Efficient Bounding of Displaced Bézier Patches

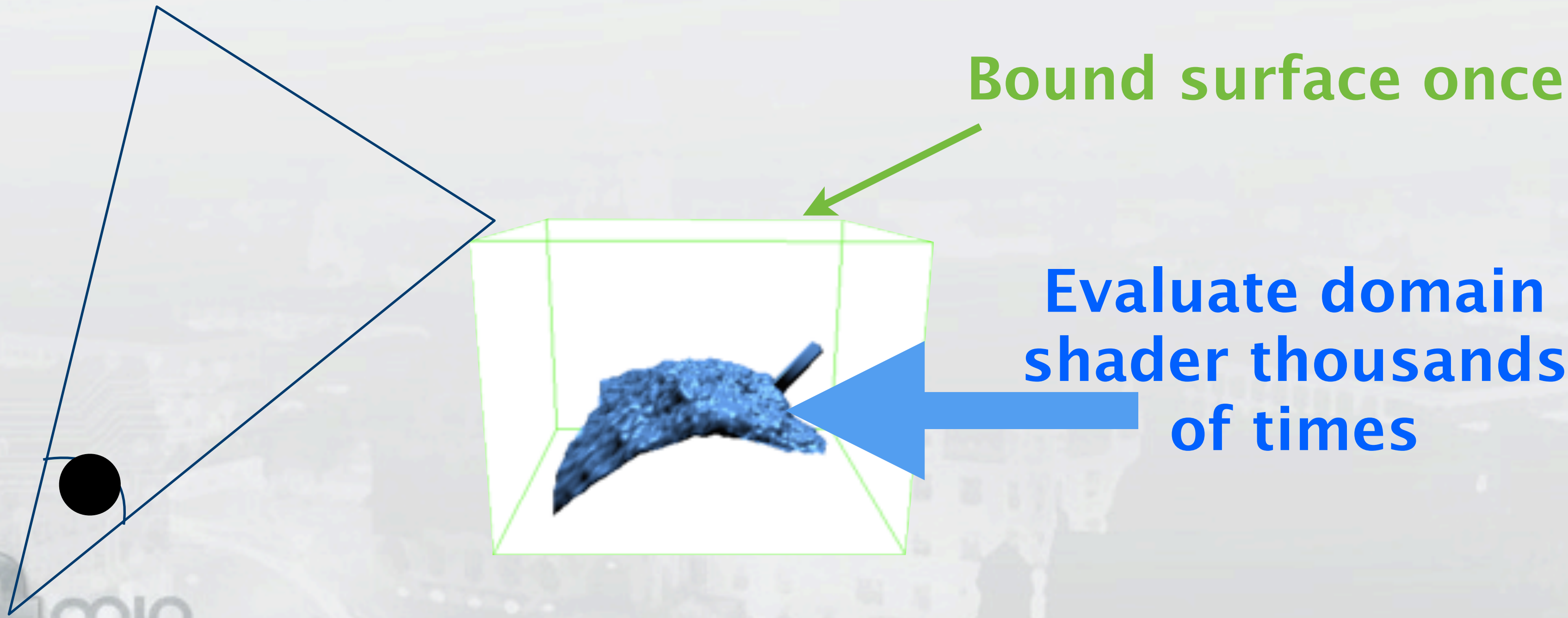
Jacob Munkberg, Jon Hasselgren,
Robert Toth, Tomas Akenine-Möller

Intel Corporation & Lund University

hpg 2010

Motivation

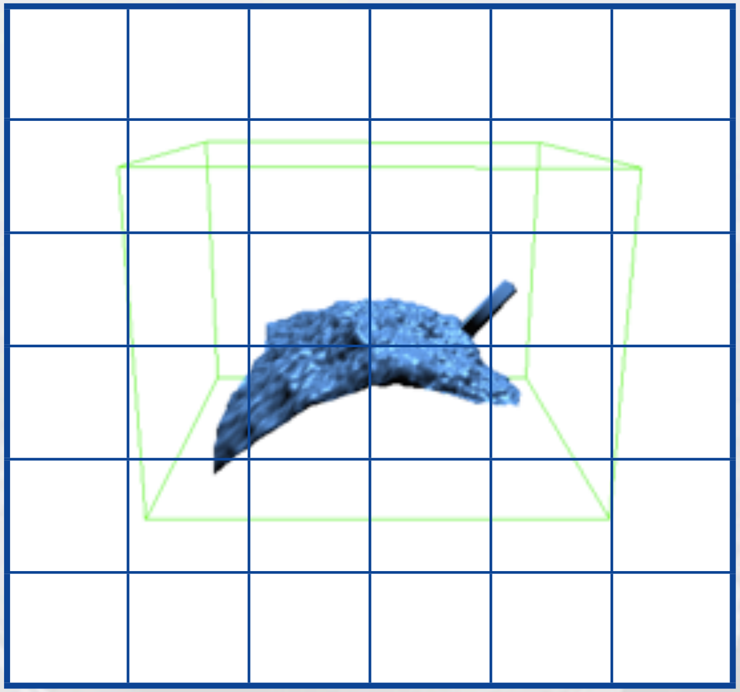
- Tessellation is increasingly important
  - **Displaced parametric surfaces** is a prime use case
  - Significant data amplification
- Efficiently compute hierarchical bounds of a patch
  - Cull as early as possible - save domain shader work
  - Bounds used for binning in rendering frameworks (PRMan)



Sunday, June 27, 2010

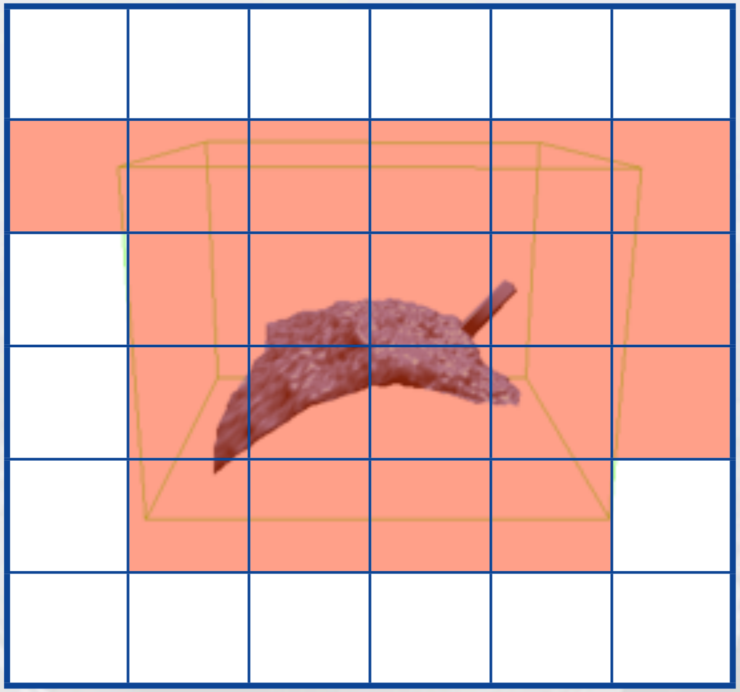
Motivation

- Tessellation is increasingly important
  - **Displaced parametric surfaces** is a prime use case
  - Significant data amplification
- Efficiently compute hierarchical bounds of a patch
  - Cull as early as possible - save domain shader work
  - Bounds used for binning in rendering frameworks (PRMan)



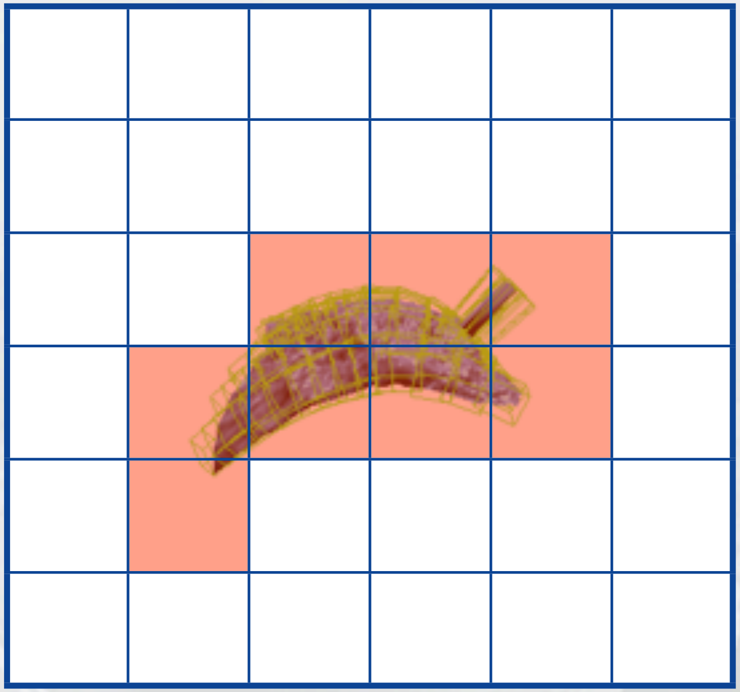
Motivation

- Tessellation is increasingly important
  - **Displaced parametric surfaces** is a prime use case
  - Significant data amplification
- Efficiently compute hierarchical bounds of a patch
  - Cull as early as possible - save domain shader work
  - Bounds used for binning in rendering frameworks (PRMan)



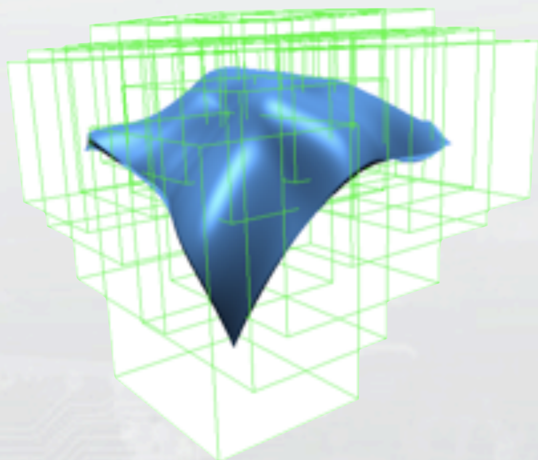
Motivation

- Tessellation is increasingly important
  - **Displaced parametric surfaces** is a prime use case
  - Significant data amplification
- Efficiently compute hierarchical bounds of a patch
  - Cull as early as possible - save domain shader work
  - Bounds used for binning in rendering frameworks (PRMan)

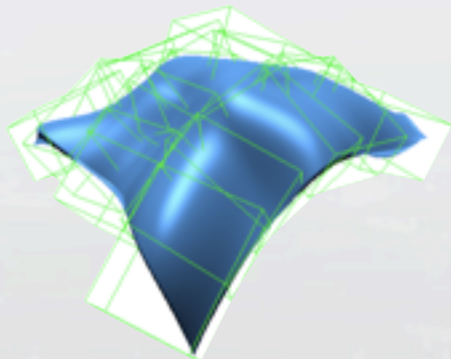


Previous Work

- Simple bounding approaches do not converge
  - For example, constant displacement bounds. c.f. Eye split problem in PRMan [Apodaca & Gritz, 2000]
- Optimize for the common case
  - General techniques, such as Pre-Tessellation Culling [Hasselgren et. al, 2009] not fine-tuned for special use case



Constant displacement bounds



Our algorithm

Optimize for common case

- Displaced Bézier surface

$$\mathbf{q}(u,v) = \mathbf{M}(\mathbf{p}(u,v) + \hat{\mathbf{n}}(u,v)t(u,v))$$

Base patch — $\mathbf{p}(u,v)$; **Normal** — $\hat{\mathbf{n}}(u,v)$; **Displacement** — $t(u,v)$

- Base Bézier patch
- Scalar displacement along the geometric normal vector
- Displacement generally from texture map
- Final surface point transformed to clip space

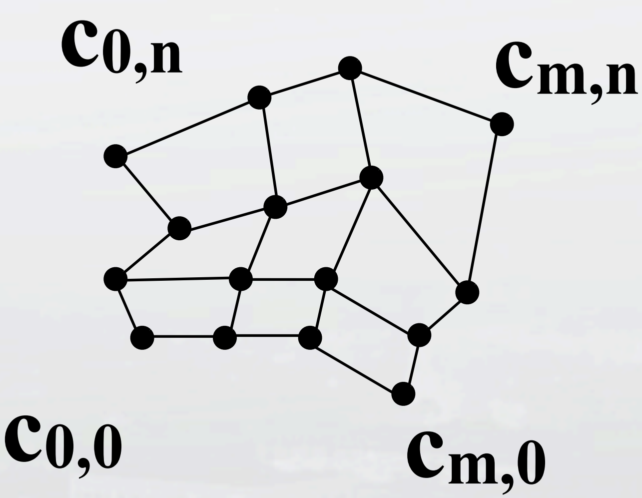
Algorithm Summary

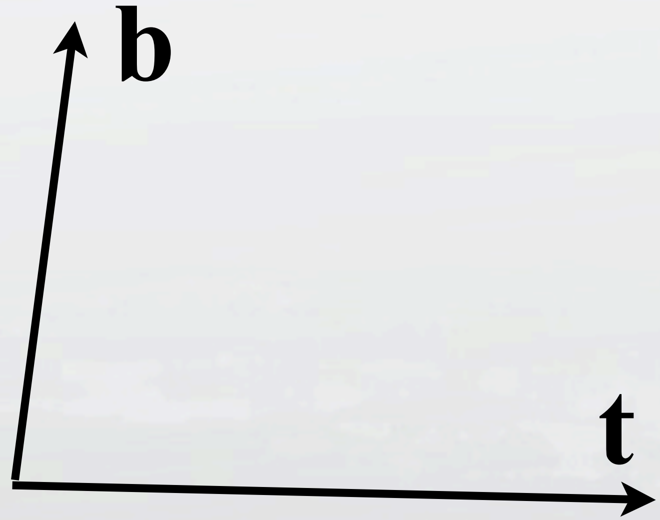
- Find OBB coordinate frame from Bézier control cage
- Bound all terms of the displaced Bézier patch
  - Base patch
  - *Normalized* surface normal
  - Displacement height over patch
- Use bounds for culling / binning

$$\mathbf{q}(u,v) = \mathbf{M}(\mathbf{p}(u,v) + \hat{\mathbf{n}}(u,v)t(u,v))$$

OBB Coordinate Frame

- Simple heuristic
  - Compute approximate patch tangent/binormal
  - Approximate patch normal $\mathbf{n} = \mathbf{t} \times \mathbf{b}$
  - Create orthonormal coordinate frame

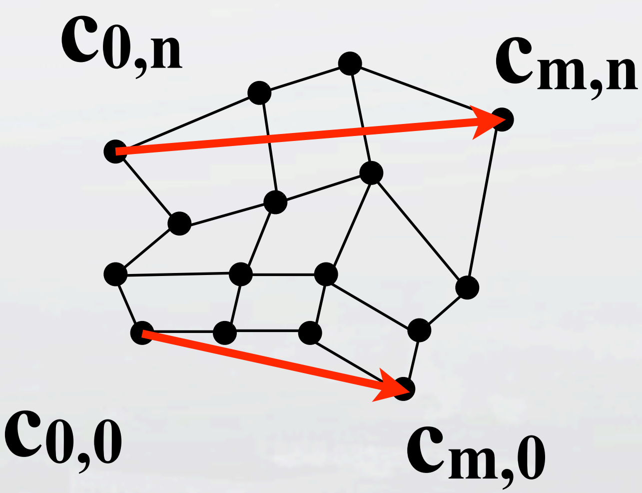


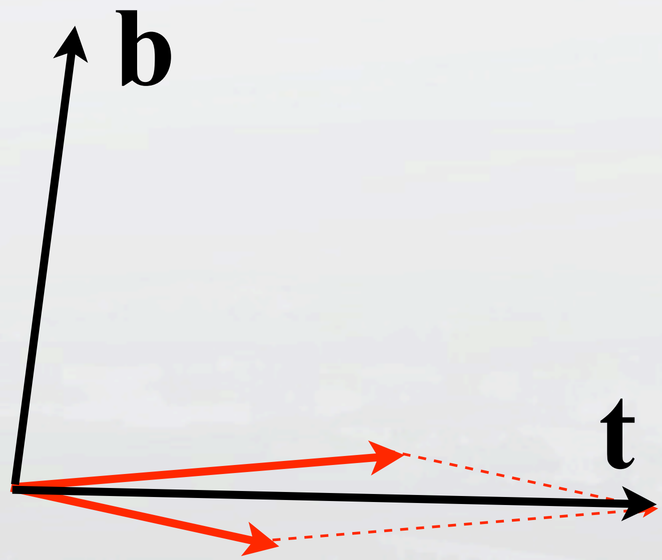


- Reuse coordinate frame for all steps in bounding algorithm

OBB Coordinate Frame

- Simple heuristic
  - Compute approximate patch tangent/binormal
  - Approximate patch normal $\mathbf{n} = \mathbf{t} \times \mathbf{b}$
  - Create orthonormal coordinate frame

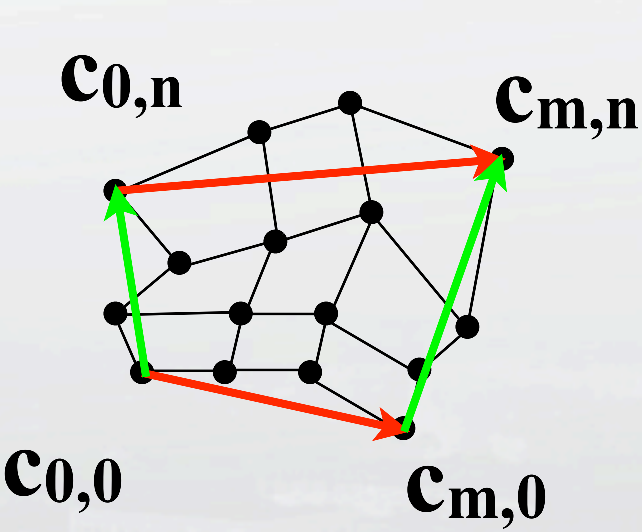


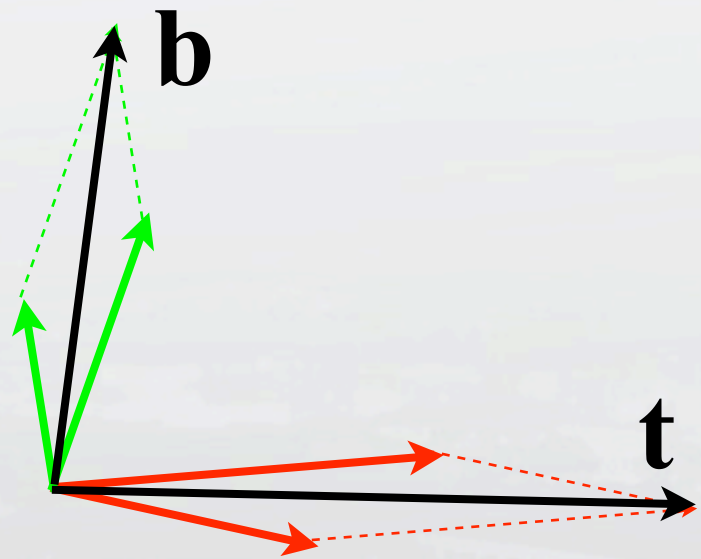


- Reuse coordinate frame for all steps in bounding algorithm

OBB Coordinate Frame

- Simple heuristic
  - Compute approximate patch tangent/binormal
  - Approximate patch normal $\mathbf{n} = \mathbf{t} \times \mathbf{b}$
  - Create orthonormal coordinate frame



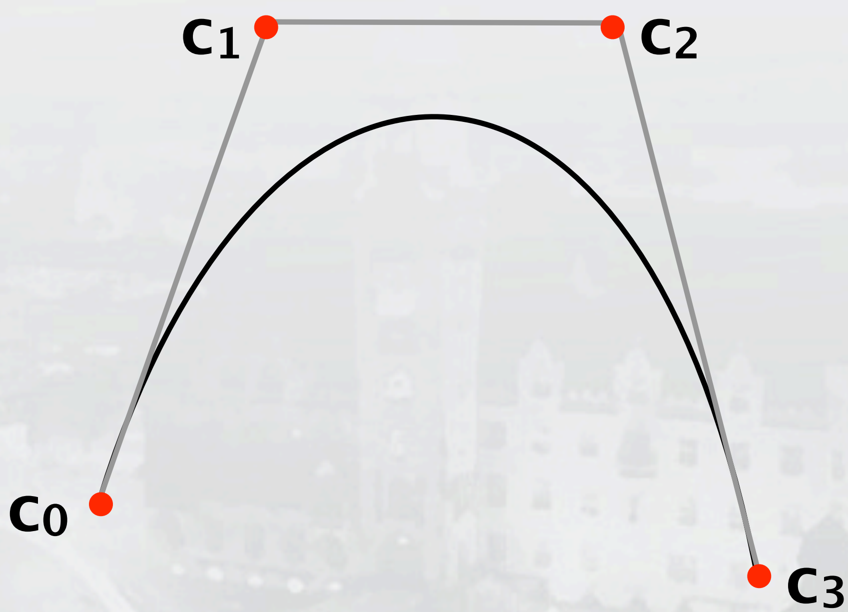


- Reuse coordinate frame for all steps in bounding algorithm

Bound Base Patch

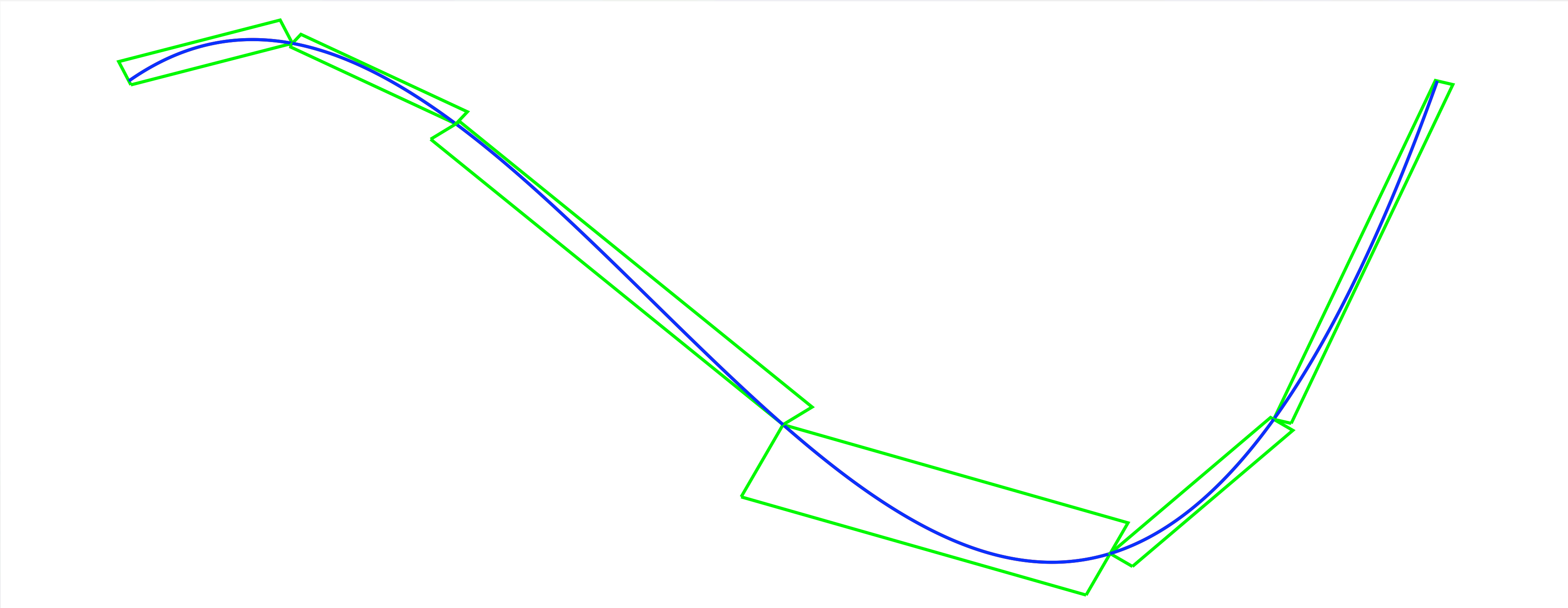
- Bézier Patches have convex hull property
  - Surface bounded by its control points, $\mathbf{c}_{i,j}$

$$\mathbf{p}^{m,n}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} \mathbf{c}_{i,j} B_i^m(u) B_j^n(v),$$



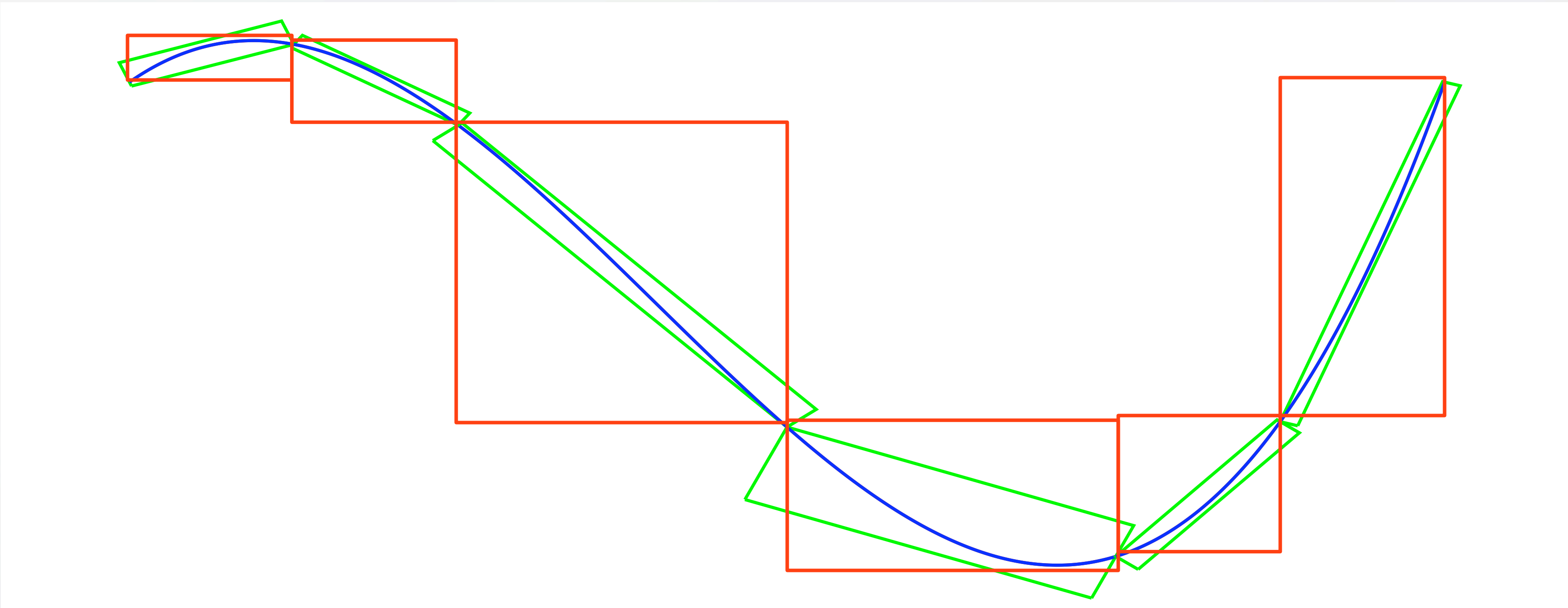
Bound Base Patch

- Transform control points to OBB coordinate frame



Bound Base Patch

- Transform control points to OBB coordinate frame



Surface Normal Bounds

- Normal vector patch is cross product of tangent vector patches

$$\mathbf{n}(u,v) = \frac{\partial \mathbf{p}}{\partial u}(u,v) \times \frac{\partial \mathbf{p}}{\partial v}(u,v)$$

$$= \sum_{i=0}^{m-1} \sum_{j=0}^{n} \mathbf{a}_{i,j} B_i^{m-1}(u) B_j^n(v)$$

$$\times \sum_{k=0}^{m} \sum_{l=0}^{n-1} \mathbf{b}_{k,l} B_k^m(u) B_l^{n-1}(v)$$

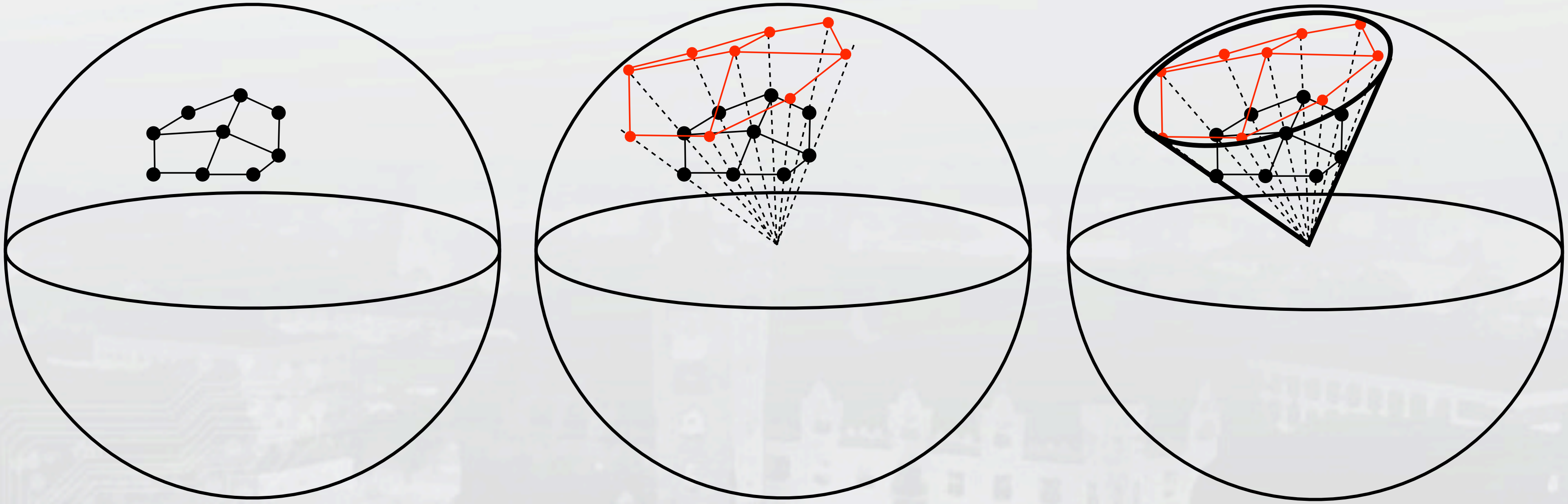
  - Normal vector patch is also a Bézier patch of degree n + m -1 [Yamaguchi, 1997]

$$\mathbf{v}_{p,q} = \sum_{\substack{i+k=p \\ j+l=q}} \mathbf{a}_{i,j} \times \mathbf{b}_{k,l} \frac{\binom{m-1}{i}\binom{m}{k}\binom{n}{j}\binom{n-1}{l}}{\binom{m+n-1}{i+k}\binom{m+n-1}{j+l}}.$$

hpg 2010

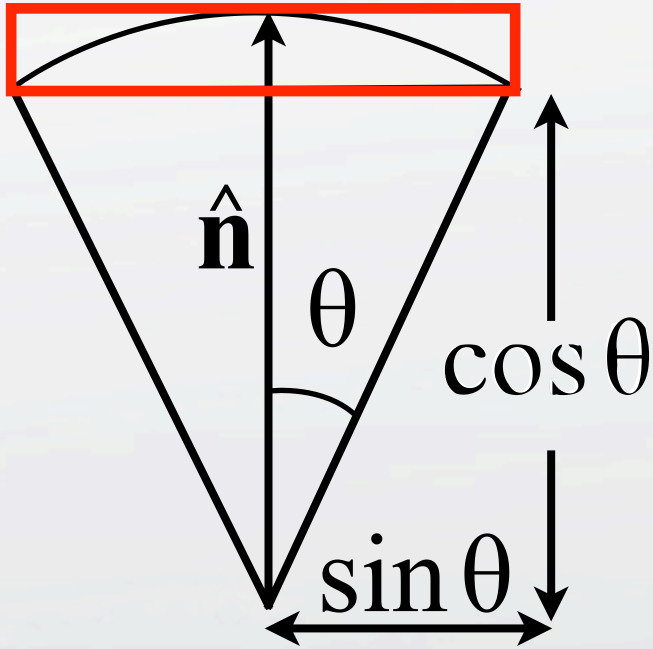
Bound Normal

- We need bounds of the *normalized* normal
  - Project control points of normal vector patch on unit sphere
  - Bound with a cone [Sederberg & Meyers, 1988]
  - Use the OBB coordinate frame to choose cone axis
    - Motivation: $\mathbf{n} = \mathbf{t} \times \mathbf{b}$ approximate surface normal



Bounds of Cone

- Cone axis aligned with OBB coordinate frame's z-axis
- Rotation symmetric
- Bounds in OBB coordinate frame given by cone angle:



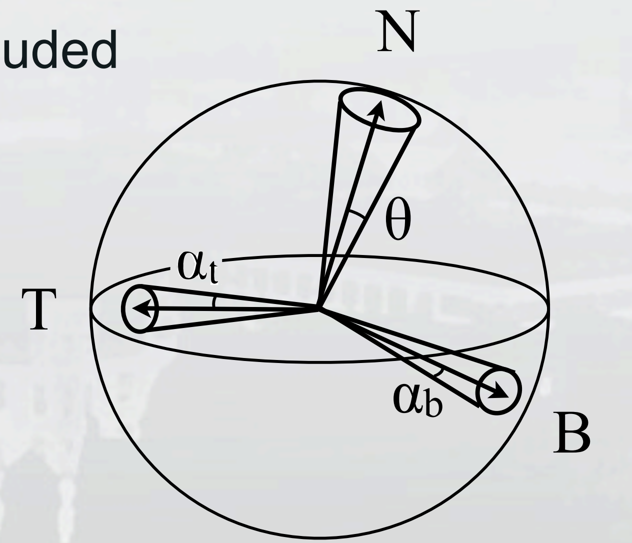
$$([-\sin\theta, \sin\theta], [-\sin\theta, \sin\theta], [\cos\theta, 1])$$

Faster Normal Bounds - Tangent Cones

- Deriving normal vector patch is costly
  - For bi-cubic patch: 144 cross products and 36 normalization operation needed to derive normal patch of bi-degree (5,5)

- Idea: Bound tangent patches by cones
  - Conservative “cross product of cones” gives normal bounds

- Coarser than normal vector patch
  - If tangent cones overlap, zero vector is included



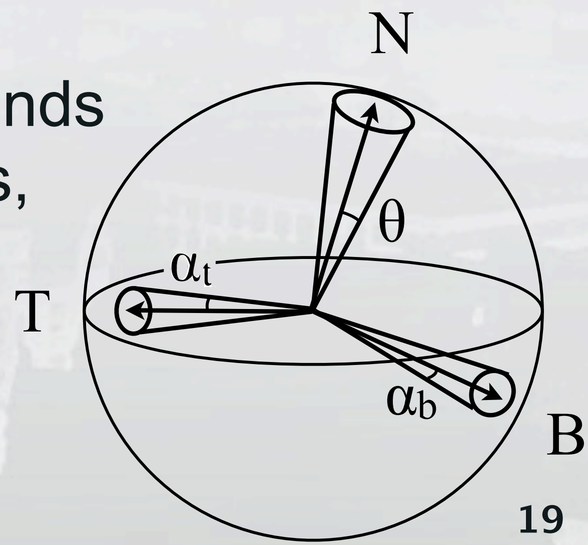
Bounds from Tangent Cones

- Use axes **t**, **b** (from OBB derivation) for tangent cones
  - Find cone angles α_t and α_b

- Normal cone given by [Sederberg & Meyers, 1988]:
  - Axis **n** = **t** x **b**
  - By construction, **n** is aligned with OBB frame
  - Cone angle:

$$\sin\theta = \frac{\sqrt{\sin^2\alpha_t + 2\sin\alpha_t\sin\alpha_b\cos\beta + \sin^2\alpha_b}}{\sin\beta}$$

- If the tangent cones don't overlap, N bounds all possible cross products of two vectors, one from each of T and B

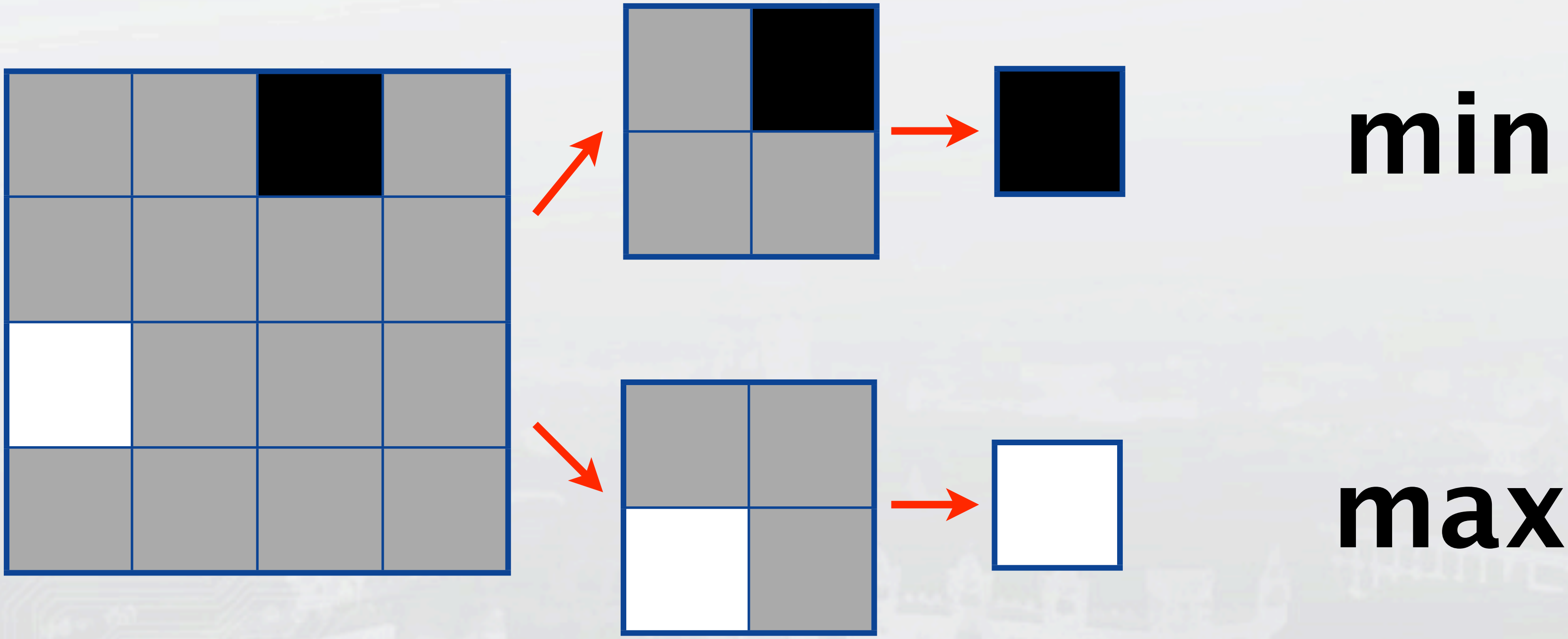


Sunday, June 27, 2010

Bounded Texture Lookups

- Use min/max MIP hierarchies [Moule & McCool, 2002]

$$[t_{min}, t_{max}]$$



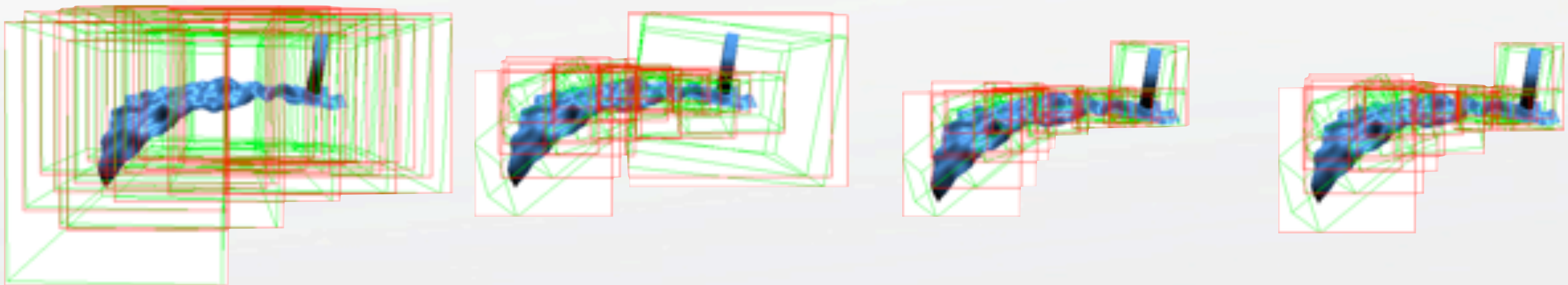
Final bounds

- All bounds expressed in the same OBB frame
  - Easy to combine, and give an OBB in object space
  - Transform OBB to clip space
  - Use resulting OBB for culling / binning

$$\mathbf{q}(u,v) = \mathbf{M}(\mathbf{p}(u,v) + \hat{\mathbf{n}}(u,v)t(u,v))$$

OBB + OBB x Interval

Evaluation - Algorithm Comparison



	CBOX Prev. Work	OBBTEX	TPATCH	NPATCH
Coordinate frame	AABB	OBB	OBB	OBB
Base patch	Bound CP	Bound CP	Bound CP	Bound CP
Normal vector	Unit sphere	Unit sphere	Tangent cones	Normal patch
Displace	User constant	min/max tex	min/max tex	min/max tex

Cost comparison

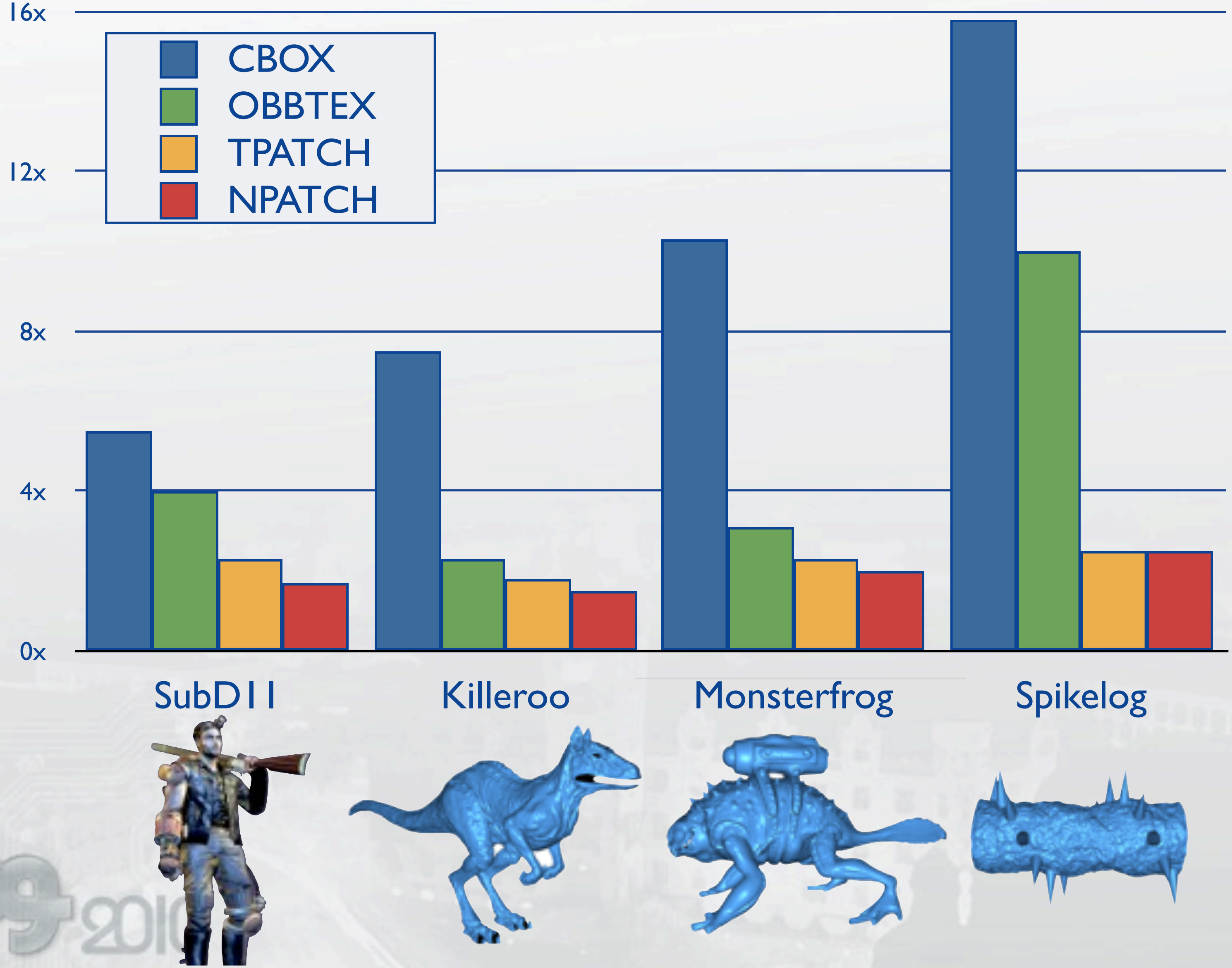
- Evaluate and bound a patch:
  - Compute bounds per patch - **one execution**
  - Evaluate per domain point - **thousands of executions**

$$\mathbf{q}(u,v) = \mathbf{M}(\mathbf{p}(u,v) + \hat{\mathbf{n}}(u,v)t(u,v))$$

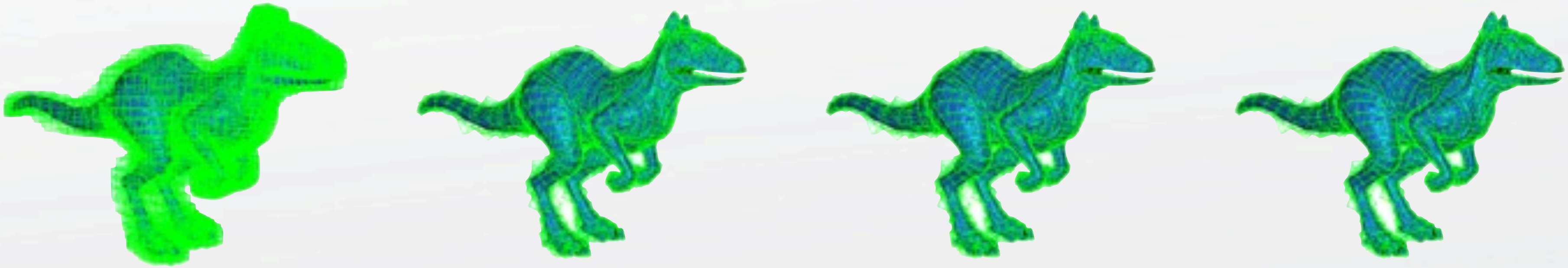
	#instr	ATI 5870	Intel Core i7
Domain shader	1	1	1
CBOX	1.5	1.6	1.5
OBBTEX	2.7	2.7	2.4
TPATCH	4.5	3.8	4.5
NPATCH	11	83	11

hpg 2010

Total Screen Space Area



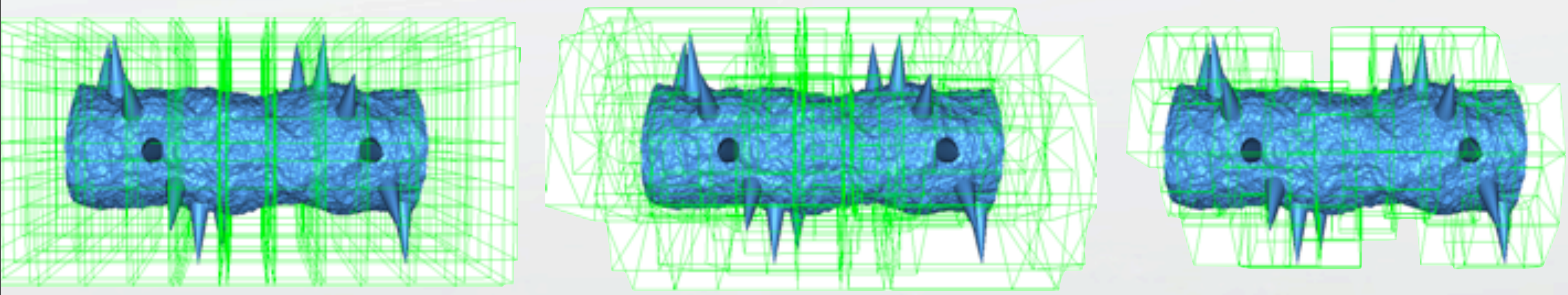
Killeroo



CBOX OBBTEX TPATCH NPATCH

Heatmap - Screen space bounds overlap

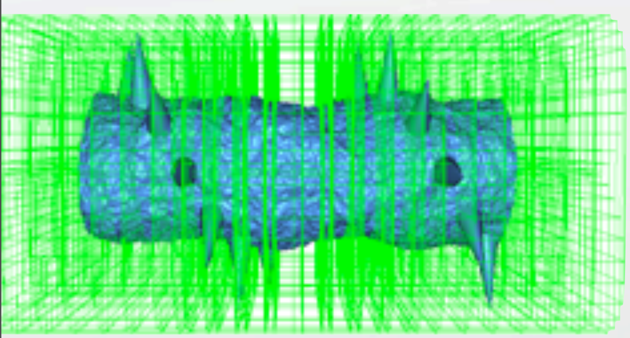
Convergence



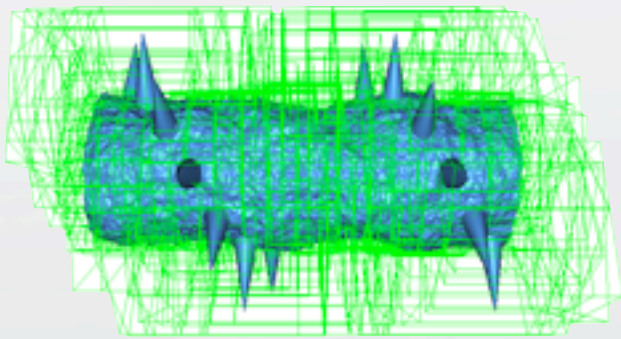
CBOX | OBBTEX | TPATCH

Subdivision: 1x

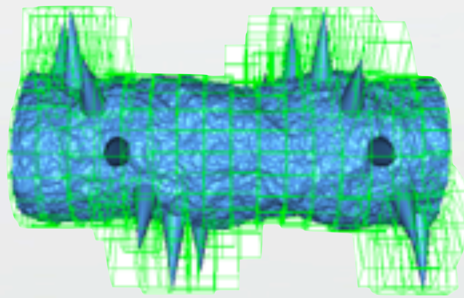
Convergence



CBOX



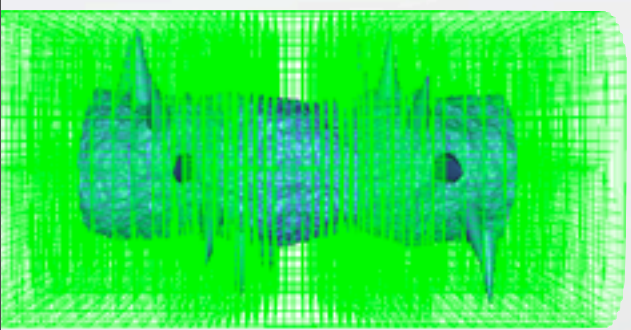
OBBTEX



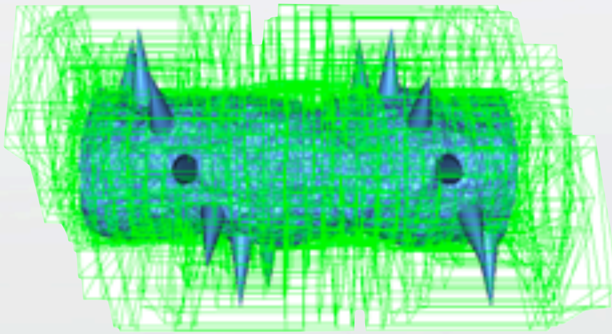
TPATCH

Subdivision: 4x

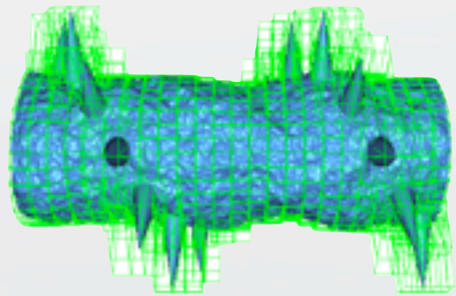
Convergence



CBOX



OBBTEX

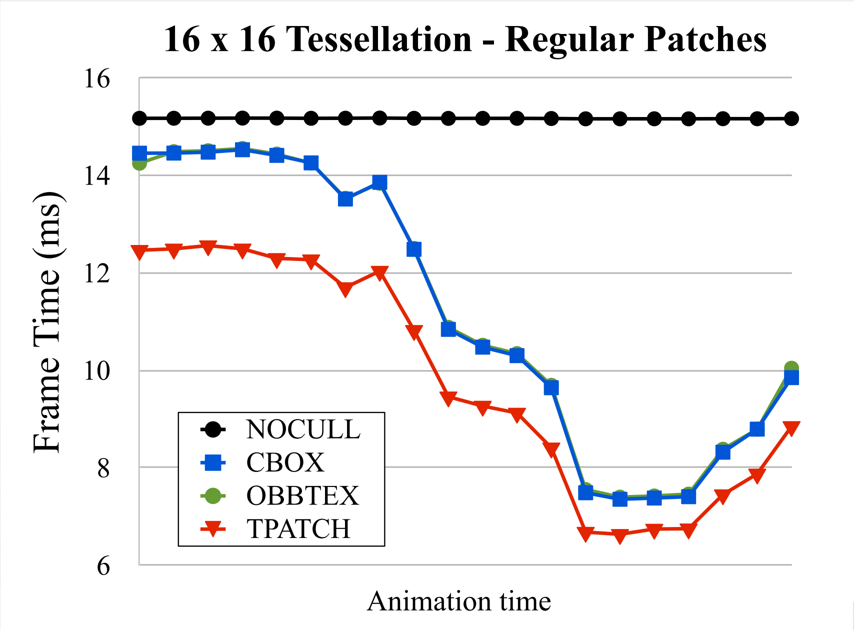


TPATCH

Subdivision: 16x

DX11 implementation

- Implemented all algorithms in DX11 hull shader for SubD11 SDK example
- Constant displacement in normal direction
  - Special case - allows for backface culling



- Improves slowest frame

Summary

- Algorithms for bounding displaced parametric surfaces
- Pros
  - Handles difficult cases, e.g. large displacements, well
  - Converges quickly when subdividing base patch
  - Low bounding cost
    - ~4x compared to a **single** domain shader execution
- Cons
  - Approximate catmull clark + bounding algorithms put strain on graphics hardware
  - Increased memory footprint (min/max mipmaps)

hpg 2010

Acknowledgements

- Thanks
  - Royal Swedish Academy of Sciences - Knut & Allice Wallenberg Foundation
  - Swedish Foundation for strategic research
  - Intel Advanced Rendering Technology team
  - Anonymous reviewers

- Models
  - SubD11 - Microsoft DirectX11 sample
  - Killeroo - Headus 3d tools
  - Monsterfrog - Bay Raitt, Valve Software

Thank you