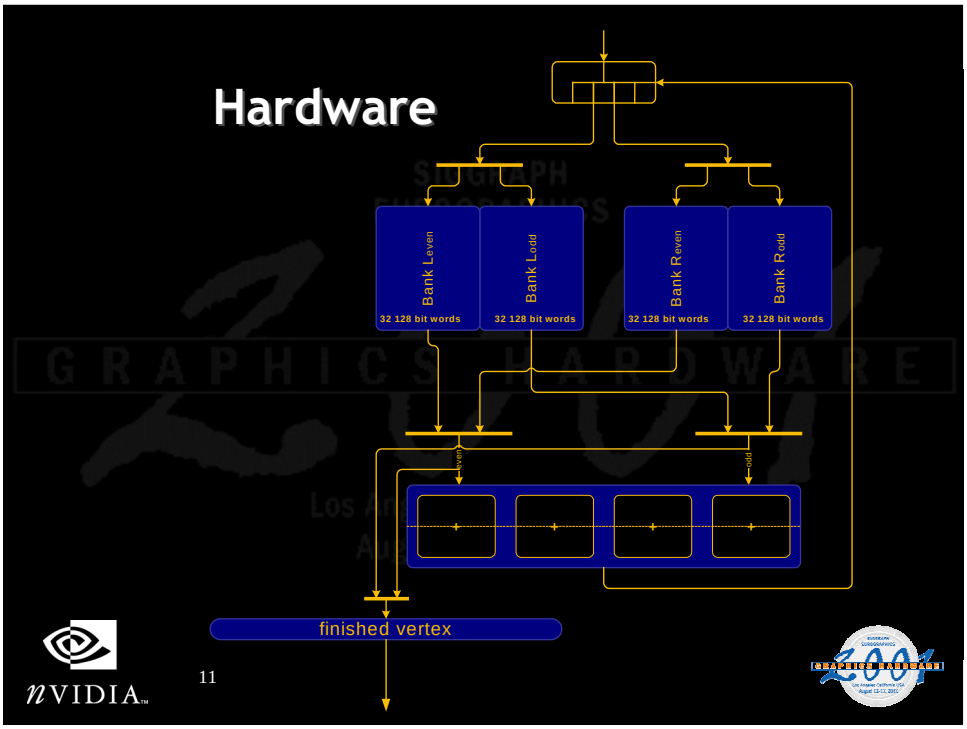
de Casteljau

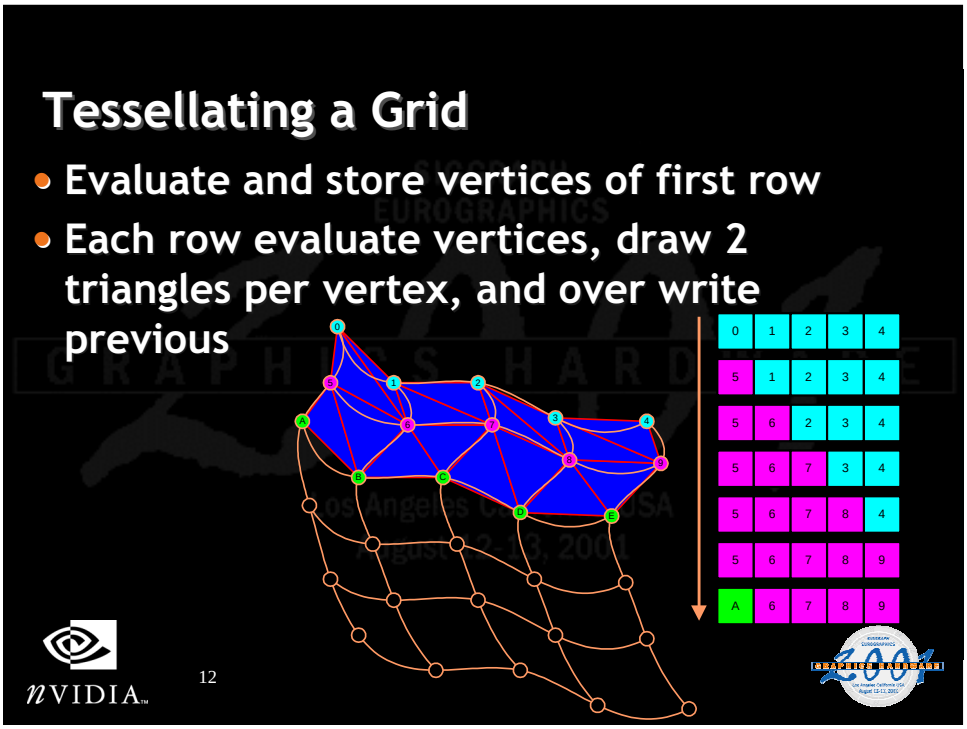


8

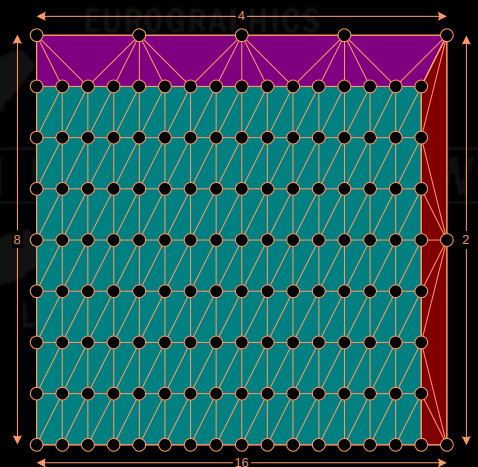

Forward Differencing
p'''
p''
p'
p
p''(1/5)-p''(0/5)
p''(2/5)-p''(1/5)
p''(3/5)-p''(2/5)
p'(1/5)-p'(0/5)
p'(2/5)-p'(1/5)
p'(3/5)-p'(2/5)
p'(4/5)-p'(3/5)
p(1/5)-p(0/5)
p(2/5)-p(1/5)
p(3/5)-p(2/5)
p(4/5)-p(3/5)
p(5/5)-p(4/5)
p(0/5)
p(1/5)
p(2/5)
p(3/5)
p(4/5)
p(5/5)
nVIDIA
9


Tensor Product
Forward Differencing
p03
p13
p23
p33
p30
p02
p12
p22
p32
p20
p01
p11
p21
p31
p10
p00
p10
p20
p30
p00
Driver
GPU
nVIDIA
10


Hardware
Bank Leven
Bank Lodd
Bank Reven
Bank Rodd
32 128 bit words
32 128 bit words
32 128 bit words
32 128 bit words
even
odd
finished vertex
NVIDIA
11

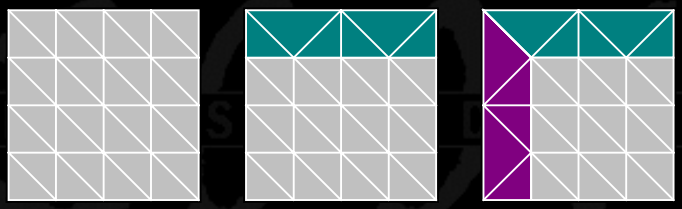

Tessellating a Grid
Evaluate and store vertices of first row
Each row evaluate vertices, draw 2 triangles per vertex, and over write previous
NVIDIA
12

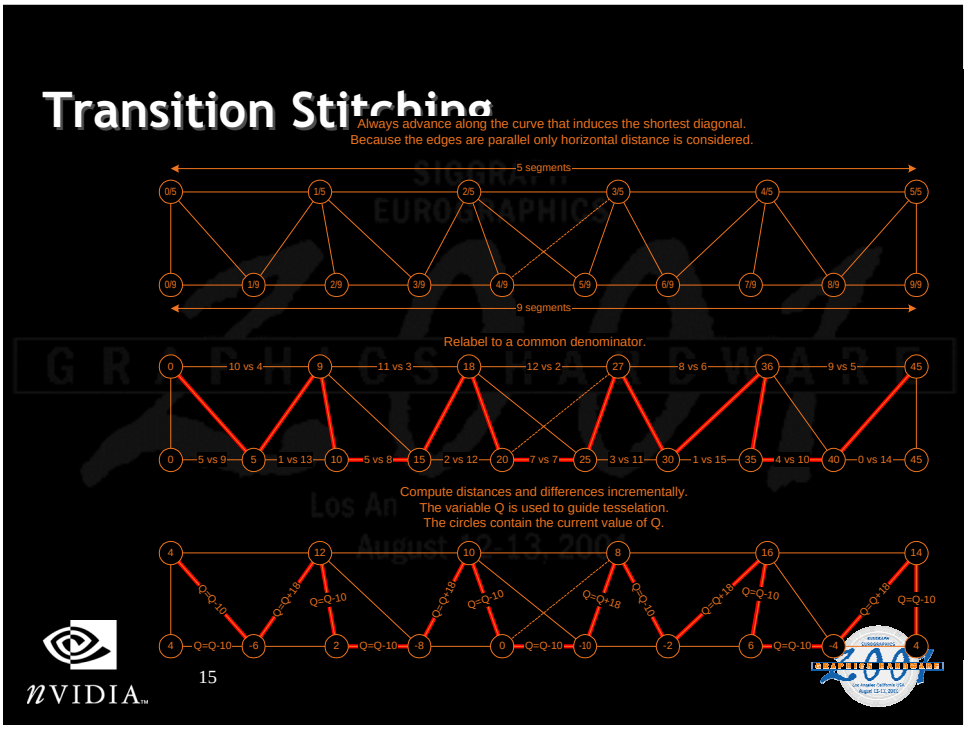
Flexible Tessellation - Transition Regions

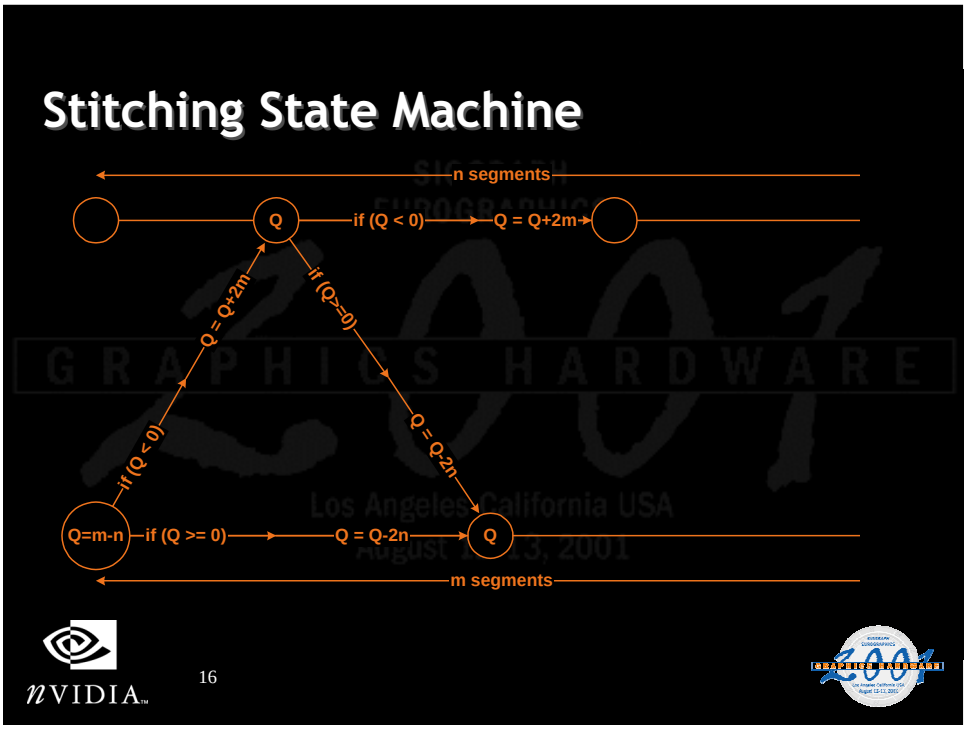


sample tessellation
with sides 16,2,4,8

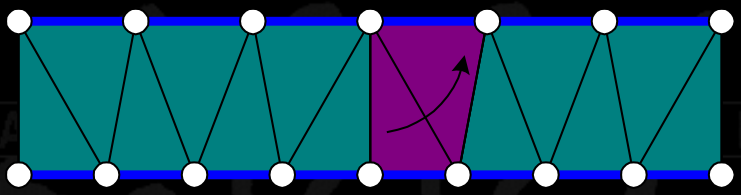
Transition Combinations




Transition Stitching
Always advance along the curve that induces the shortest diagonal.
Because the edges are parallel only horizontal distance is considered.
5 segments
9 segments
Relabel to a common denominator.
10 vs 4
11 vs 3
12 vs 2
8 vs 6
9 vs 5
5 vs 9
1 vs 13
5 vs 8
2 vs 12
7 vs 7
3 vs 11
1 vs 15
4 vs 10
0 vs 14
Compute distances and differences incrementally.
The variable Q is used to guide tesselation.
The circles contain the current value of Q.
Q=Q-10
Q=Q+18
Q=Q+18
Q=Q-10
Q=Q-10
Q=Q+18
Q=Q-10
Q=Q-10
Q=Q+18
Q=Q-10
NVIDIA
15


Stitching State Machine
n segments
Q
if (Q < 0)
Q = Q+2m
if (Q>=0)
Q = Q+2m
if (Q < 0)
Q = Q-2n
Q=m-n
if (Q >= 0)
Q = Q-2n
Q
m segments
NVIDIA
16

Strips and Fans



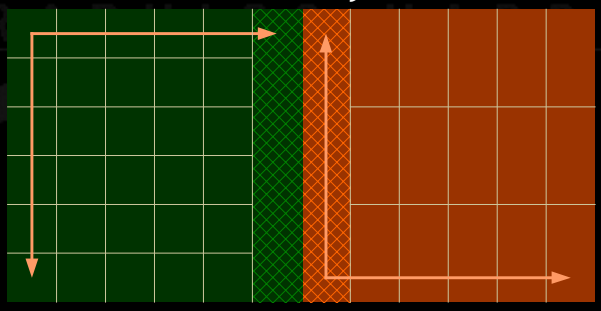
NVIDIA

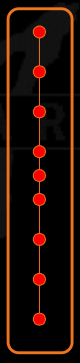
17

Fractional Tessellation

Independently drawn patches must match at shared boundaries

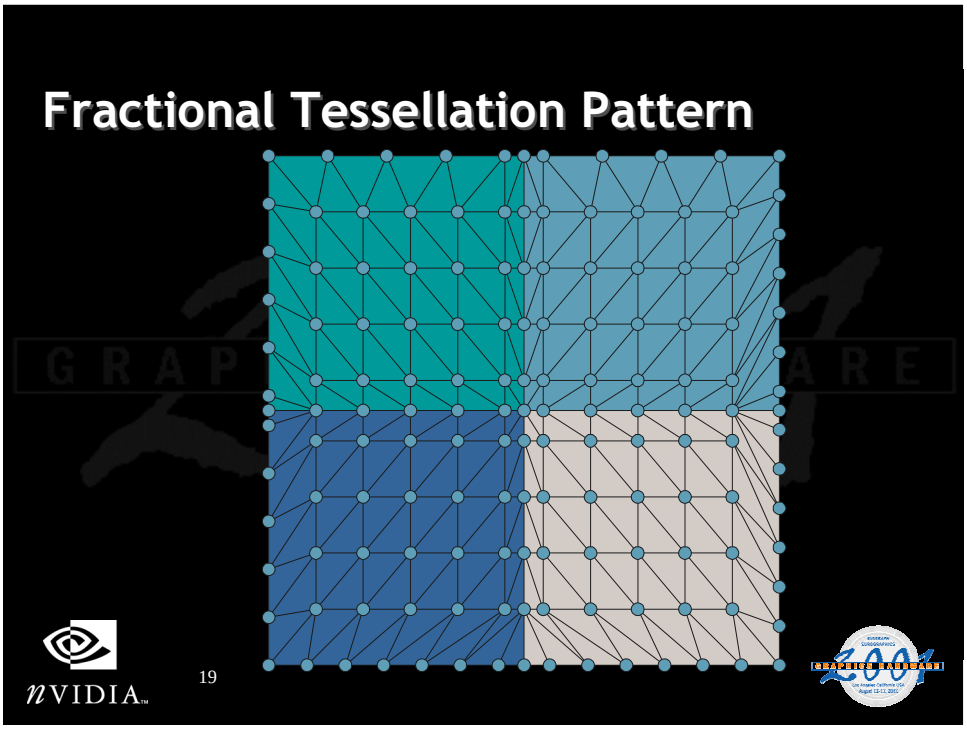
- Tessellation must be symmetric

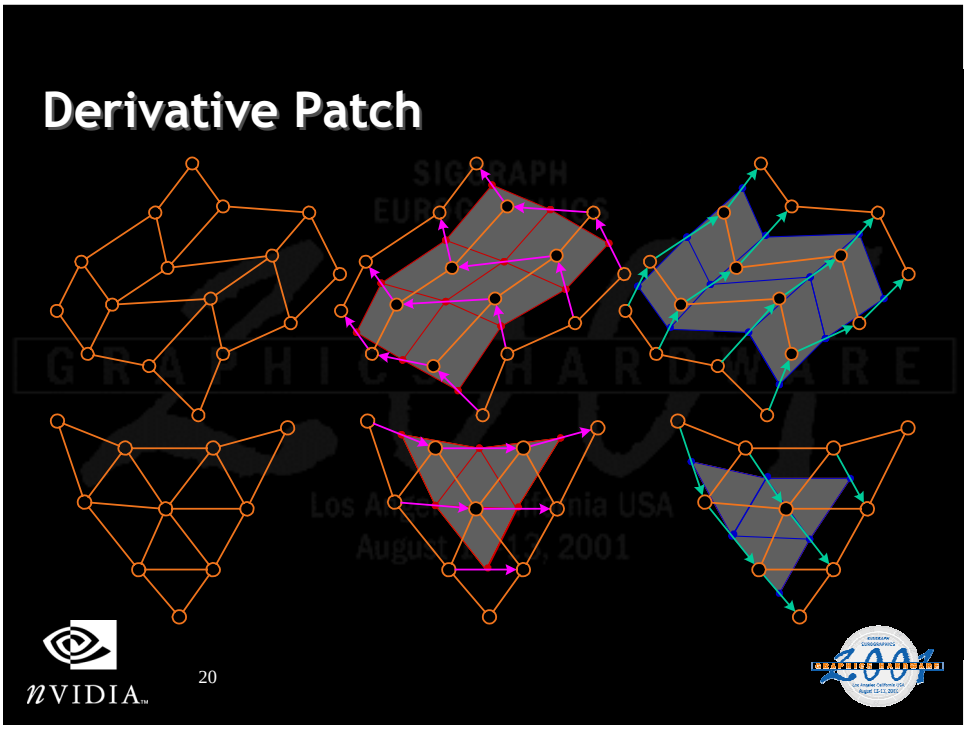




NVIDIA

18


Fractional Tessellation Pattern
19


Derivative Patch
20

The Normal Patch

$$\text{Bezier patch} = \mathbf{t} \cdot \mathbf{B} \cdot \mathbf{P} \cdot \mathbf{B}^t \cdot \mathbf{s}$$

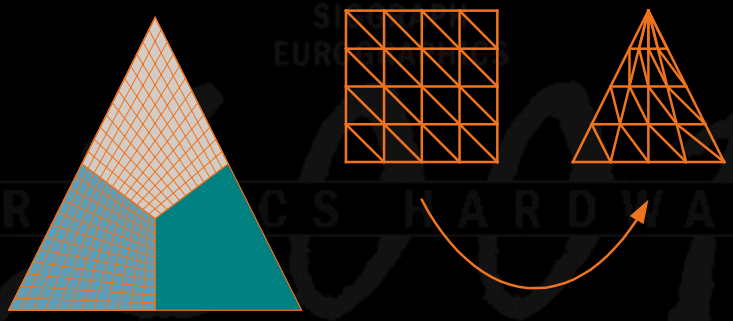
$$\text{Bezier patch}_s = \mathbf{t} \cdot \mathbf{B} \cdot \mathbf{P} \cdot \mathbf{B}^t \cdot \mathbf{s'}$$

$$\text{Bezier patch}_t = \mathbf{t'} \cdot \mathbf{B} \cdot \mathbf{P} \cdot \mathbf{B}^t \cdot \mathbf{s}$$

$$\text{Bezier patch}_n = \text{Bezier patch}_s \times \text{Bezier patch}_t$$

21

Triangular Patches



Three patches
same total degree

One patch with
degenerate vertex and
irregular tessellation
(compute normal
before reparameterization)

22

Precision Problems

- Serious round-off error
- Multiple floating point engines

$$A_{Intel} B_{Intel} \neq A_{GPU} B_{GPU}$$

- Independently drawn patches
  we can't fix problems
- A single bit error can cause a dropout or double hit

23

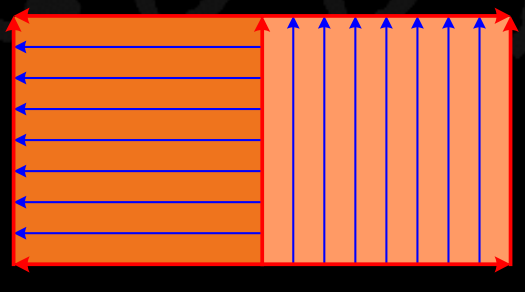
Consistency over Accuracy

- Perform computations consistently
  make the same mistake twice
- Traverse curves in the same direction
  reproduce round-off error
- Always perform calculations on the same floating point unit

24

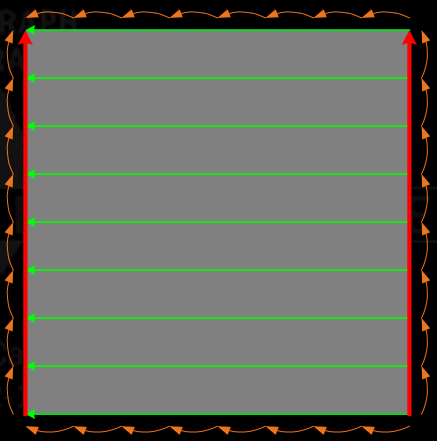
Matching Boundaries of Adjacent Patches

- Application specifies tessellation amount
- Driver selects traversal direction
  vertex sorting to determine traversal

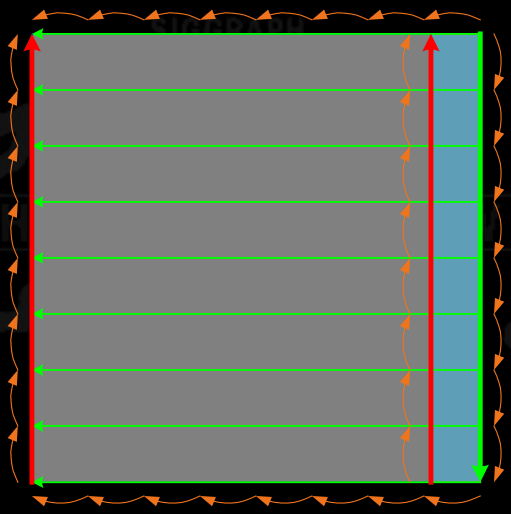


Guard Curves

Position and normal only

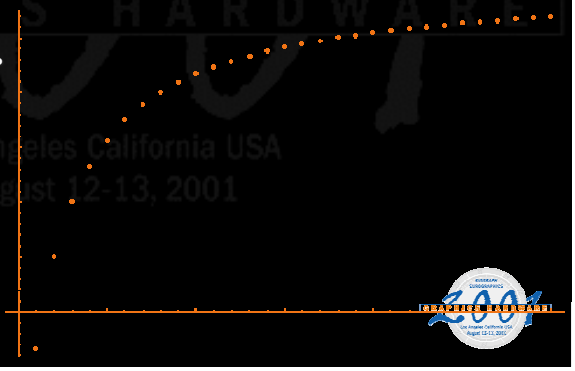


Reversed Transitions

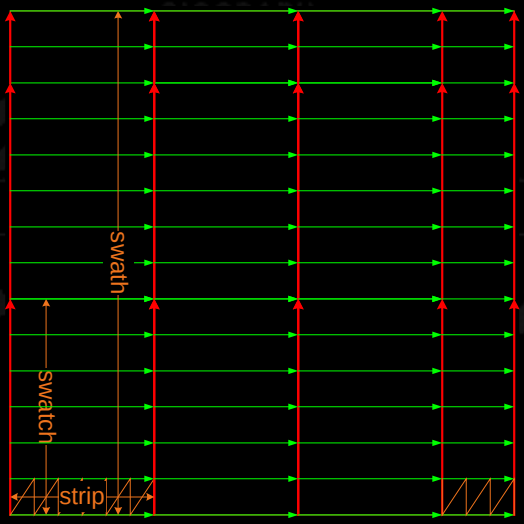


Vertex Reuse

- Tessellator maximizes vertex reuse.
- Vertex RAM stores vertices for reuse.
- Managed by HW.



Tessellations are broken up



Special Vertices

Unit Testing

Random testing

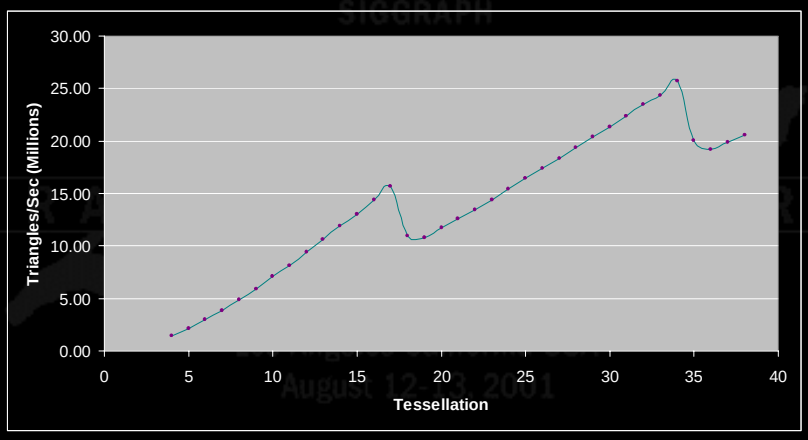
Geometric consistency checks

- Vertex count
- Triangle count
- Triangle orientation
- Degenerate triangles
- Consistent images under all combinations of patch edge reversals

NVIDIA

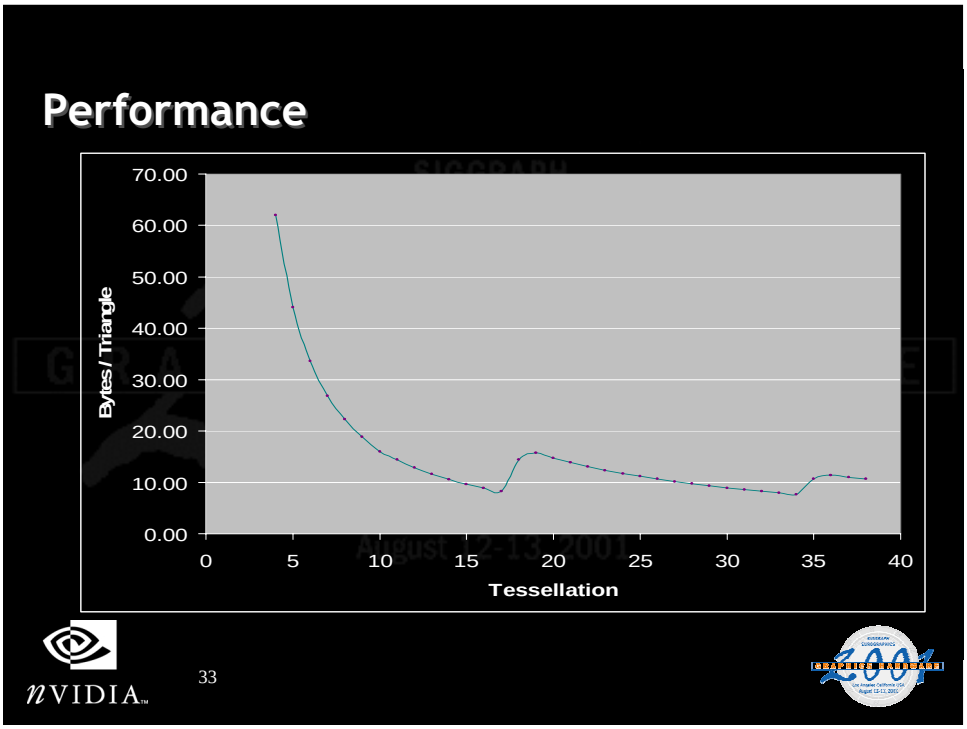
31

Performance



NVIDIA

32


Performance
70.00
60.00
50.00
40.00
30.00
20.00
10.00
0.00
Bytes / Triangle
0
5
10
15
20
25
30
35
40
Tessellation
NVIDIA
33