DirectX11最新
リアルタイム映像事例集

Philosophy

- Motivation: Movie Quality in Real-Time
- Approach: DX11 latest features

Remi

Ivan

Iwata

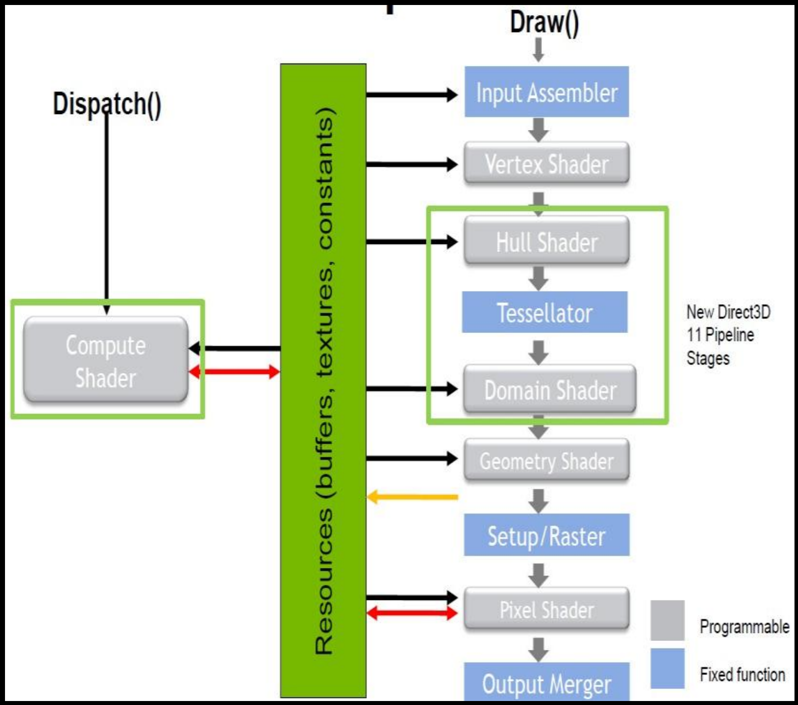
Tech

Art

Agenda

- Tessellation
  - Displacement mapping
  - Subdivision (ACC)
  - Creases
  - Performance & Scalability
  - Problems
  - Non-surface tessellation (hairs)
- Compute Shader
  - Cloth
  - Particles
- Other
  - Reflection

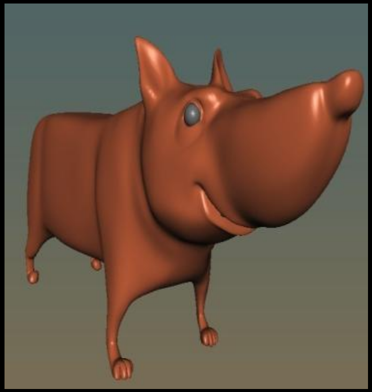
DirectX 11 new features



PART 1
TESSELLATION

Motivation

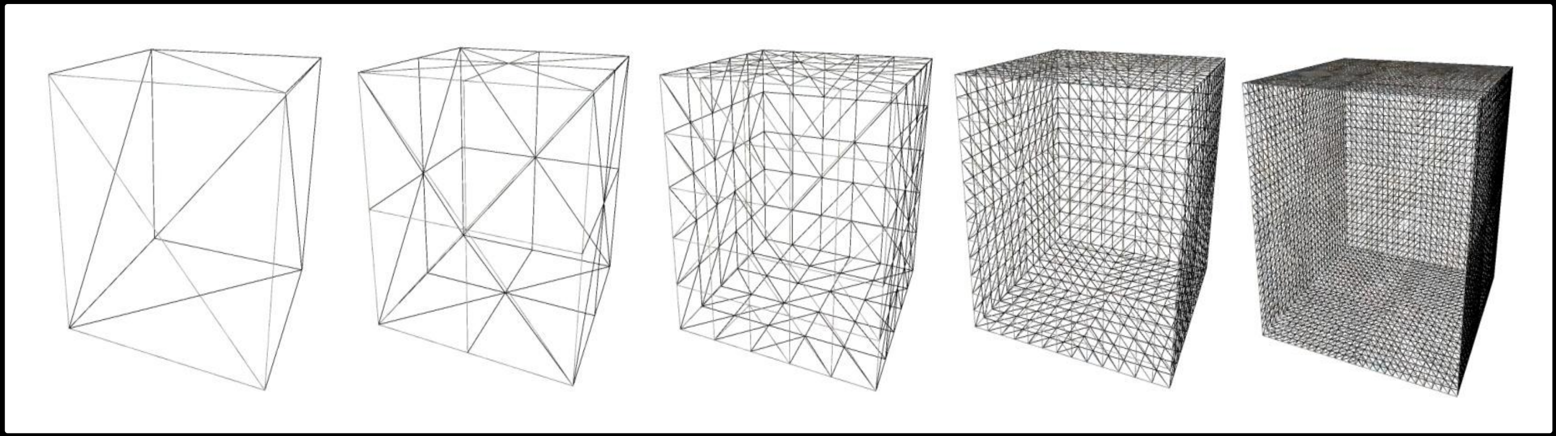
A scalable tech to render & animate wide range of high quality surfaces

Both sharp and smooth details!

Concept

take a polygon and dice it
into smaller pieces

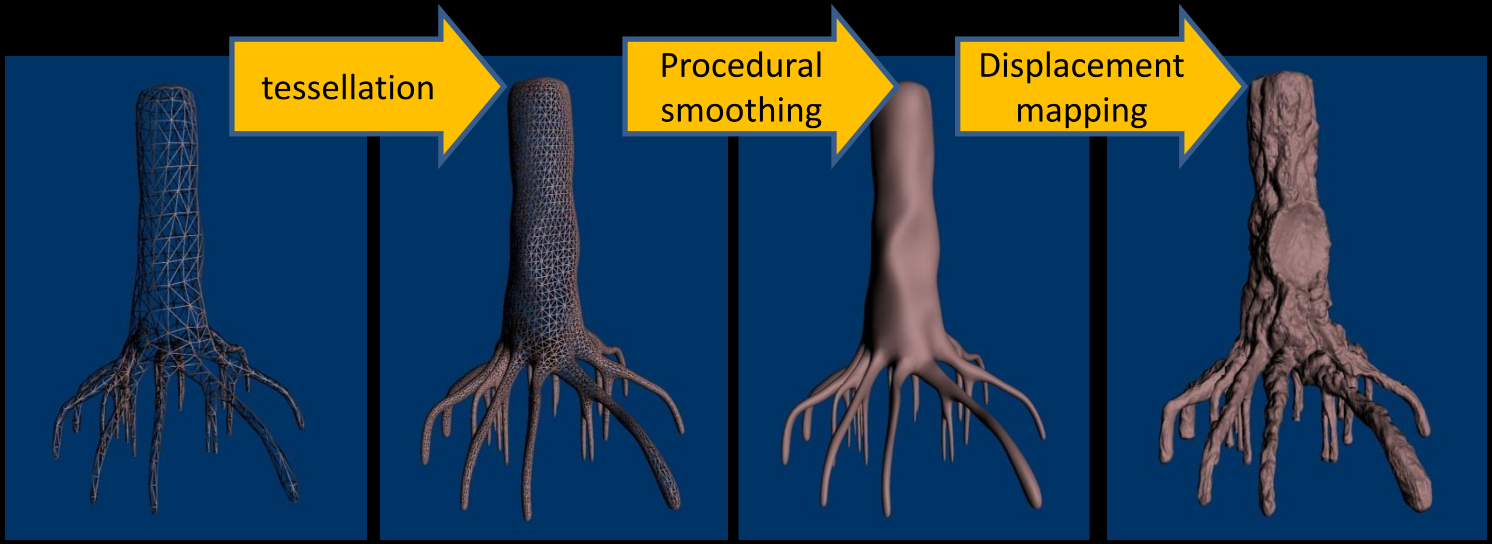


…but: flat surfaces remains flat

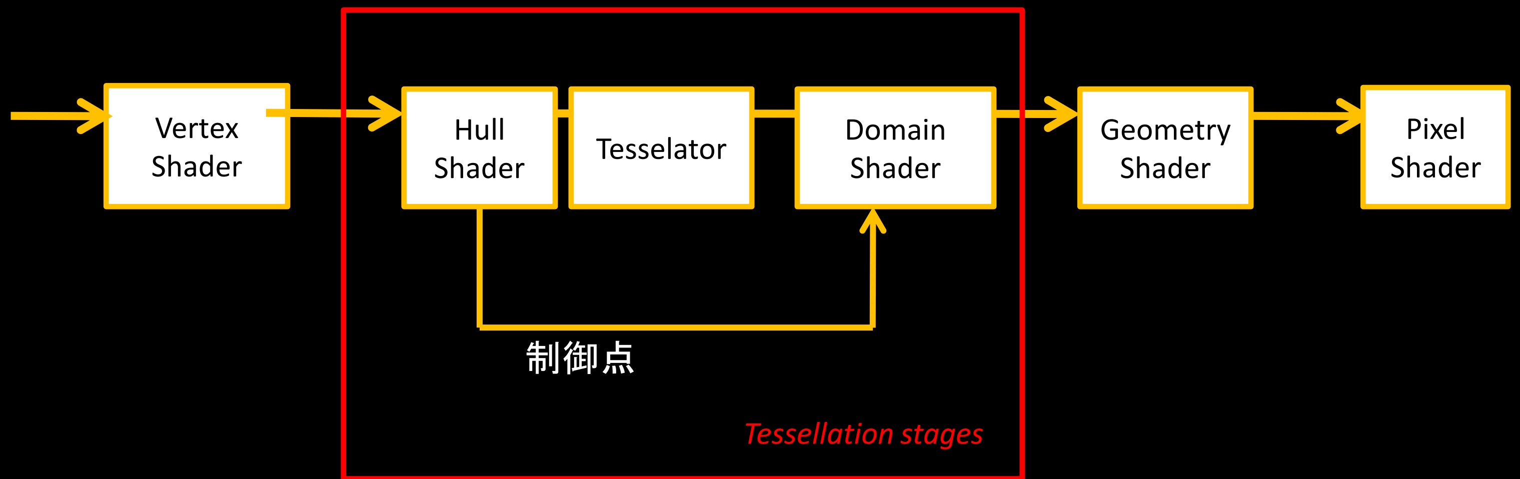
Refinement schemes

Can use displacement procedures to create high resolution surfaces:

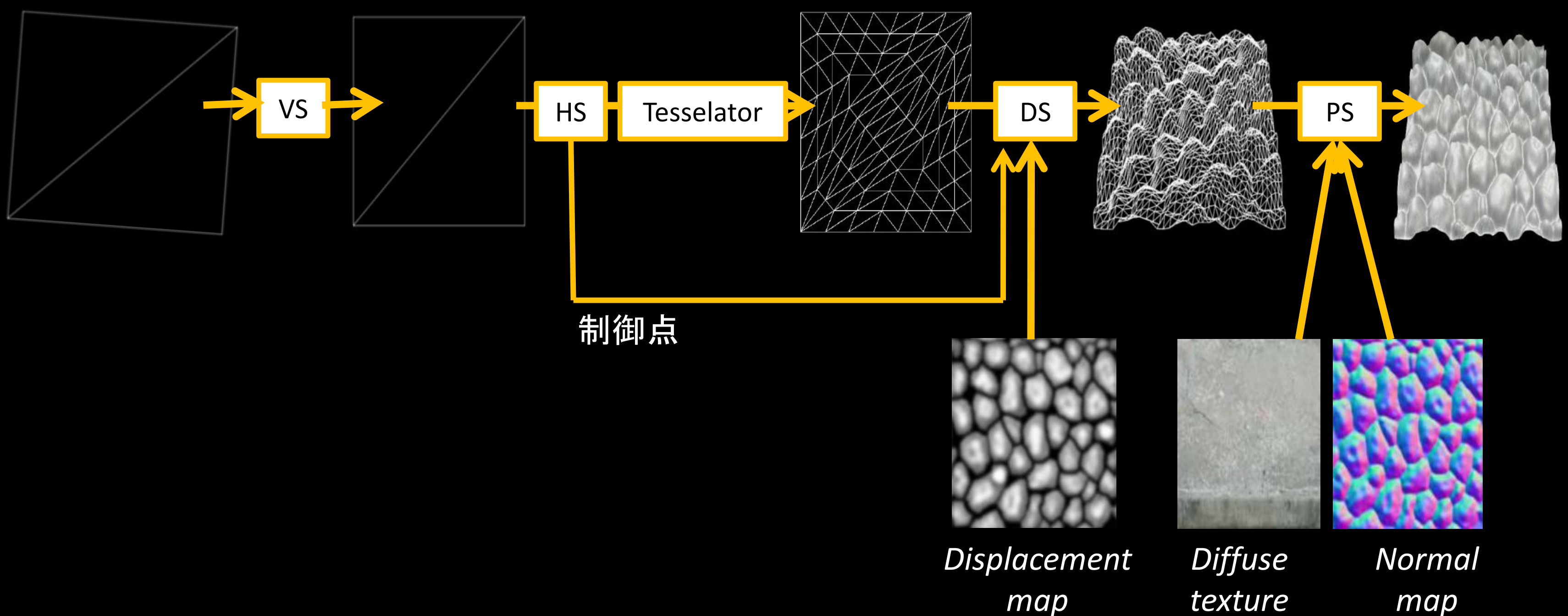
– Data-Sampling
– Procedural



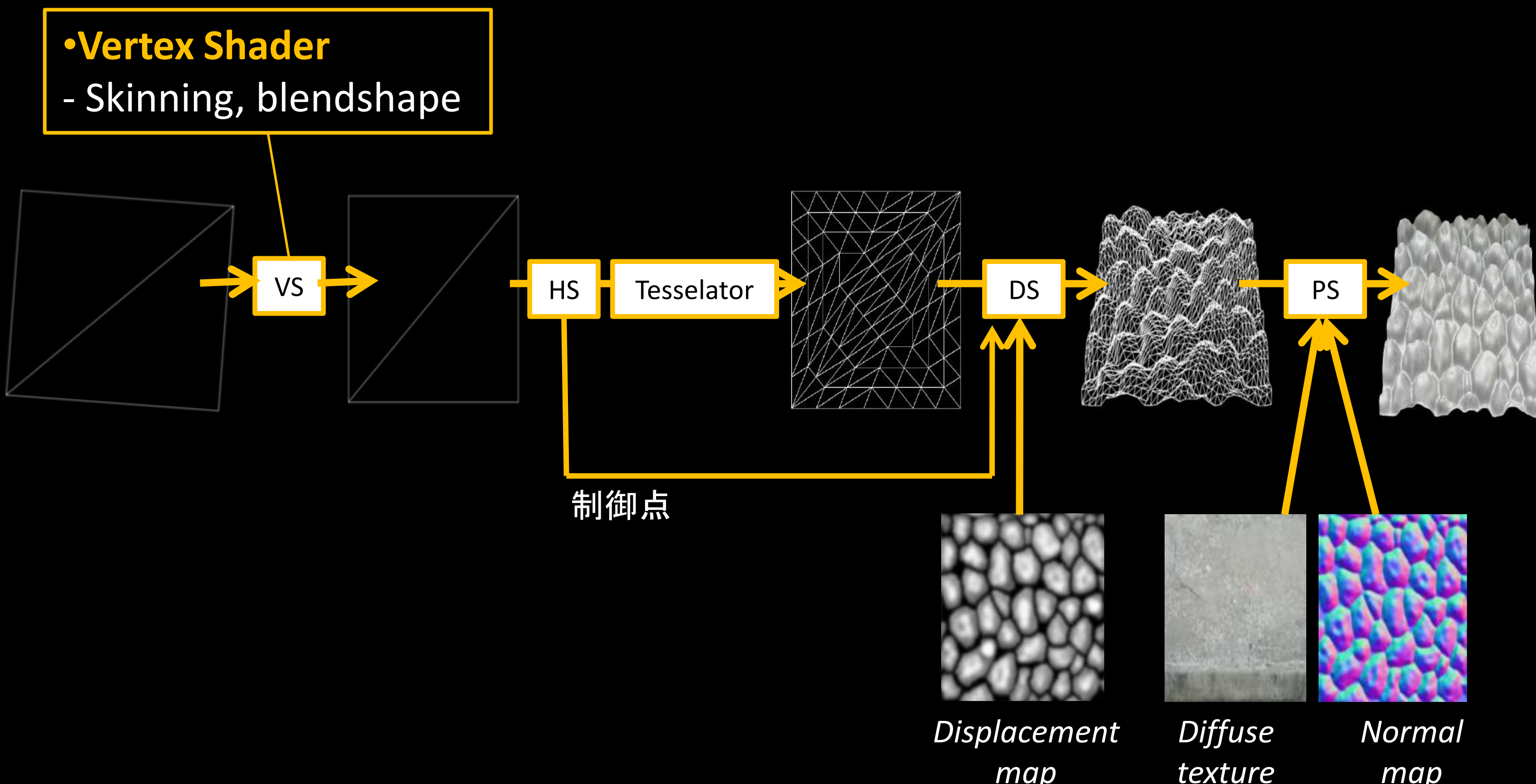
DX11 pipeline



Basic pipeline



Basic pipeline



Basic pipeline

•Hull Shader

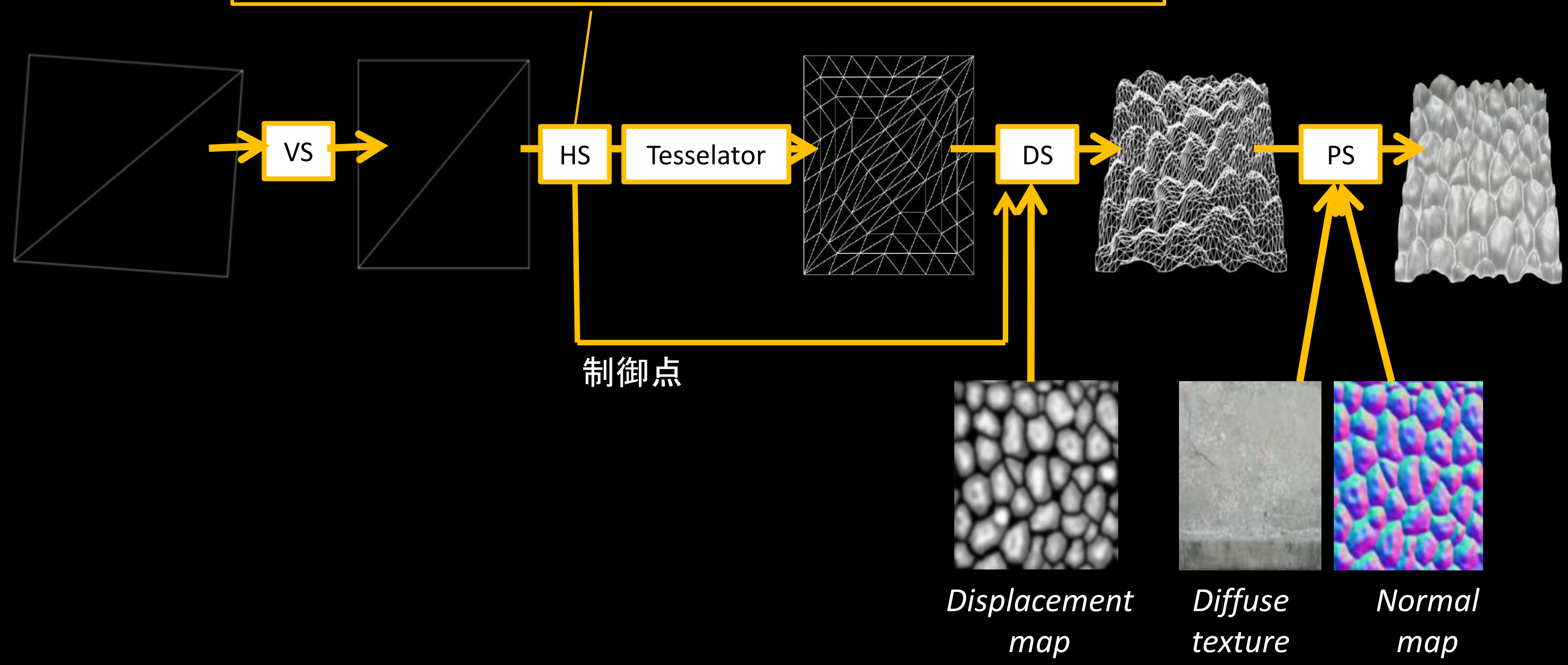
-New primitive type : patch

-Conversion from one surface type to another

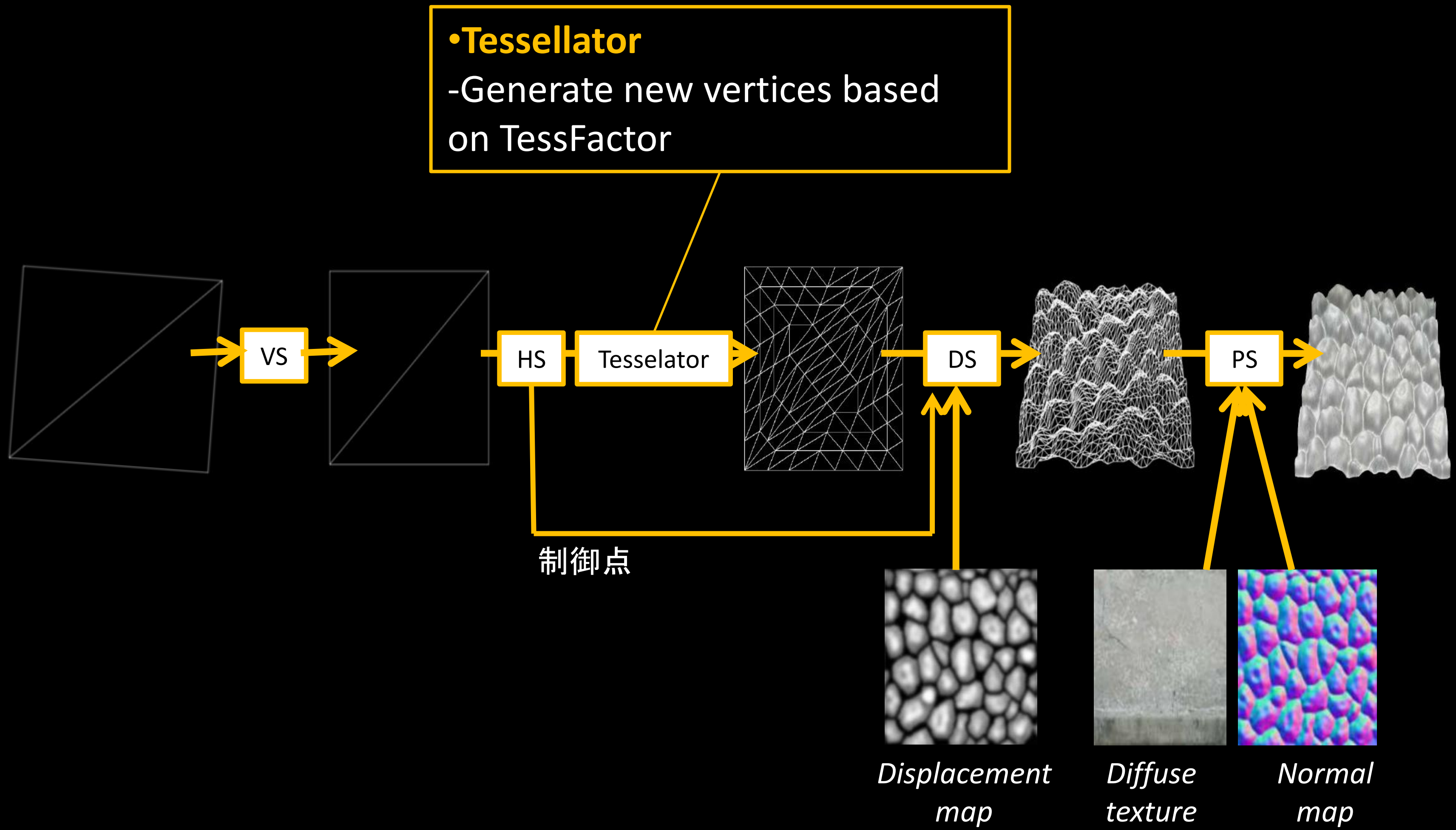
(e.g. subdivision surface to bezier patches)

-Tessellation factors per patch

-Runs once per control point

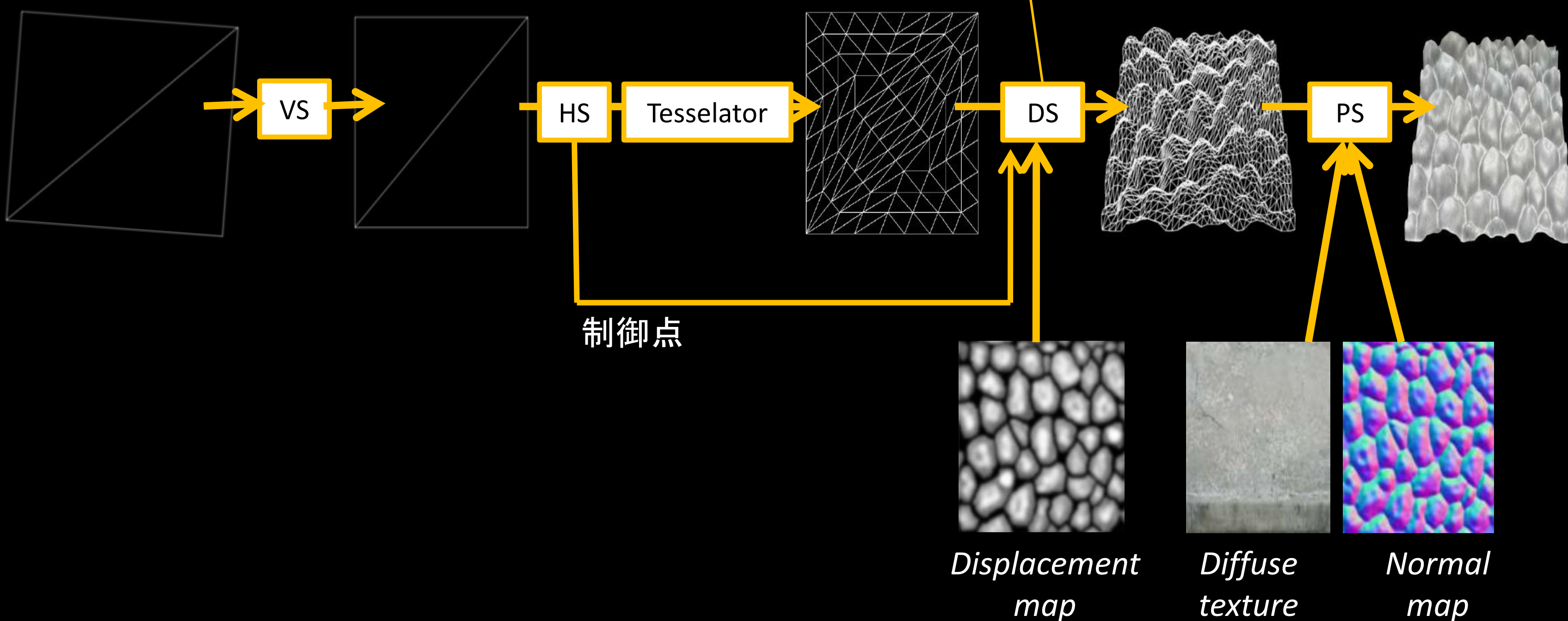


Basic pipeline

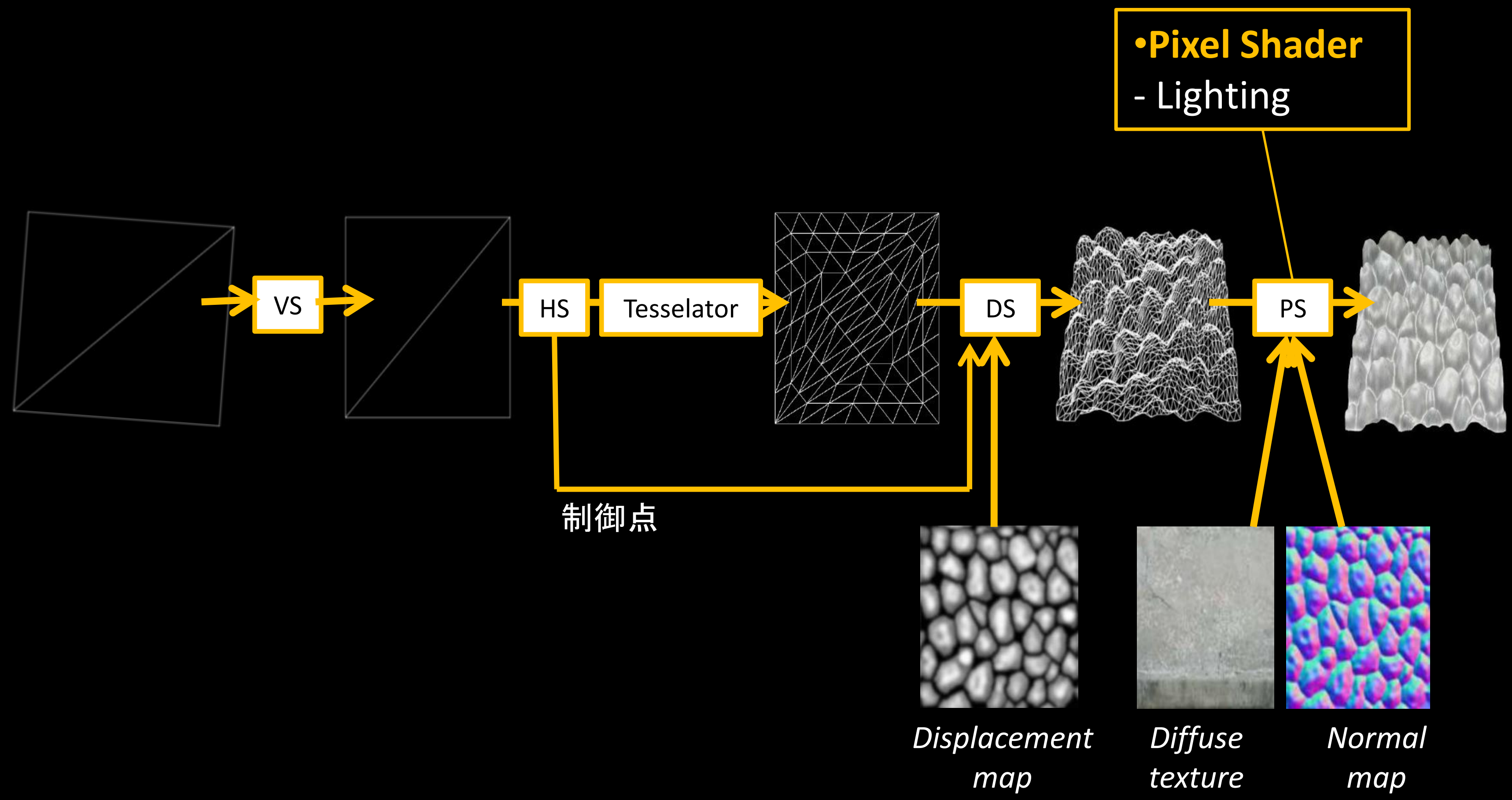


Basic pipeline

- **Domain Shader**
  - -One invocation per vertex
  - -Evaluate surface given UV coordinates
  - -Apply displacements

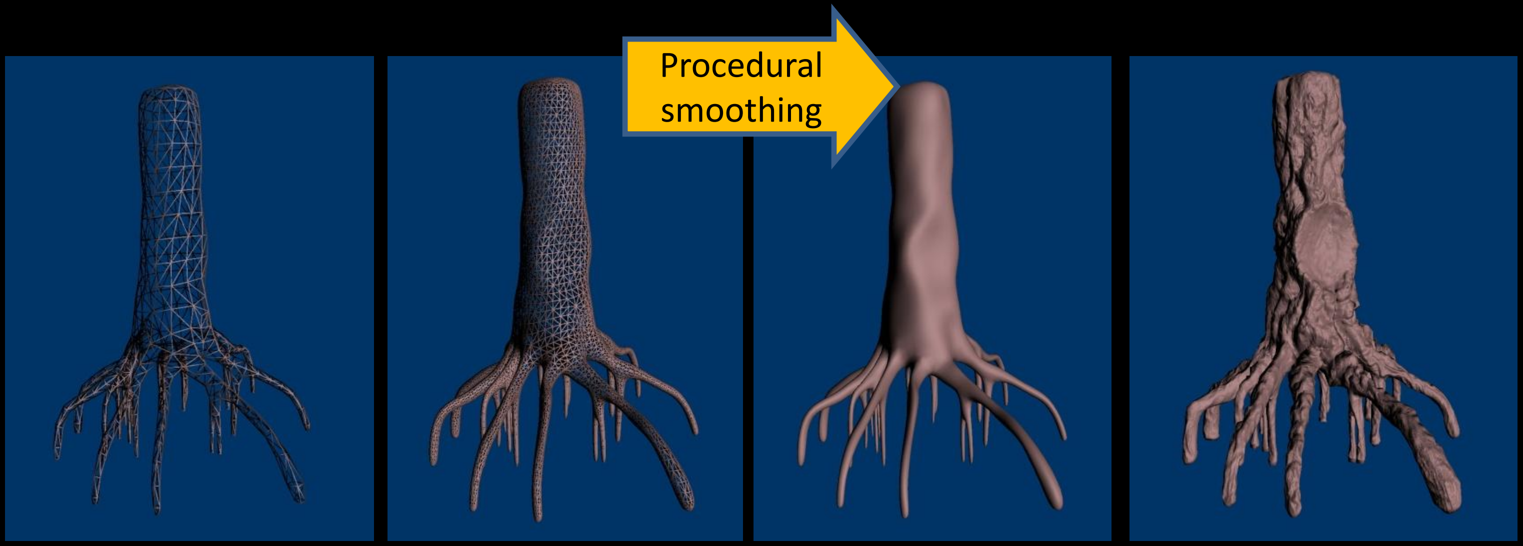


Basic pipeline



“Smoothing” Schemes

= procedural displacements



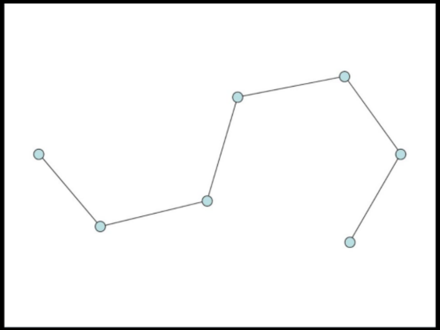
Local schemes

- Use local vertex info
  - (PN-Triangles, Phong tessellation, etc…)
- Merit
  - Easy to implement
- Problems
  - No guarantees on higher level continuity
  - Is limited in the amount of curvature

Subdivision

•Global scheme = depends on the surrounding vertices

•Subdivision surface(細分割曲面) =
the limit of successive refinements



•History

-3D surface subdivision dates back to 1978

-Used for ages in the Movie Industry

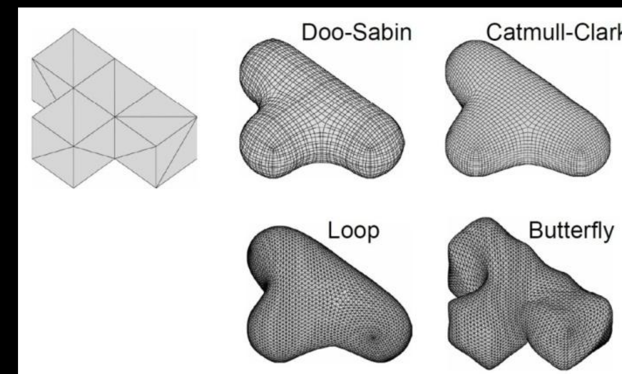
Geri's Game (1989) : Pixar Animation Studios

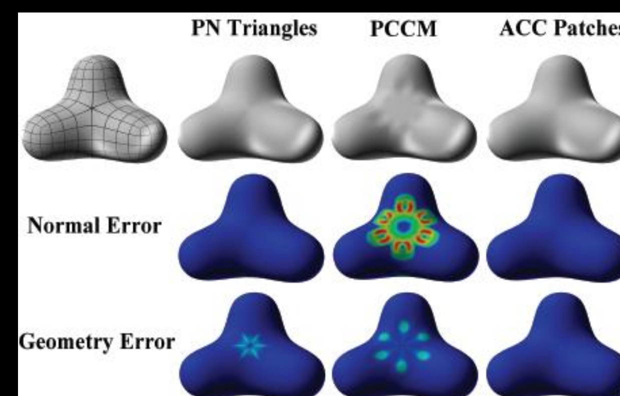
...But it was not real time until now !

Our Choice

- Many “smoothing” Schemes exist:
  - *Catmul Clark (Pixar, Maya)*
  - *Doo Sabin*
  - *Loop*
  - *Butterfly – Nira Dyn*
  - *Etc…*

	Primal		Dual
	Triangles	Rectangles	
Approximating	Loop	Catmull-Clark	Doo-Sabin Midedge
Interpolating	Butterfly	Kobbelt	





From [Loop & Schaefer]

•Our choice: ACC *[Loop & Schaefer]*

-Approximating Catmull-Clark Subdivision Surfaces with Bicubic Patches

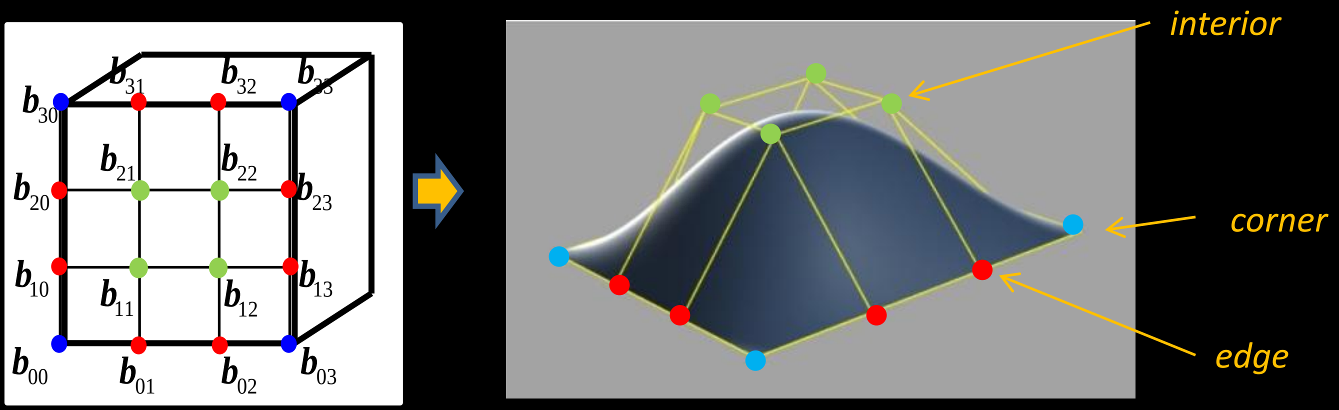
-Catmull Clark is what all tools are using !

References

A good survey of local refinement schemes	Tianyun Ni, Ignacio Casta no, Jörg Peters, Jason Mitchell, Philip Schneider, and Vivek Verma. Efficient substitutes for subdivision surfaces. In ACM SIGGRAPH Courses, pages 1–107, 2009.
A good introduction to subdivision surfaces	D. Zorin, P. Schröder, A. Levin, L. Kobbelt, W. Sweldens, and T. DeRose. Course Notes Subdivision for Modeling and Animation. In ACM SIGGRAPH, 2000
ACC	"Approximating Catmull-Clark Subdivision Surface with BicubicPatches" by Charles Loop and Scott Schaefer, ACM Transactions on Graphics, Vol. 27 No. 1 Article 8 March 2008.
ACC(gregory patches)	"Approximating Subdivision Surface with Gregory Patches for hardware Tessellation" by Charles Loop, Scott Schaefer, Tianyun Ni, Ignacio Castano, Siggraph Asia 2009.

ACC principle

- Principle:
  - Use Bezier patches to approximate Catmull-Clark surface
  - Shape of patches depends on control points
    - 3 Types: Corner / Edge / Interior
  - For each quad construct a geometry patch and tangent patches



ACC principle

- Geometry patches
  - For each patch, control points are calculated from surrounding vertices as a weighted average using weight masks
  - *(almost)* smooth everywhere

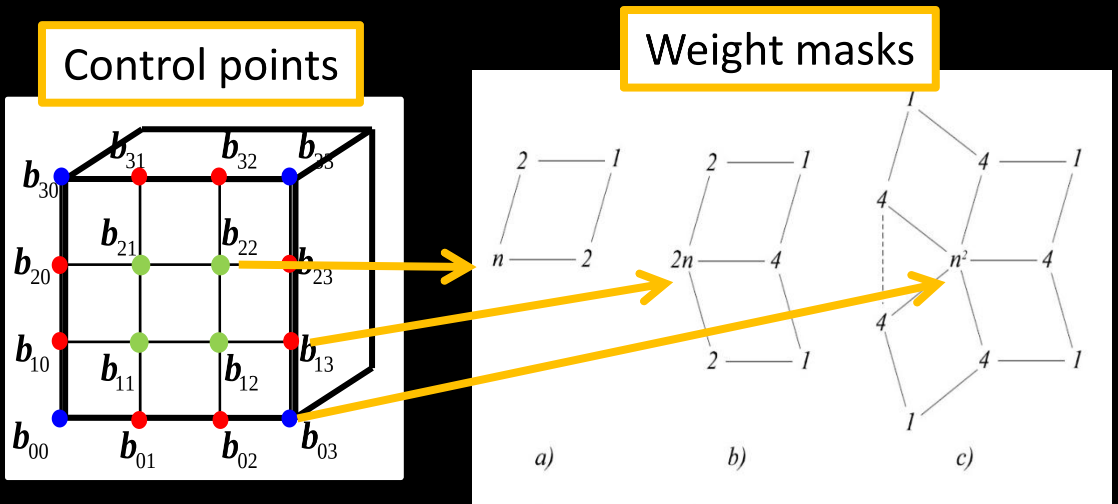
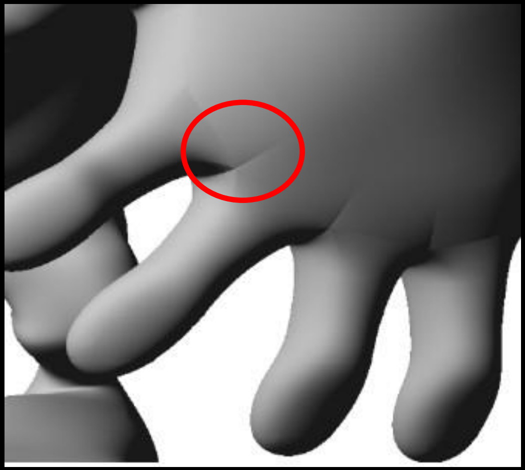


Fig. 4. Generalized masks for interior, edge and corner points.

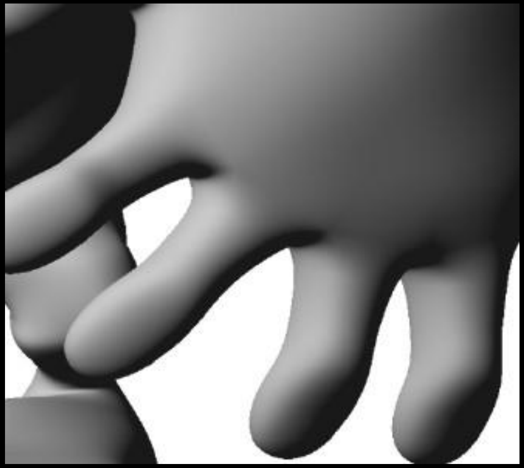
(almost) smooth everywhere ?!

- Patches are not smooth along edges leading to an extraordinary vertex !
- Tangent patches
  - Produce a continuous normal field
  - Make the patch surface appear smooth

Geometry patch only

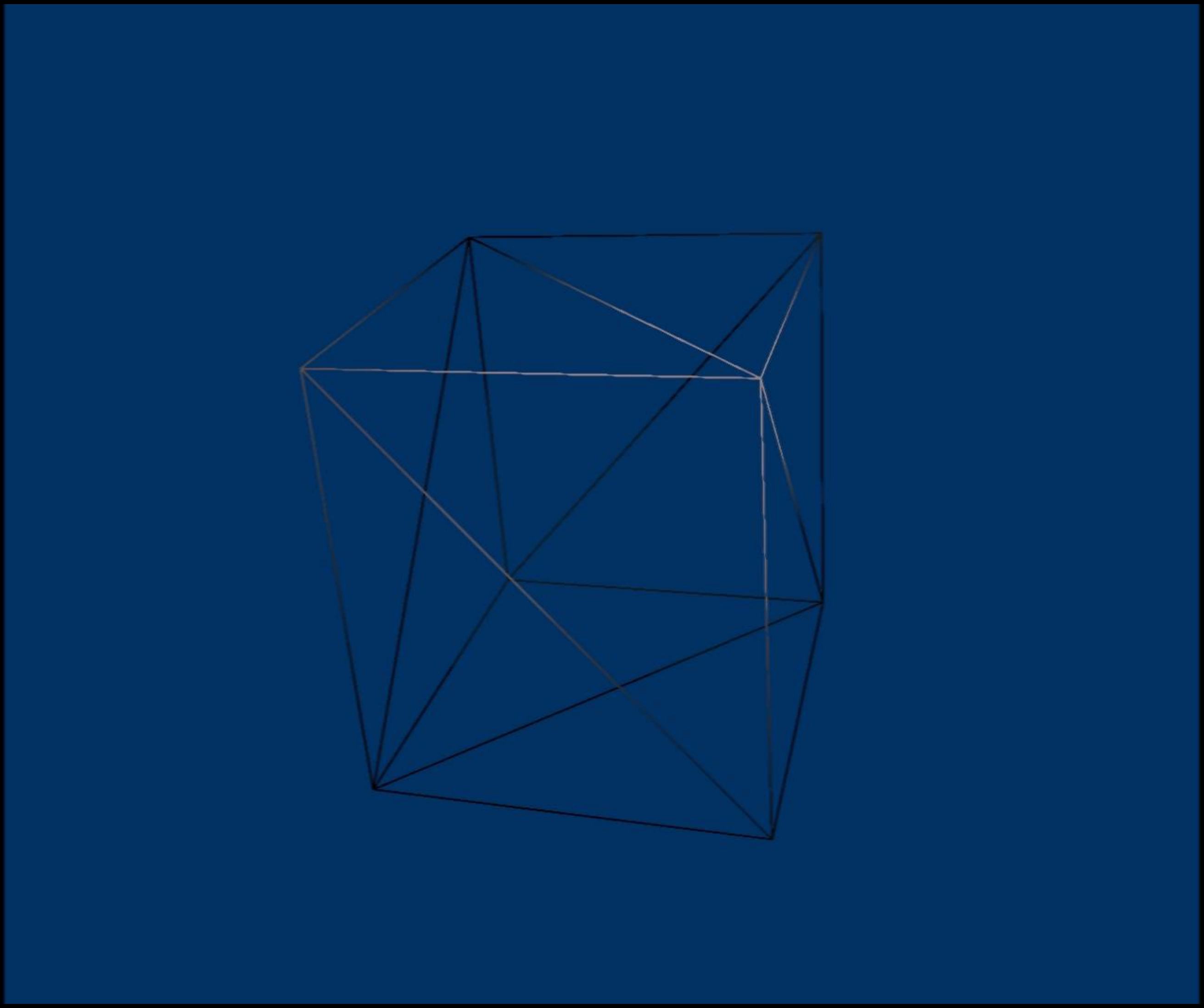


Geometry + tangent



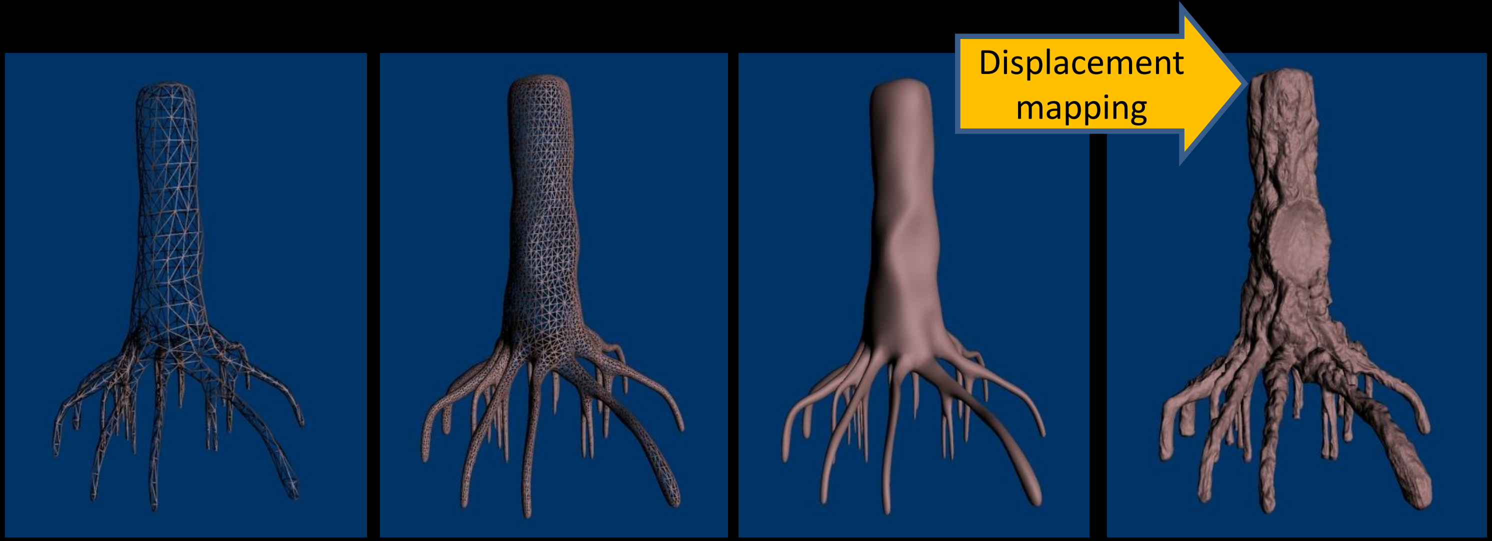
(From Loop & Schaefer)

Basic ACC Result



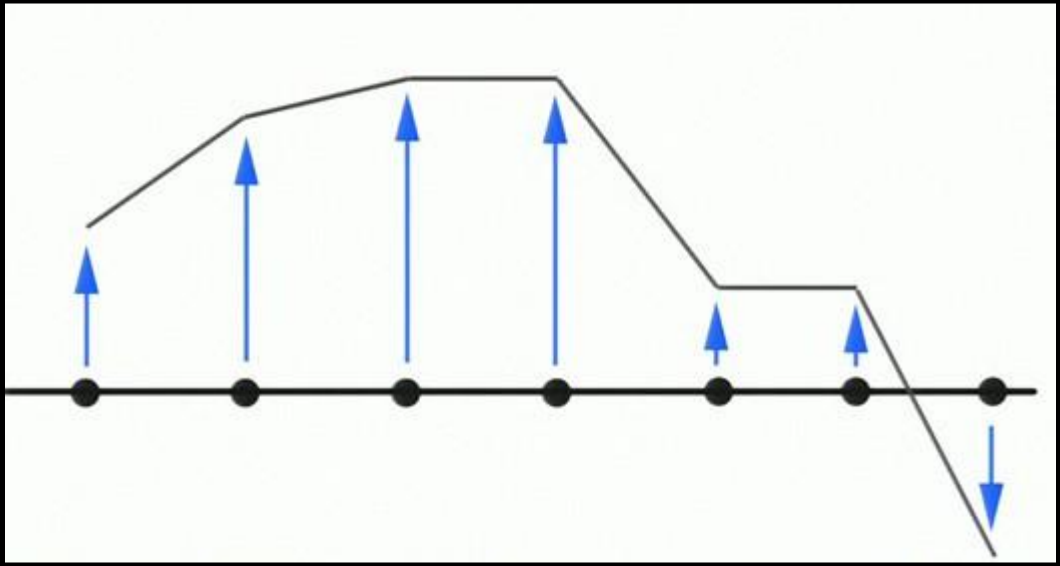
Displacement mapping

(=displacement based on data sampling)

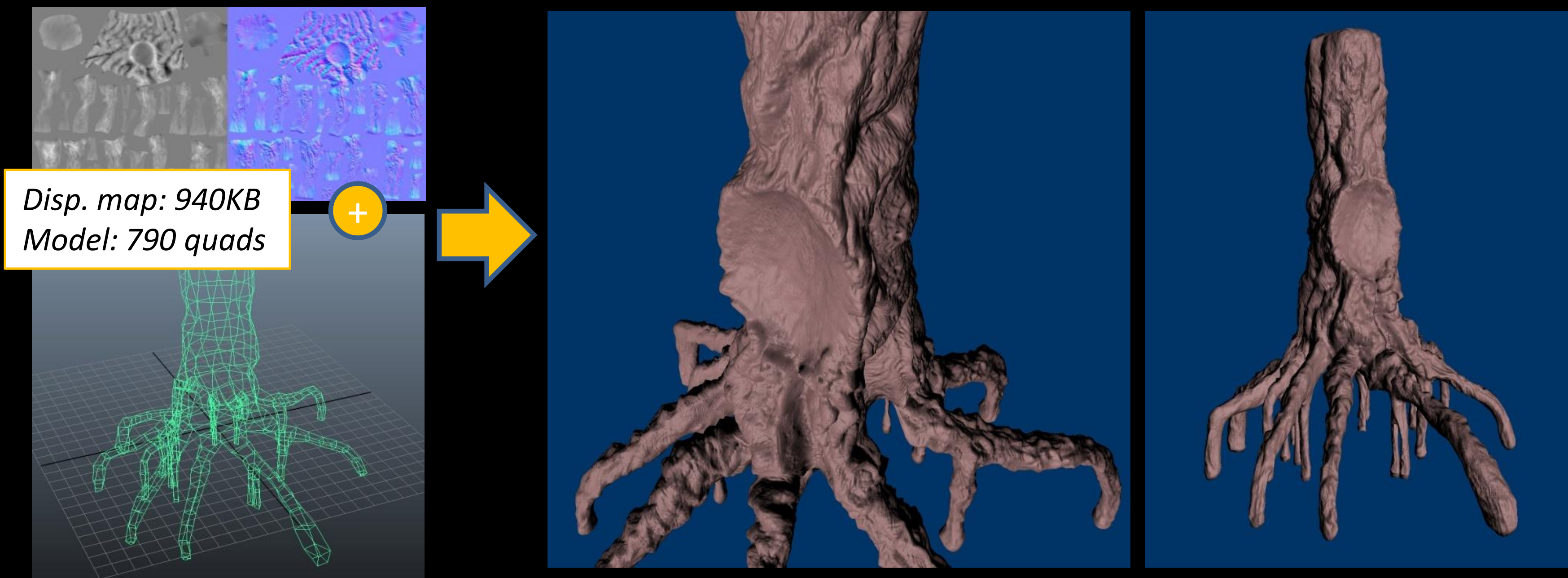


Concept

*Vertex position += normal * height from texture*



Example

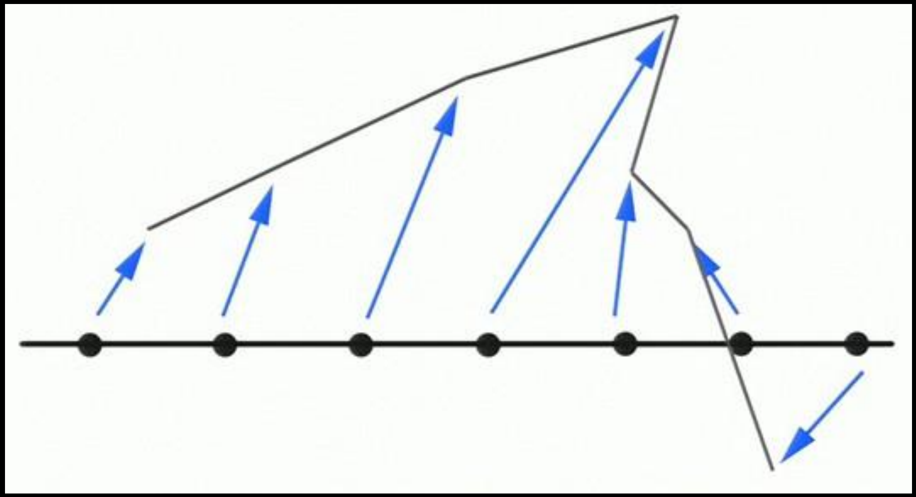


- Nice silhouette
- Workflow: similar to normal map

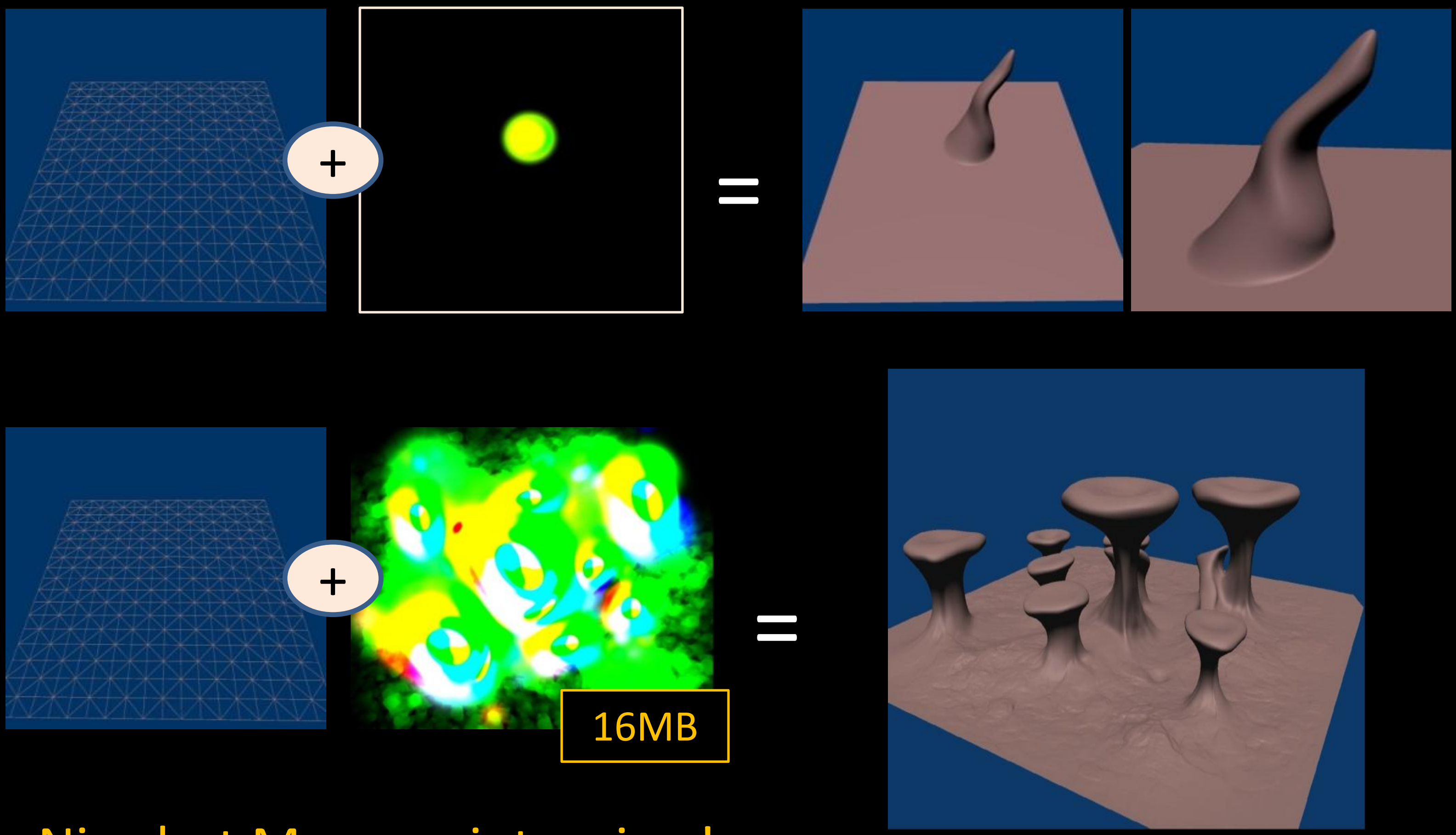
Results

“Vector” Displacement

- Vertex position += 3D info from texture
- Overhangs are possible !

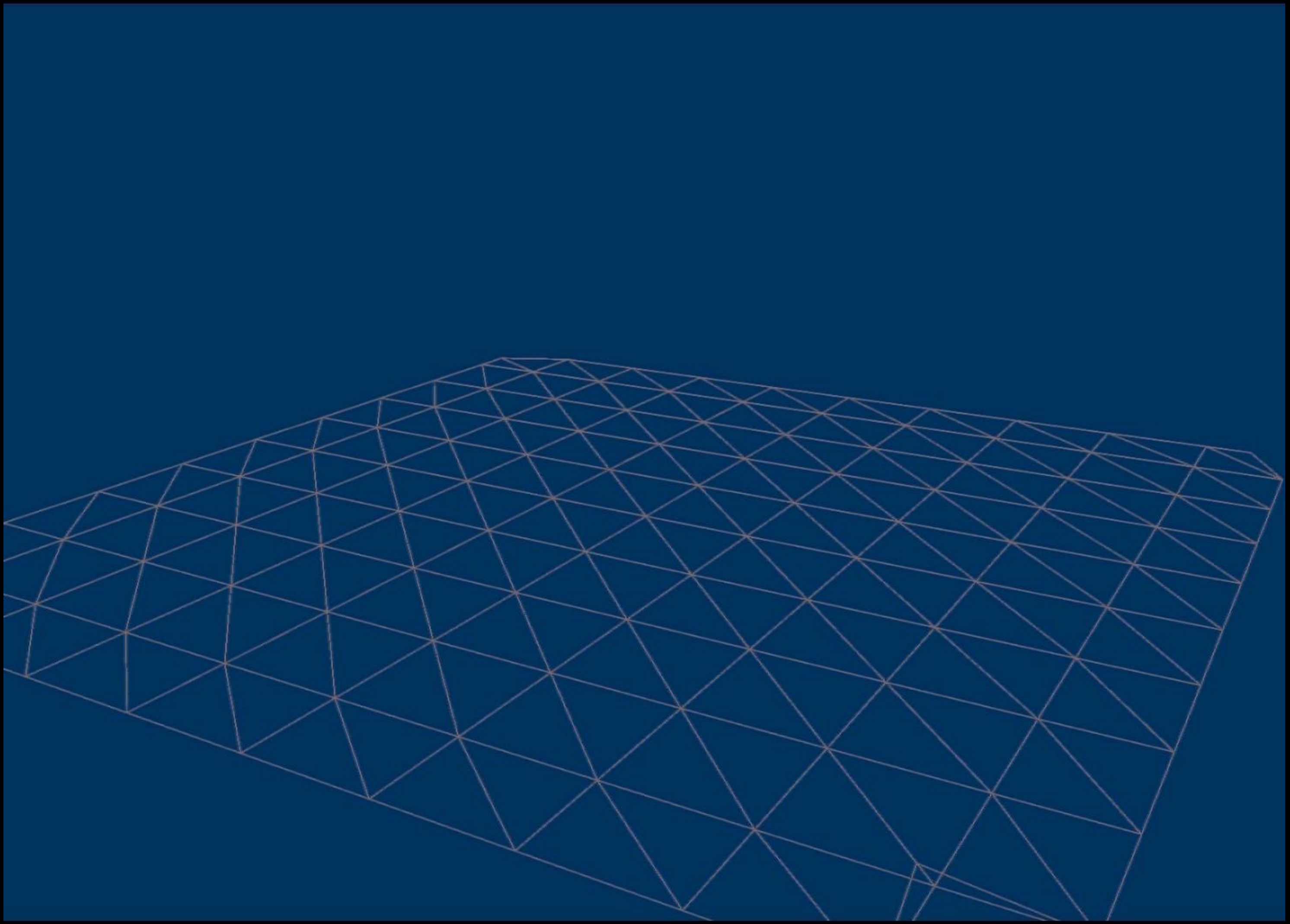


Example



Nice but Memory intensive !

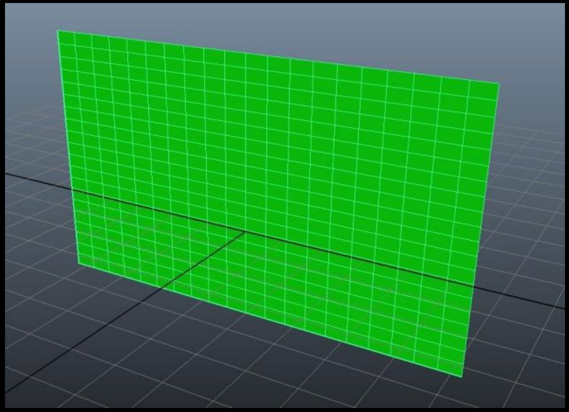
Results



Tests:
performance & scalability

GPU Performance

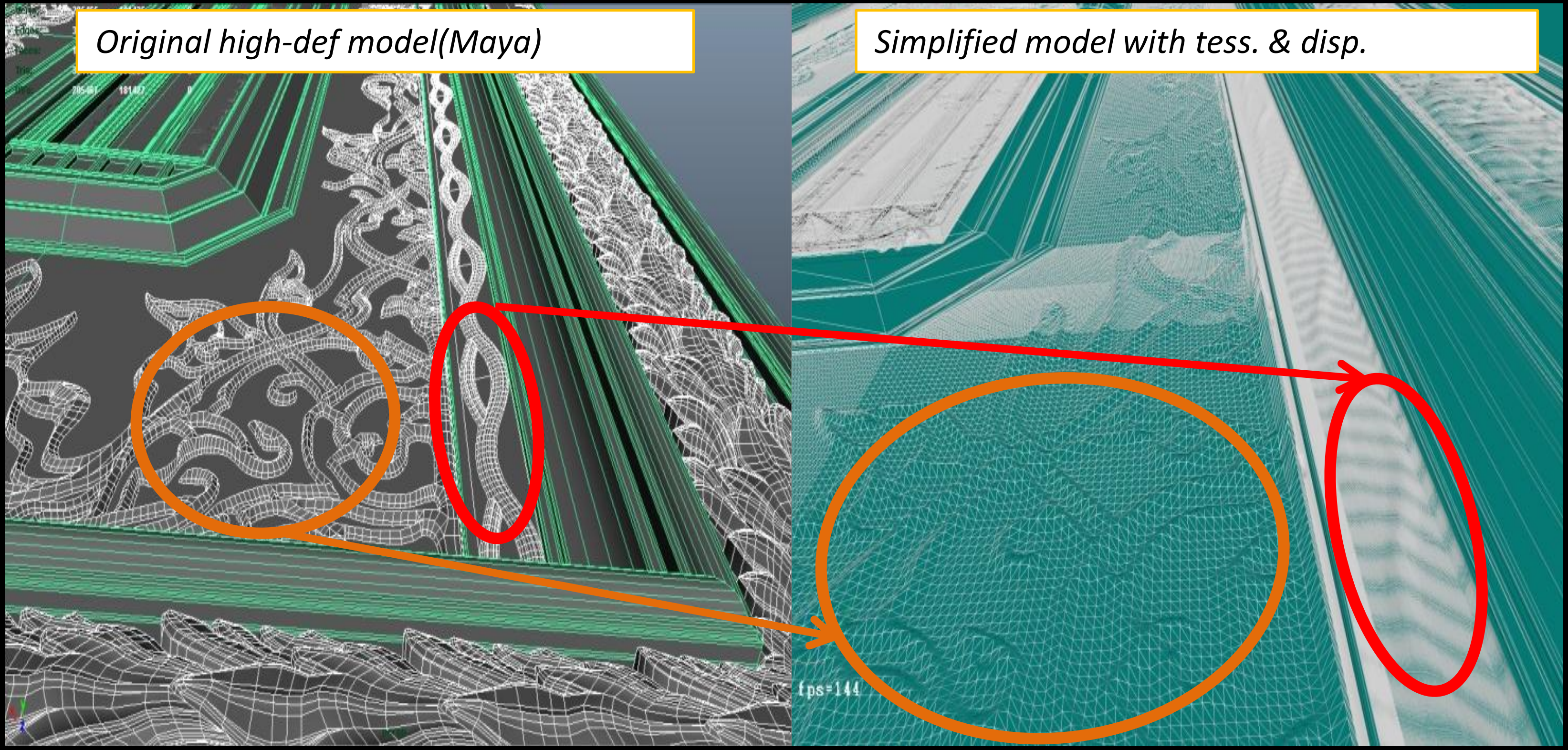
- GPU Time (NSight)
- Model: Static "grid" mesh
- Results:
  - Complex shader = slow
  - Tessellation has overhead
    - TessFactor=1 bad!
  - faster to display the high-res model



			Use tessellation + diffuse texture + displacement map + normal map									
	No tessellation geom. + diffuse tex		TessFactor=1		TessFactor=2		TessFactor=4		TessFactor=8		TessFactor=16		TessFactor=32	
Model Quads NB	Screen QuadNb	microsec	Screen QuadNb	microsec	Screen QuadNb	microsec	Screen QuadNb	microsec	Screen QuadNb	microsec	Screen QuadNb	microsec	Screen QuadNb	microsec
1	1	183	1	185	4	185	16	197	64	196	256	196	1024	203
100	100	197	100	195	400	203	1600	209	6400	258	25600	352	102400	791
250	250	189	250	197	1000	205	4000	237	16000	305	64000	503	256000	1651
500	500	194	500	203	2000	216	8000	260	32000	364	128000	882	512000	3359
750	750	195	750	207	3000	224	12000	277	48000	422	192000	1294	768000	4955
1000	1000	198	1000	219	4000	244	16000	304	64000	522	256000	1690	102400 0	6459
2000	2000	203	2000	230	8000	267	32000	359	128000	910	512000	3299		
4000	4000	220	4000	278	16000	334	64000	573	256000	1809	102400 0	6621		
8000	8000	275	8000	317	32000	395	128000	821	512000	3535				
10000	10000	284	10000	348	40000	456	160000	1359	640000	4429				
16000	16000	296	16000	475	64000	661	256000	2127	102400 0	6980				
20000	20000	304	20000	563	80000	793	320000	2656						
30000	30000	326	30000	784	120000	1116	480000	3975						
40000	40000	345	40000	1001	160000	1439	640000	5285						
60000	60000	383	60000	1418	240000	2074								
80000	80000	415	80000	1797	320000	2670								
120000	120000	556	120000	2550	480000	3799								
160000	160000	735	160000	3232										
200000	200000	916												
300000	300000	1364												

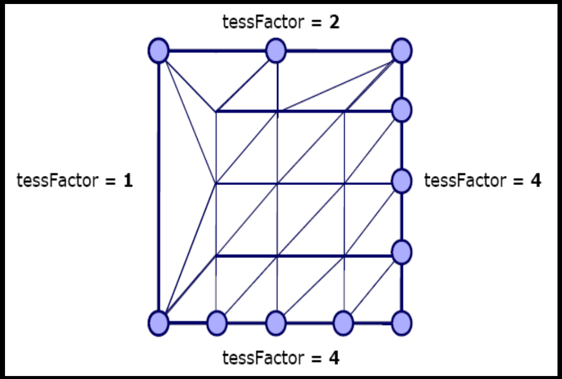
How not to use Tessellation

- Add local relief details on a static object ?
  - Naïve tessellation = bad idea
  - at least use a density map to avoid waste…

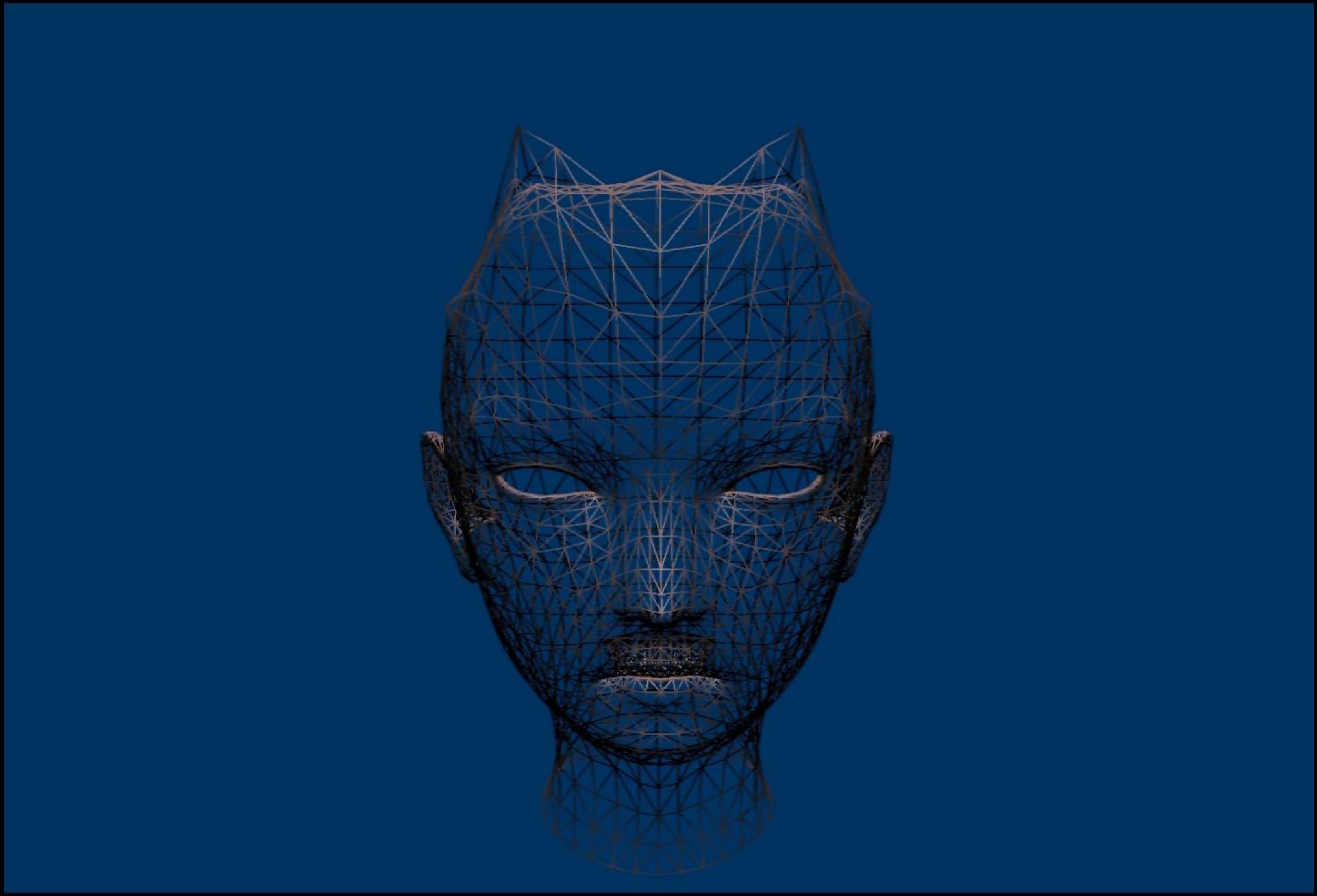


Adaptive Tessellation

- LOD algorithm can run on GPU
- Patch-level LOD
  - Detail-density maps
  - Distance of patch to camera
  - Orientation of patch to camera
  - Screen-space size of patch
  - Frustum culling(tessFactor=0)

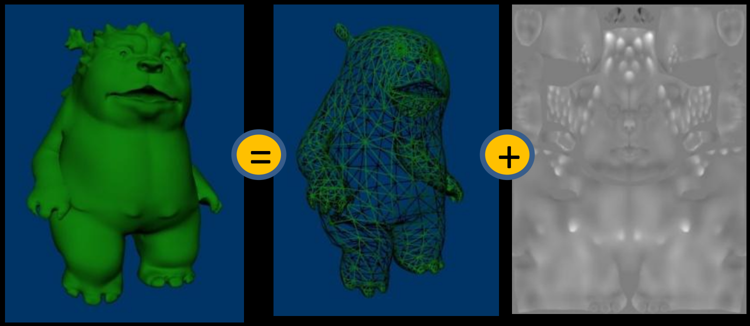


Video



How to use Tessellation

- Model: artist-created skinned mesh
- Test:
  - High-Res model vs. ACC + Disp. Map
  - Similar visual quality



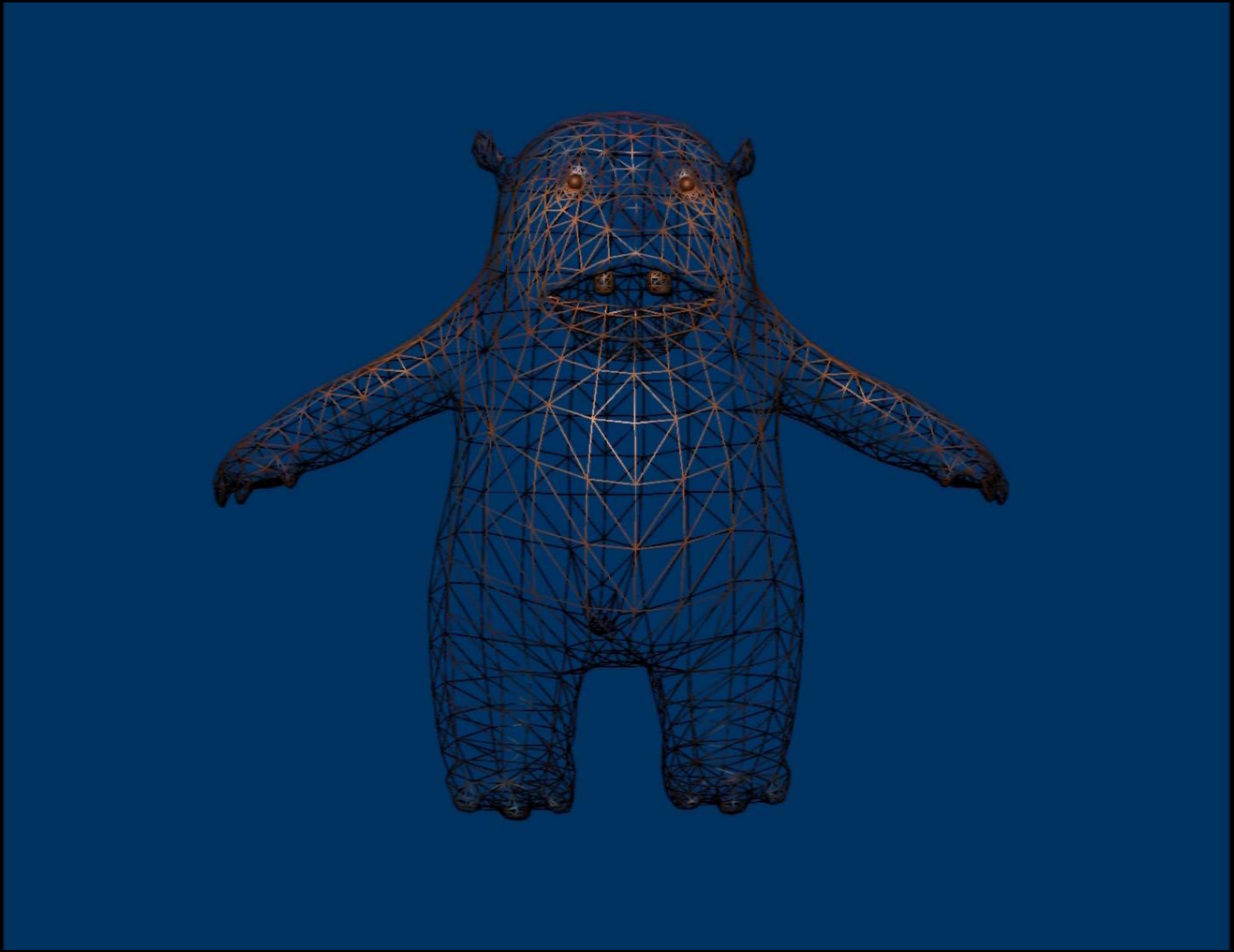
	Poly	Memory	Performance (static)	Performance (animated)
High Res	40,000 triangles	10,936 KB	378 fps	265fps
Low Res + Disp Map + ACC	2,544 quads	628 KB + 223 KB = 851 KB	410 fps	290fps

Not yet optimized !

GeForce GTX 460

Faster rendering using less space !

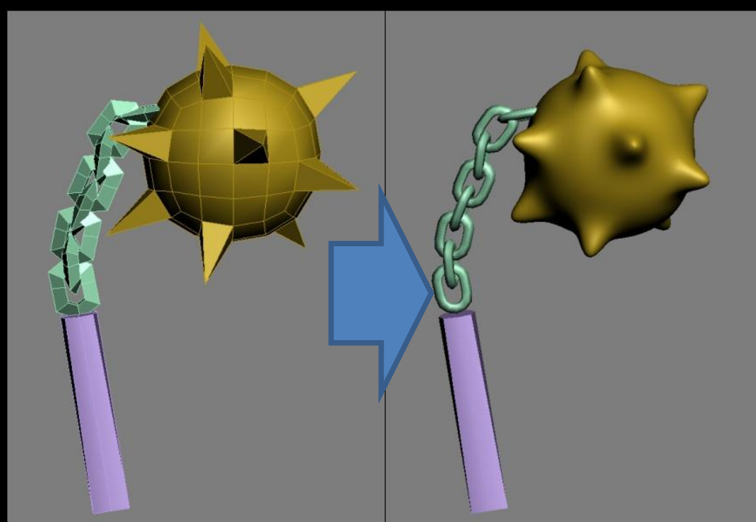
Animated Mesh : Result



How to improve ACC?

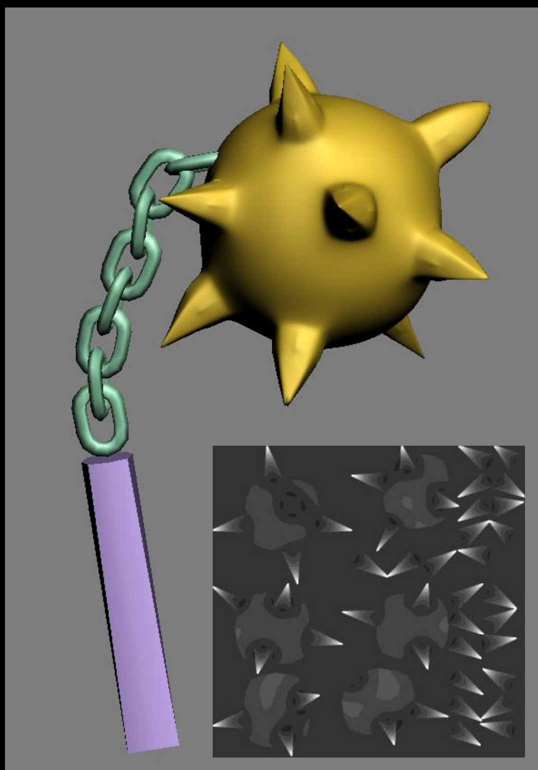
ACC is too smooth

- How to express hard creases?



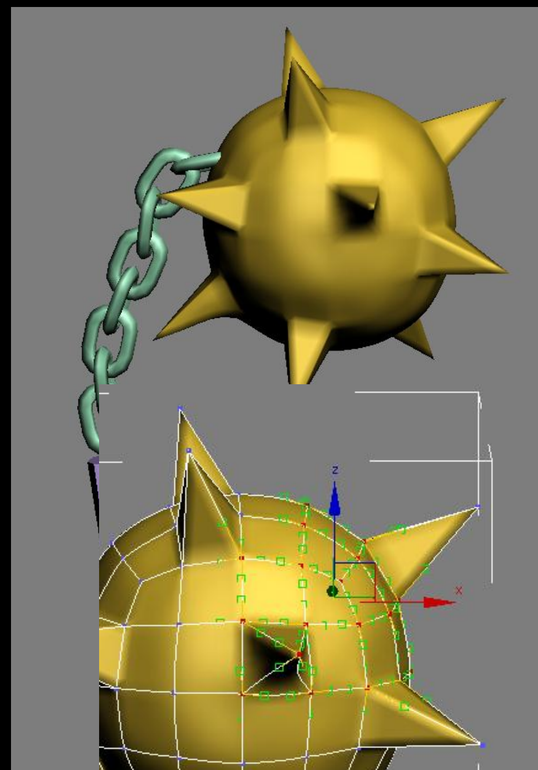
Various methods

Displacement



Artifacts.. memory

Patches



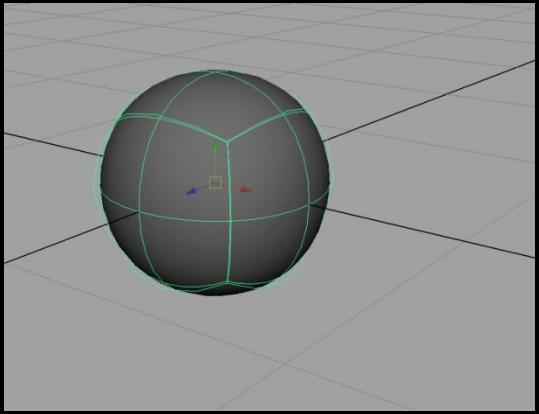
Hard to use..

Add polygons

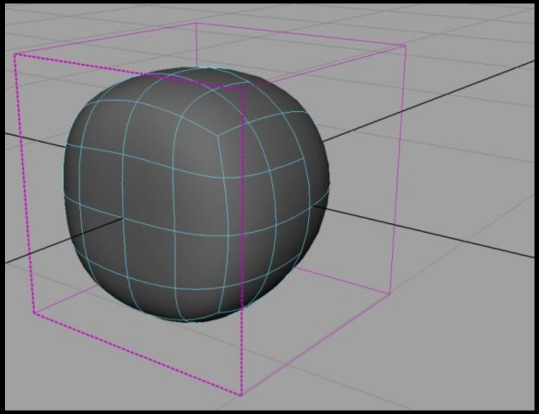
Not very scale-able..

Solution = Creasing

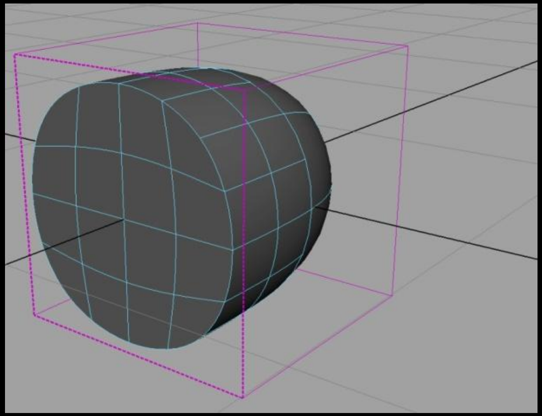
- Give a “weight” to edges
- Possible to control “sharpness”
- Used in Maya



none



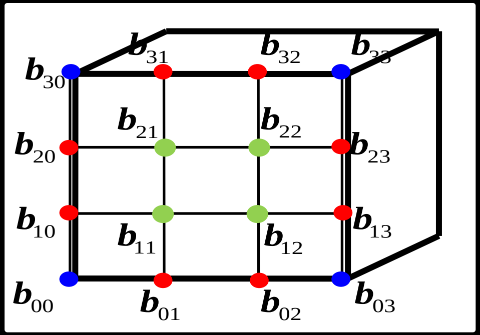
partial



full

Crease algorithm

•Offline: give a weight to creased edges
•Hull Shader:
 for each control point, get the 3 highest crease values of incoming edges: C1 > C2 > C3



•*(C1=C2=C3=0)*
 no crease, usual ACC equation
•*(C1>0 , C2=C3=0)*
 1 creased edge. Corner does not move. Only edge points.
•*(C1>0, C2 >0, C3=0)*
 2 creased edges. Corner converges to plan defined by these edges
•*(C3>0)*
 more than 3 creases. Corner converges to original corner

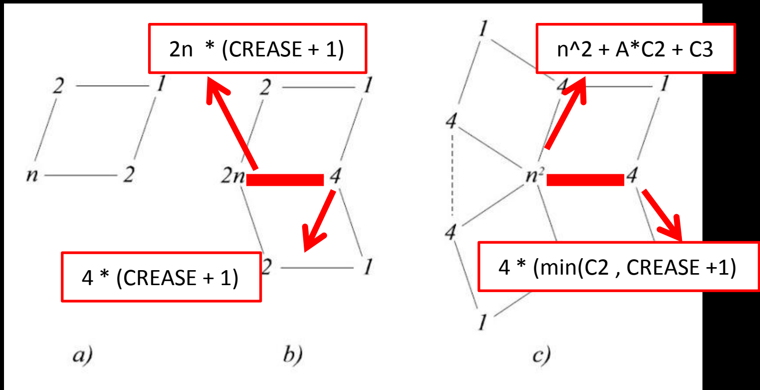
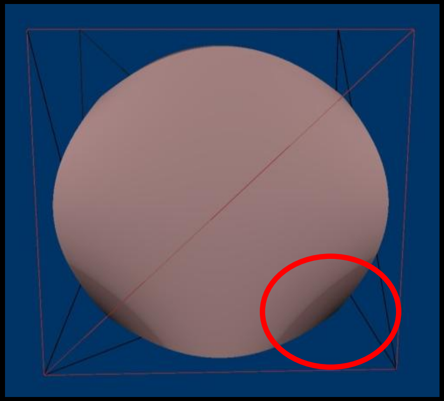
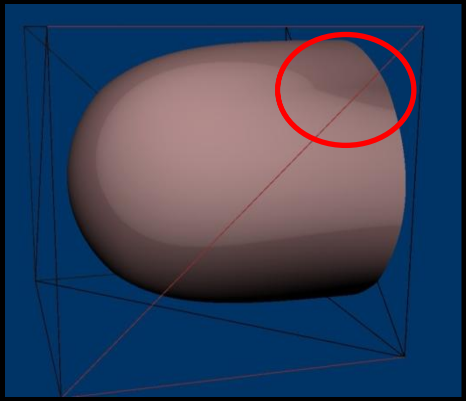
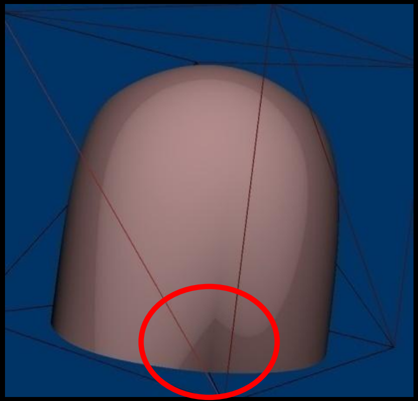


Fig. 4. Generalized masks for interior, edge and corner points.

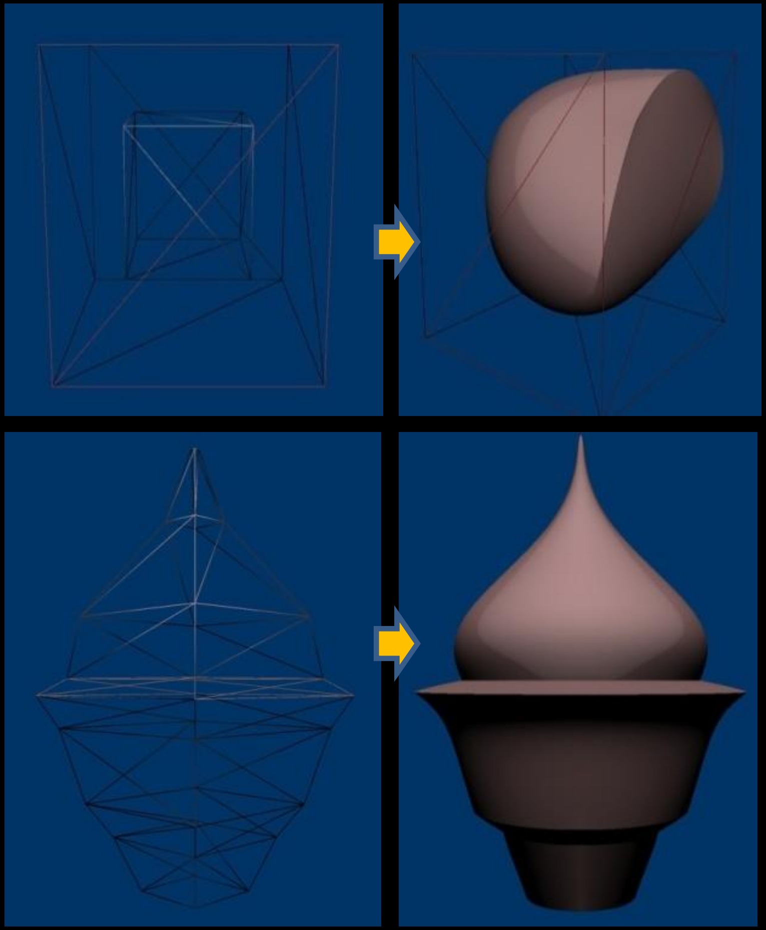
Crease Implementation

- When creasing :
  - Have to keep the normals coherent with geometric changes
  - LERP between original normals and ACC normals

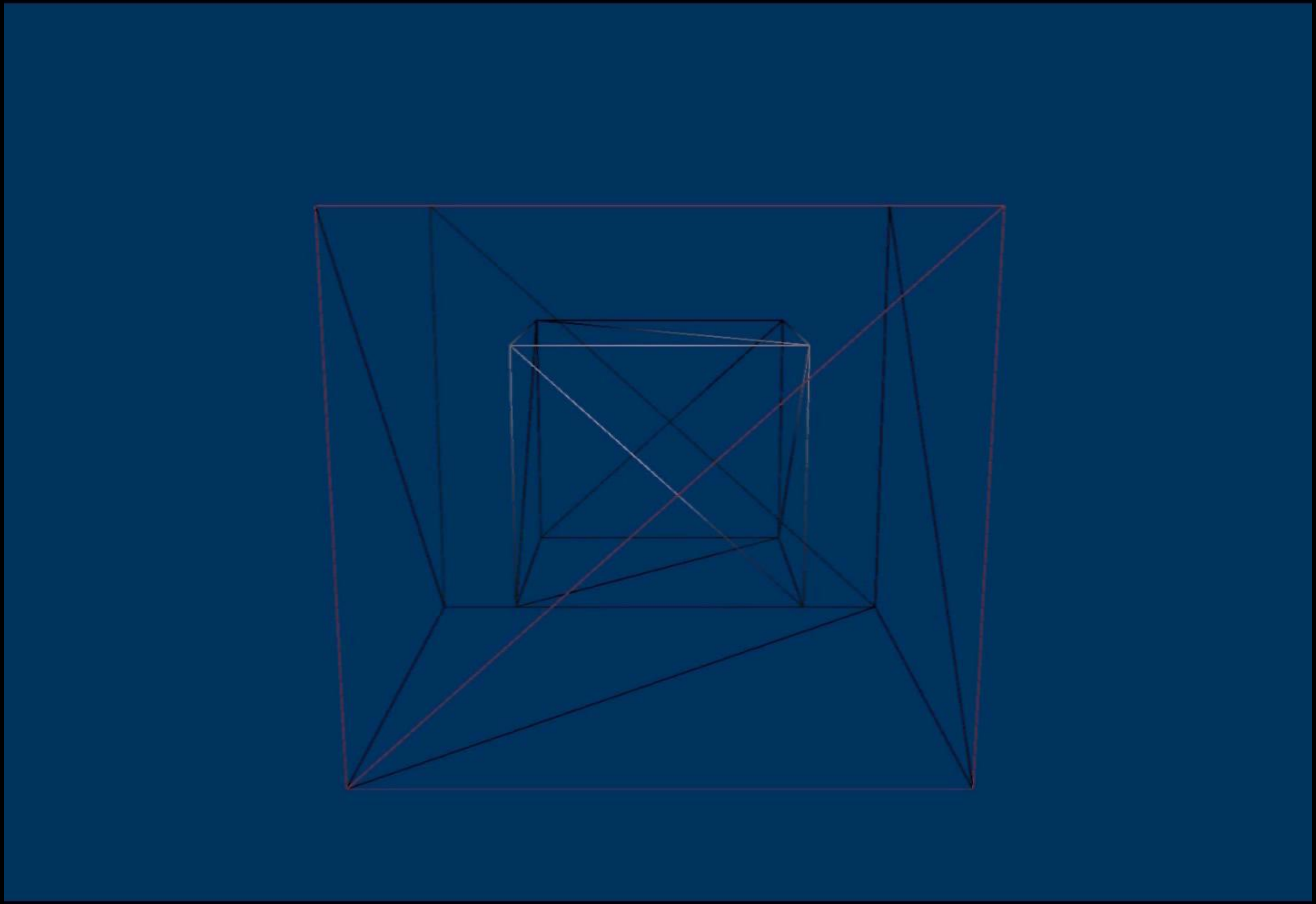




Results

- Merits
  - Easy to produce intermediary shapes
  - No need to change topology
- Demerit
  - Heavy
  - Need to preprocess edges
  - No perfect match with Maya



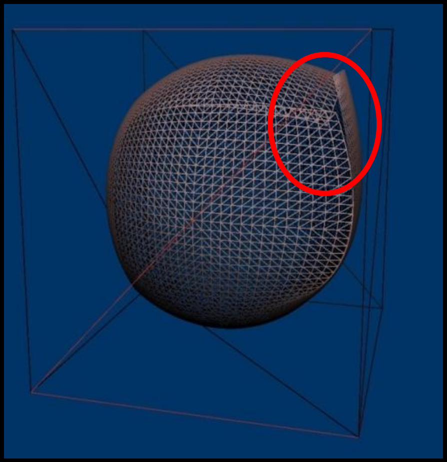
Crease: Results

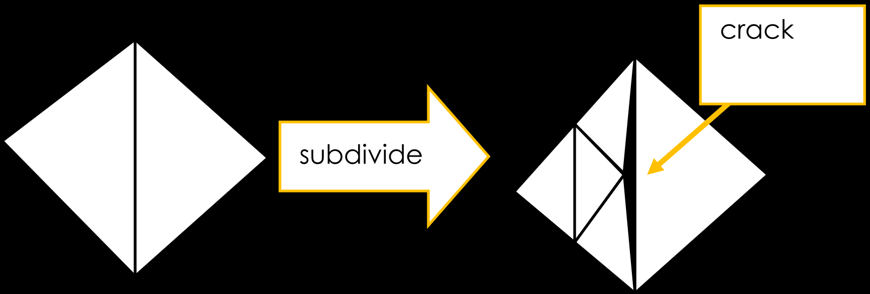


Problems & Difficulties

Basic Problems & Difficulties

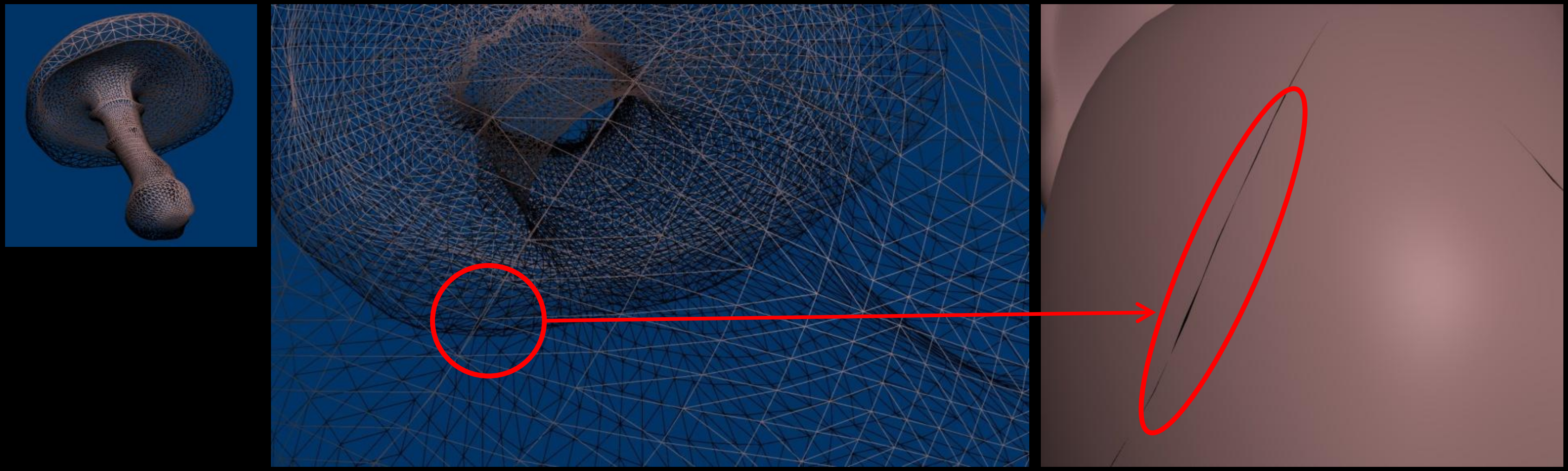
- Difficult to debug huge shaders
  - Output normal field or barycentric coordinates as color
- We don’t use only vertex data, but a whole neighborhood ring
- Have to keep all calculus on patches symmetrical





Displacement cracks

- Discontinuities
  - Displacing with disjoint normals
  - Different displacement values
    - Texture Seam / Sampling error
    - Solution: insure you use the same UV on both sides of edges



Tangent Field

- Discontinuities in the tangent field -> Cracks or artifacts !

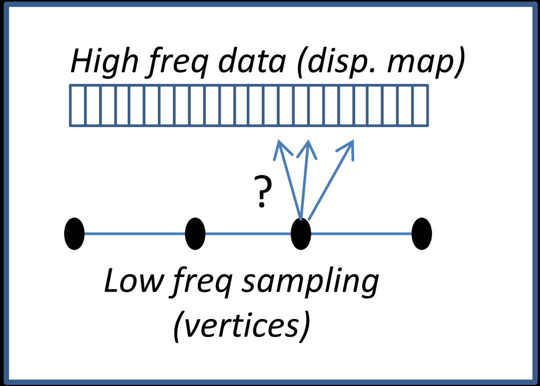

discontinuity

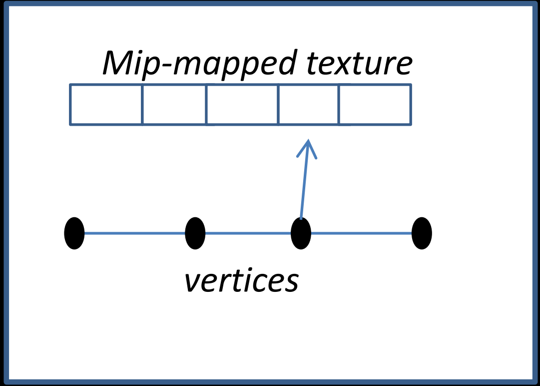

singularity

- Our solution
  – Preprocess the mesh to create a continuous tangent field
  – detect singularities and avoid displacement at those points

Minification aliasing

- Problem
  - Low-frequency sampling of high frequency texture
  - Change tessellation on a displaced mesh = shimmering
- Solution
  - Use mip-mapping

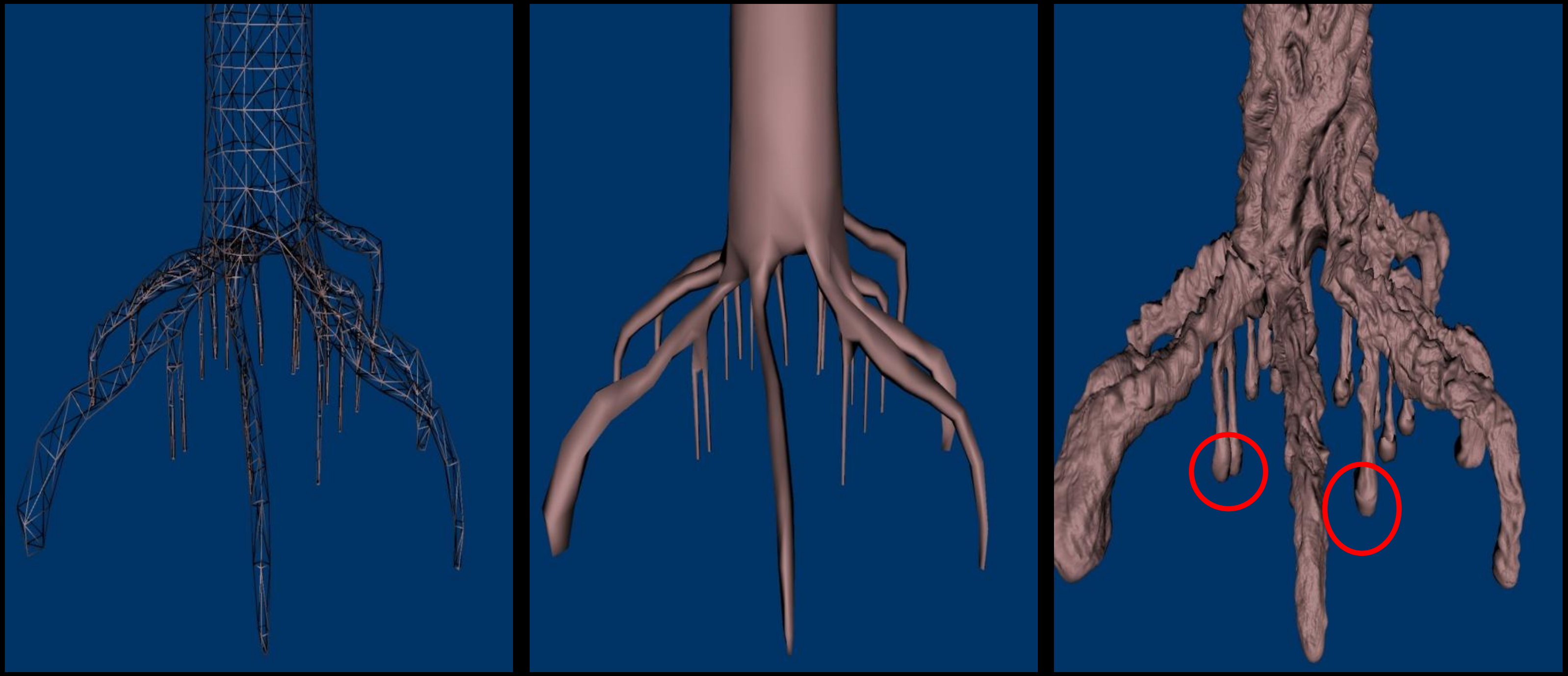




Interesting blog to check out	The problem with tessellation in DirectX 11 Posted by Sebastian Sylvan http://sebastiansylvan.wordpress.com/

Bulging

- Bulging on mesh tips when height-displacing with ACC



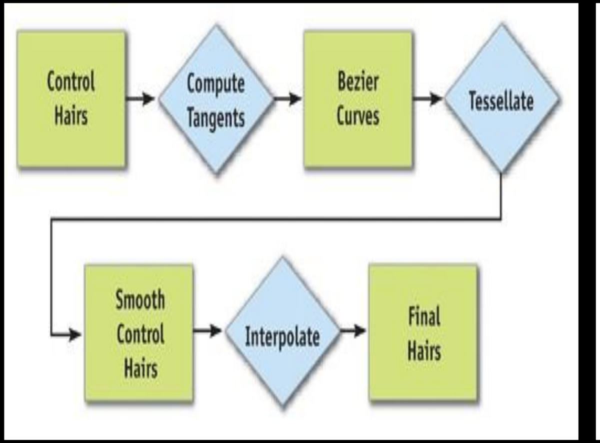
Non-surface tessellation
(lines)

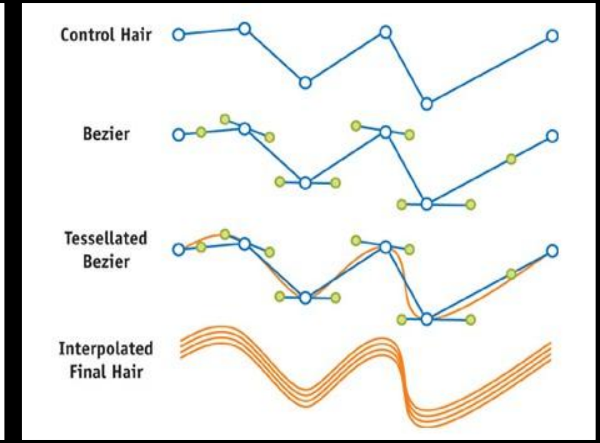
Hairs

- Sarah Tariq @ Nvidia method

NVIDIA Real Time Hair Presentations at Siggraph 2008	http://www.nvidia.com/object/siggraph-2008-hair.html

  - Tessellation on isolines
  - Create large amount of hairs (10,000+)
  - Heavy





Default performance of the demo on GTX 480.

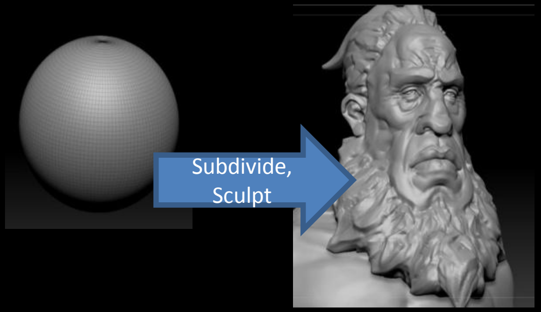
	Using Tessellation Engine and Compute Shader	Not using Tessellation engine and using Dx10 simulation
Hair rendering and simulation with wind	57 fps	44 fps
Just hair rendering, no hair simulation	77 fps	60 fps

Results

Impressions on Tessellation

A lot of merits…

- Performance
  - Faster animation with less data
- Scalability
  - Expand on-demand on GPU
- Visual Quality
  - Perfect smoothness
  - Accurate silhouette
  - Proper occlusion & shadow
- Simpler pipeline
  – subdivide & sculpt !



… and a few "buts"

- ACC is heavy and needs preprocessing
- Need to find a balance
  - Too much tessellation is bad
  - Too few tessellation is a waste

Advice

- Keep the shaders as simple as possible
- Topology matters
- Communicate a lot with the artist
  - E.g. Ask for quad-only models if you use ACC

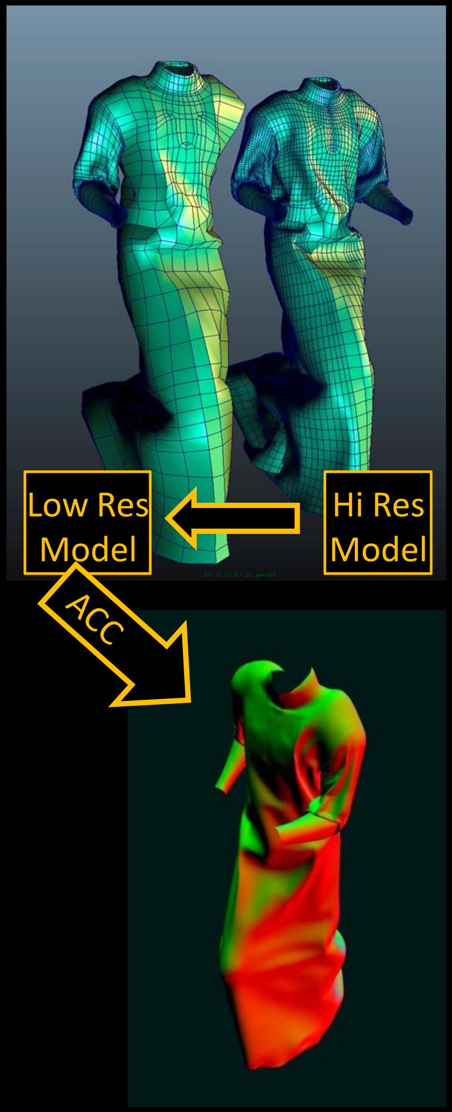
Will likely use the tech:

- For animation (cloth, facial, transformation effects)
- When seamless LOD is necessary

Present & Future work

- Asset from artist
  - Hi-Res Movie Quality Asset (e.g. Cloth)
  - Mesh Reduction & Baked vertex animation (Maya)

- Real-time framework
  - ACC only
  - Even Higher-resolution mesh on screen !



Baked vertex animation Result

PART 2
COMPUTE SHADER

Compute Shader Features

- Structured buffers
- shared memory between threads
- Un-ordered I/O operations
- Atomic operations

Application has control over dispatching and synchronization of threads

⇒ More general algorithms, beyond shading

CS applications
Currently under investigation

- Image processing
  - Post-effect, convolution, SSS, etc…
- Simulation
  - Cloth
  - Particles
- Real-time global illumination

Compute Shaders: Impressions

- CS merits
  - Easy to use
  - Flexible (not limited to graphics)
  - Can provide nice optimizations

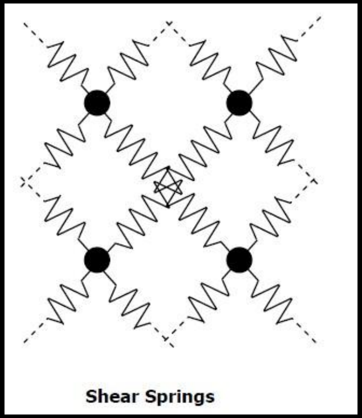
"Efficient Compute Shader programming"
Bill Bilodeau, AMD (*developer.amd.com*)

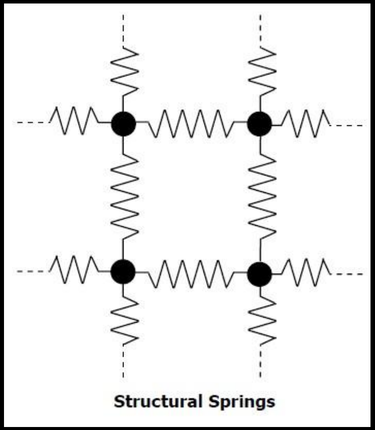
- CS demerits
  - So many things to do: tiled-rendering, SSAO, DOF, Soft bodies, AI, Etc…

Cloth Simulation

Basic approach

- Cloth is modeled as a set of particles
- Each particle is subject to:
  - External forces: gravity, wind,..
  - Internal “spring” constraints
  - collision constraints





Basic approach

- Verlet integration of particles' motion

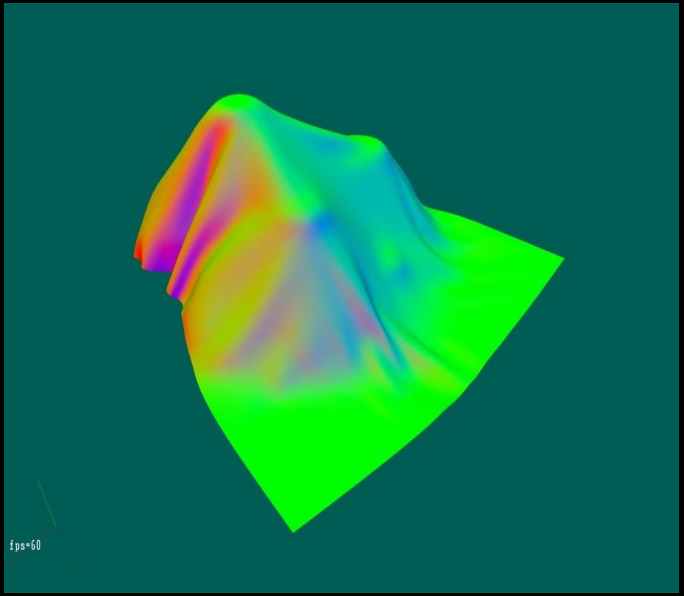
```
newx = x + (x - oldx)*damping + a*dt*dt
```

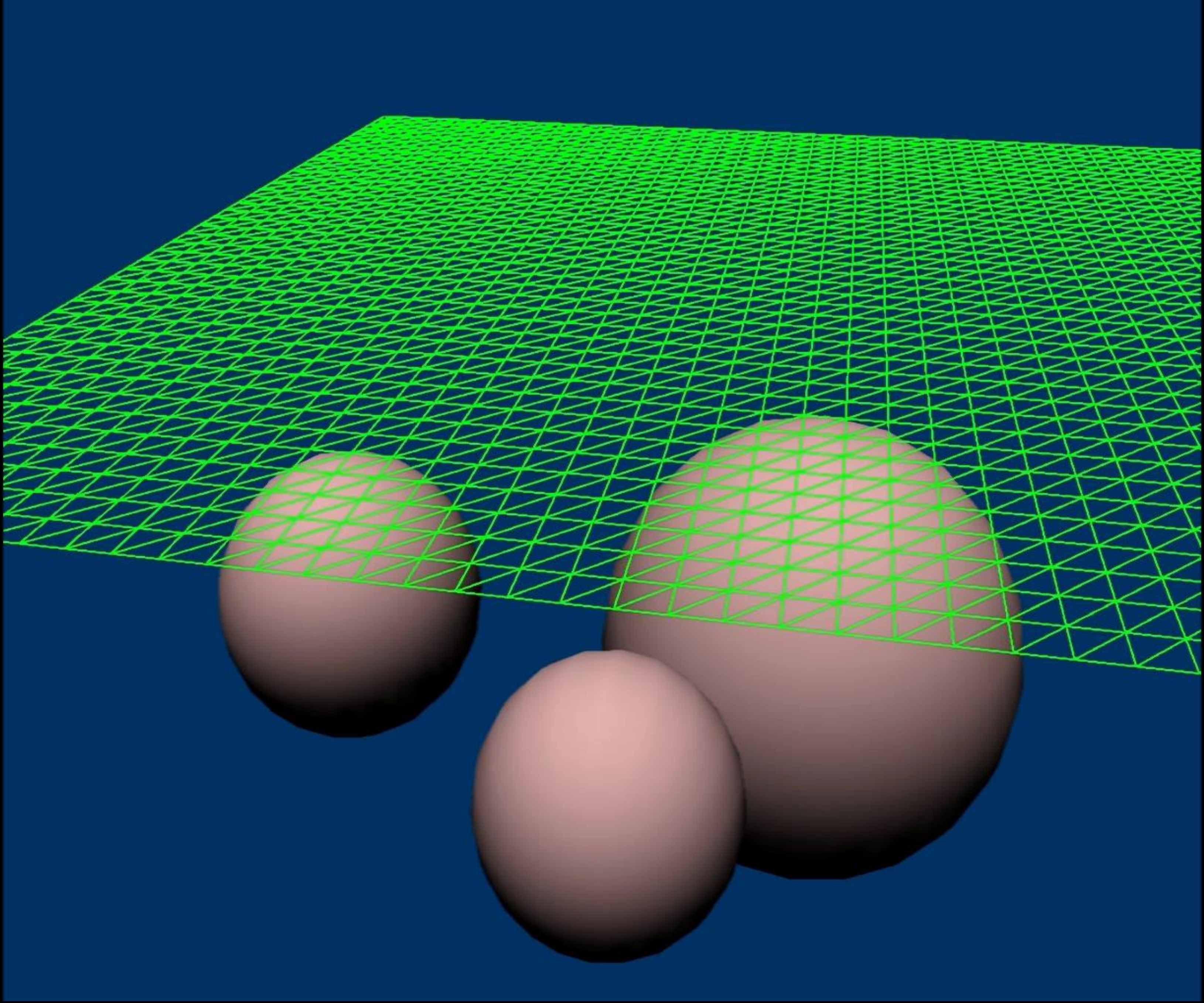
- Not always accurate, but stable!
- Constraints solved by relaxation, using a given number of iterations.

"Cloth Simulation, White Paper". NVidia
"OpenCL Cloth Simulation in the Bullet Physics SDK" [Lee Howes] AMD

Results

- Physical simulation at 30Hz
- rendering with ACC
  - 40x40 cloth & TessFactor=8 ＞ 40*40*64 = 102,400 quads on screen
  - 200+ fps





Particles
パーティクル

Next generation particle system
次世代のパーティクル・システム

- Goal
  - 100 000 particles and more
  - very small particles
  - Update, create geometry on GPU
  - Self-shadowing and scattering
- Challenges
  - Random numbers on GPU
  - Fast sorting

- 目標
  - 10万個以上
  - 極めて小さい
  - GPUでアップデートとジオメトリの生成
  - セルフシャドウ＋スカッタリング
- 挑戦
  - GPU上での乱数
  - 速いソート

Our solution
我々のソリューション

- Random numbers on GPU:
  – Tiny Encryption Algorithm [Zafar10]

- GPU上での乱数
  – Tiny Encryption Algorithm [Zafar10]

[Zafar10]	"GPU Random Numbers via the Tiny Encryption Algorithm" F. Zafar, et al.

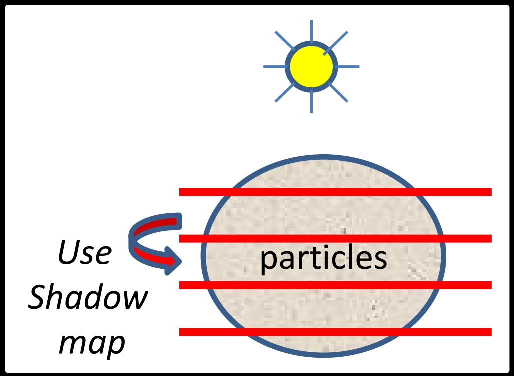
- Sort
  - "Bitonic sort"
  - not satisfied
  - In future: hybrid radix/merge sort

- ソート
  - "Bitonic sort"
  - 不十分
  - 将来：radix/mergeのハイブリッドソート

Our solution
我々のソリューション

- Self-shadowing
  - inspired by [Swoboda11]
  - Split in slices by depth from light source
  - Render slice N to a shadow map
  - Use that result to shadow particles in slice N+1

- セルフシャドウ
  - [Swoboda11]を参考
  - ライトから**奥行き**、複数のスライスに空間を割る
  - スライスNを、シャドウマップに描画
  - スライスN+1を描画するとき、そのシャドウマップを利用



[Swoboda11]	http://directtovideo.wordpress.com/

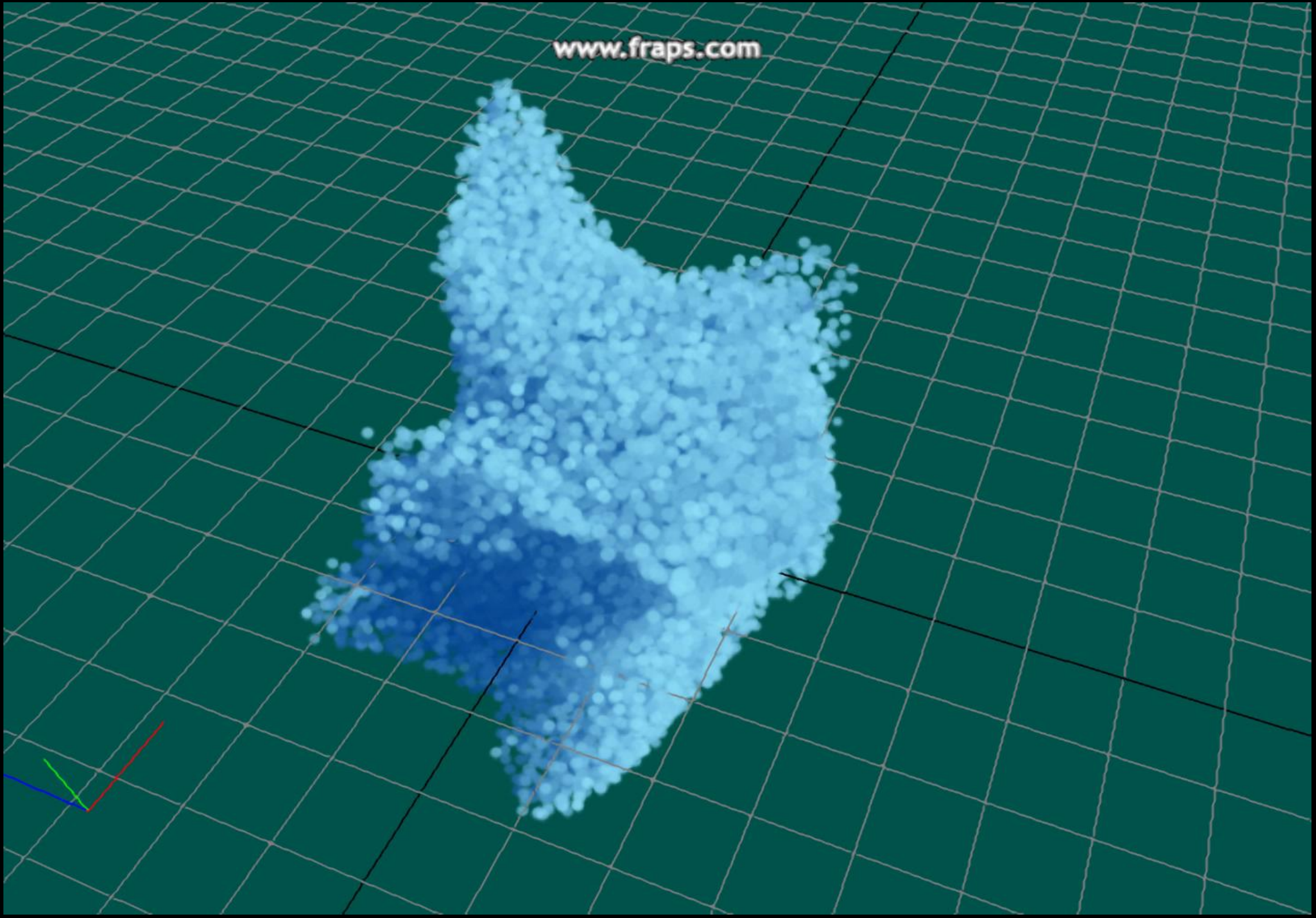
Fluid-like movement
流体

- Possible approaches:
  - Grid based fluid simulation (Stam's stable fluids)
    - Slow in 3D
  - Smoothed Particle Hydrodynamics
    - Some stability problems, not very good for smoke
- Our Choice:
  - Curl Noise on GPU [Bridson07]
  - Fake, but nice shapes and very fast

- 可能なアプローチ
  - グリッドに基づくシミュレーション (Stam's stable fluids)
    - 3次元なら遅い
  - 「Smoothed Particle Hydrodynamics」
    - 不安定
    - 煙に不向き
- 我々の選択:
  - 「Curl Noise on GPU」[Bridson07]
  - 偽物だが、きれい
  - 速い

[Bridson07]	"Curl noise for procedural fluid flow", R. Bridson, J. Hourihan, and M. Nordenstam, Proc. ACM SIGGRAPH 2007

Results



Reflections
リフレクション

Dynamic Local Reflections
ローカルな動的リフレクション

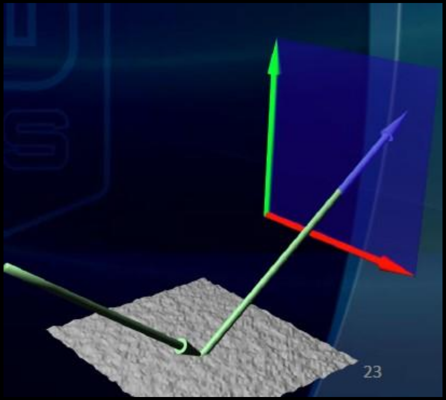
- Planar reflections
  - Reflected object: arbitrary
  - Receiver object: objects facing one general direction
- Cubemap
  - Heavy

- 平面リフレクション
  - 反射される方：なんでも
  - 反射する方：大体同じ方向に向いている平面的なオブジェクト
- キューブマップ
  - 重い

Dynamic Local Reflections
ローカルな動的リフレクション

- Billboard Reflection
  - “Epic’s Samaritan Demo”
  - Reflected object: billboard
  - Receiver object: arbitrary
  - Isotropic/Anisotropic Reflection

- ビルボード・リフレクション
  - Epicの「Samaritan」デモ
  - 反射される方：ビルボードだけ
  - 反射する方：なんでも
  - アイソトロピック・アニソトロピックリフレクションが可能



From “The Technology Behind the DirectX 11 Unreal Engine"Samaritan" Demo”

Dynamic Local Reflections
ローカルな動的リフレクション

- Future: Try screen space local reflections from Crysis 2
- 将来：Crysis２のスクリーン・スペース・ローカル・リフレクション

From “Crysis 2 DX11 Ultra Upgrade”

“Reflected-Scene Impostors for Realistic Reflections at Interactive Rates”, V. Popescu et al.
“One-Shot Approximate Local Shading”, L. Wang et al.
Game Developer Conference 2011. The Technology Behind theDirectX 11 Unreal Engine"Samaritan" Demo
“Crysis 2 DX11 Ultra Upgrade” www.mycrysis.com/sites/default/files/support/download/c2_dx11_ultra_upgrade.pdf

Results

Conclusion

A new project

Many things to do and techs to test!

Soon we will have real assets!

Next year will be awesome !

Q&A

Remi

Ivan

Iwata

Tech

Art