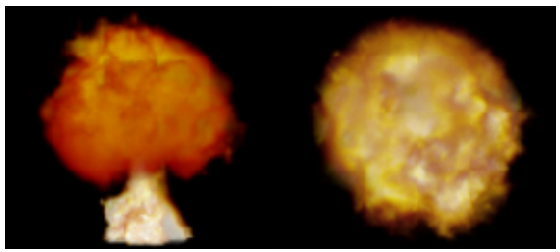
Dynamic Geometry Displacement

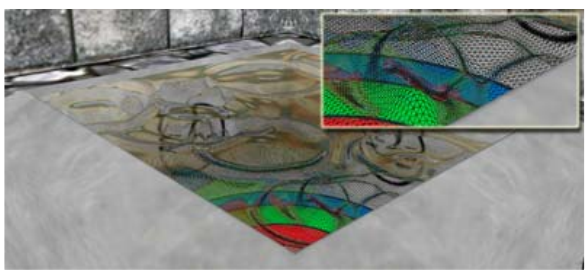
Jens Krüger
Technische Universität München

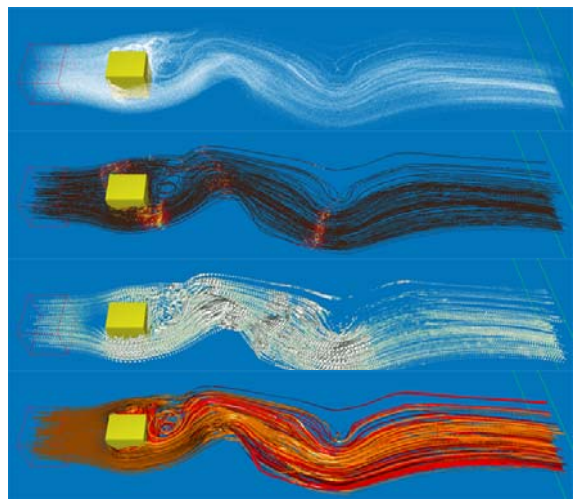
Geometry processing on GPUs

- so far: GPGPU limited to texture output



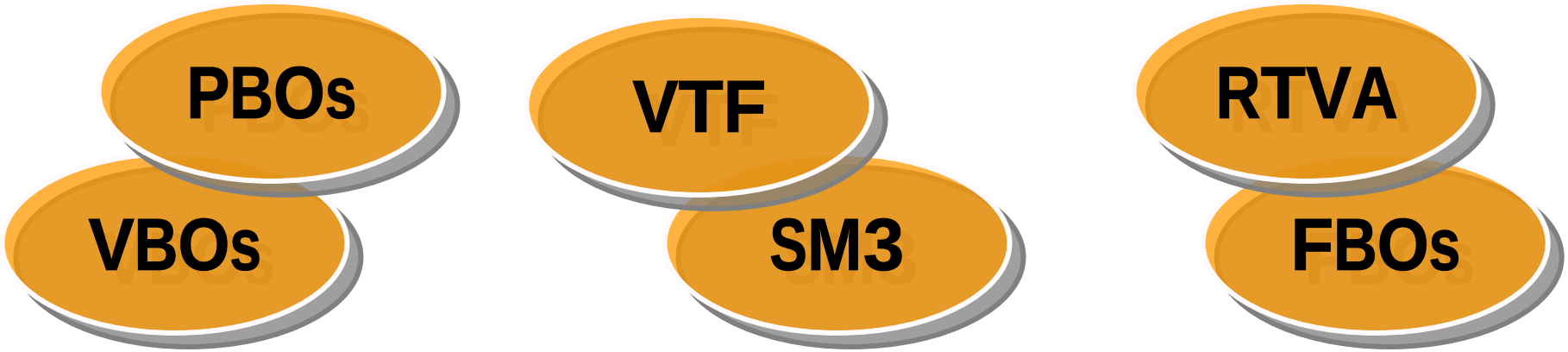
- new APIs allow geometry generation on GPU





GPGPU

API Bubbles

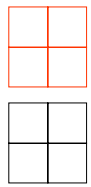


- Shader Model 3.0 with Vertex-Texture-Fetch (VTF)
- Pixel-Buffer-Objects (PBO)
  to Vertex-Buffer-Objects (VBO) copy
- Frame-Buffer-Object (FBO)
  with Cast/Render-To-Vertex-Array (RTVA)
  extension

PBO, VBO Copy

You need:

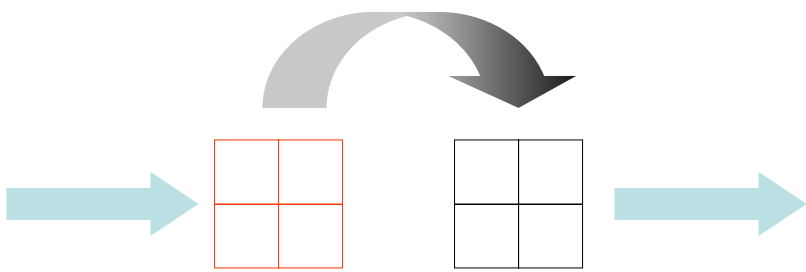
- **floating point pbuffers**
- **a VBO** *(glGenBuffersARB)*



Application Loop:

- **Write geometry data into a pbuffer**

- **Copy data from pbuffer to VBO** *(glReadPixels)*
- **Set vertex array pointers to the VBO** *(glBindBufferARB, glVertexAttribPointerARB)* *and* **Render geometry as usual** *(glDrawArrays, glDrawElements)*



GPGPU

FBO cast to Vertex-Array

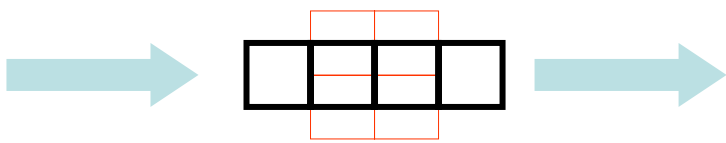
You need:

- floating point FBOs that support RTVA 

Application Loop:

- Write geometry data into the FBO

- Bind FBO as Vertex-Array
- Render geometry as usual
  (glDrawArrays, glDrawElements)

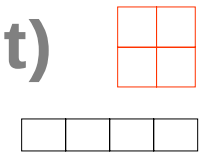


GPGPU

SM3 with VTF

You need:

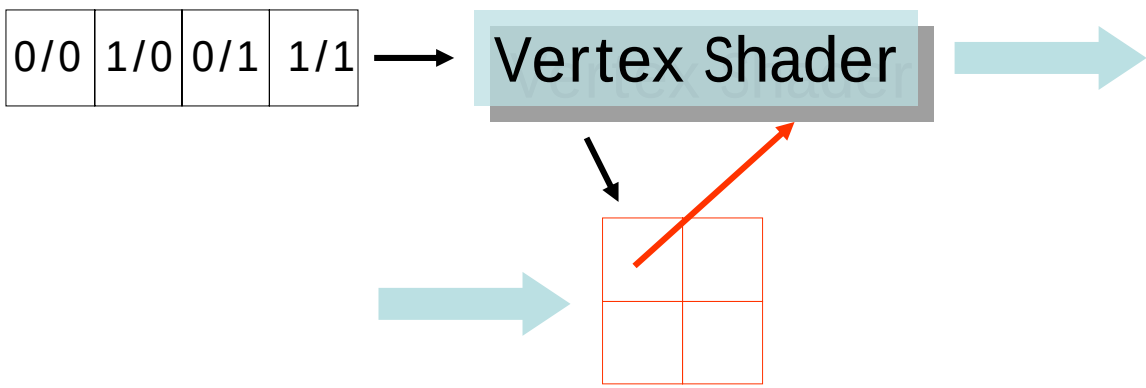
- render targets (pbuffer, FBO, DX render target)
- a static Vertex-Buffer/ Vertex-Array



Application Loop:

- Write geometry data into the render target

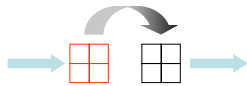
- Render the static Vertex-Buffer enable a vertex shader that fetches from the render target



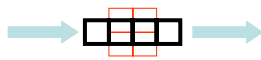
GPGPU

Comparison

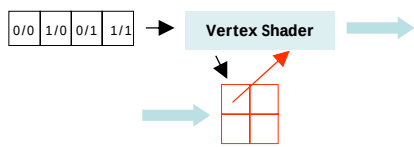
- **PBO, VBO Copy**
  - availability
  - requires memcopy , OpenGL only



- **FBO cast to Vertex-Array**
  - no memcopy
  - availability, OpenGL only

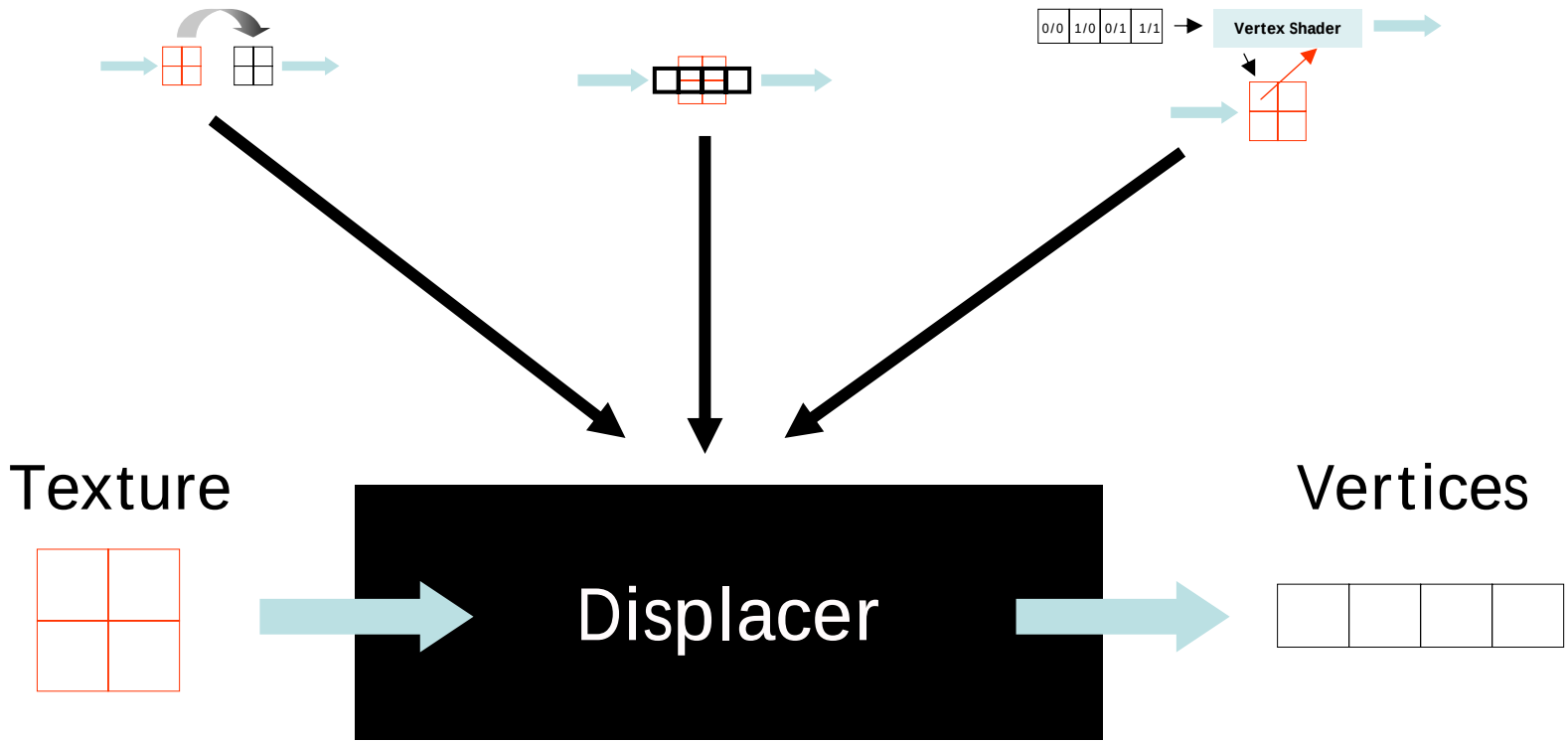


- **SM3 with VTF**
  - multi vendor, OpenGL/DirectX, flexible
  - more code, requires latest GPU



GPGPU

… it's all the same somehow …



In principle, all 3 methods take one or more textures as input and displace a vertex stream accordingly, this makes them interchangeable.

GPGPU

Examples

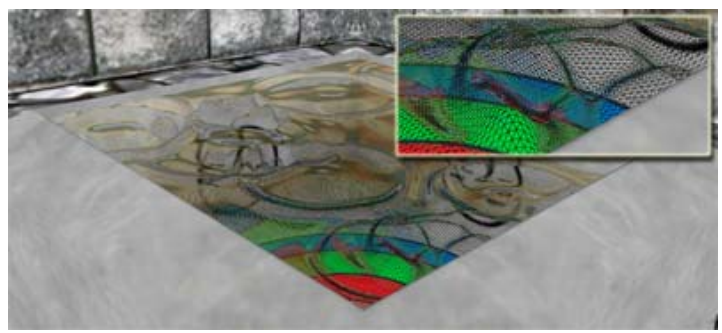
Fluid Simulation

Particles

3D Smoke & Fire

Water Simulation

Grid displacement



3D Water Surfaces

GPGPU

Examples

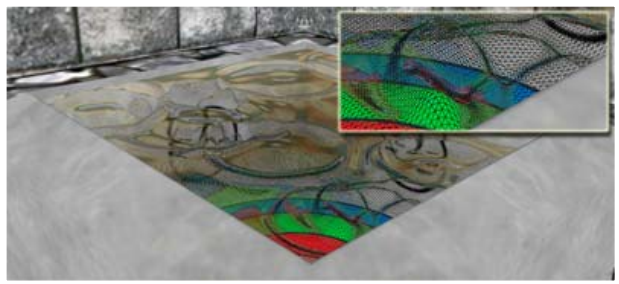
Fluid Simulation

Particles

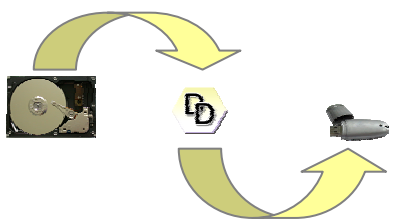
3D Smoke & Fire

Water Simulation

Grid displacement



3D Water Surfaces



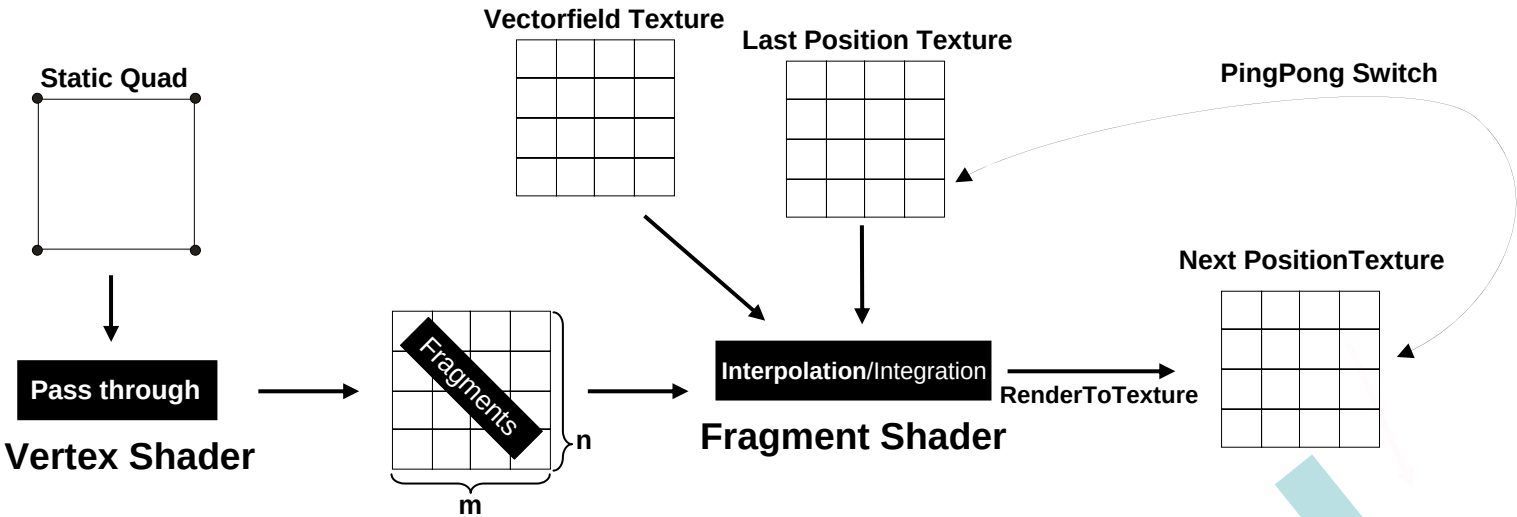
Point Compression

Point Decompression

Point Rendering

GPGPU

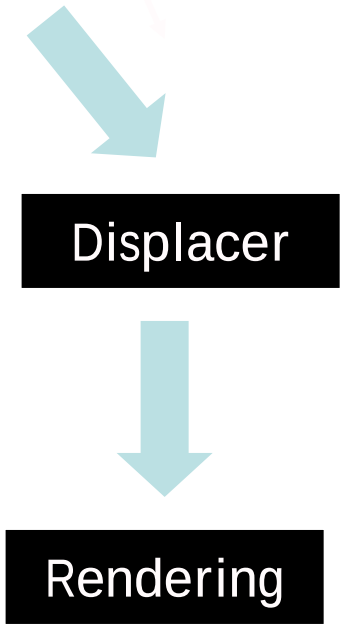
Example 1, Particle Tracing



Observed Speedups for 3D Vector fields

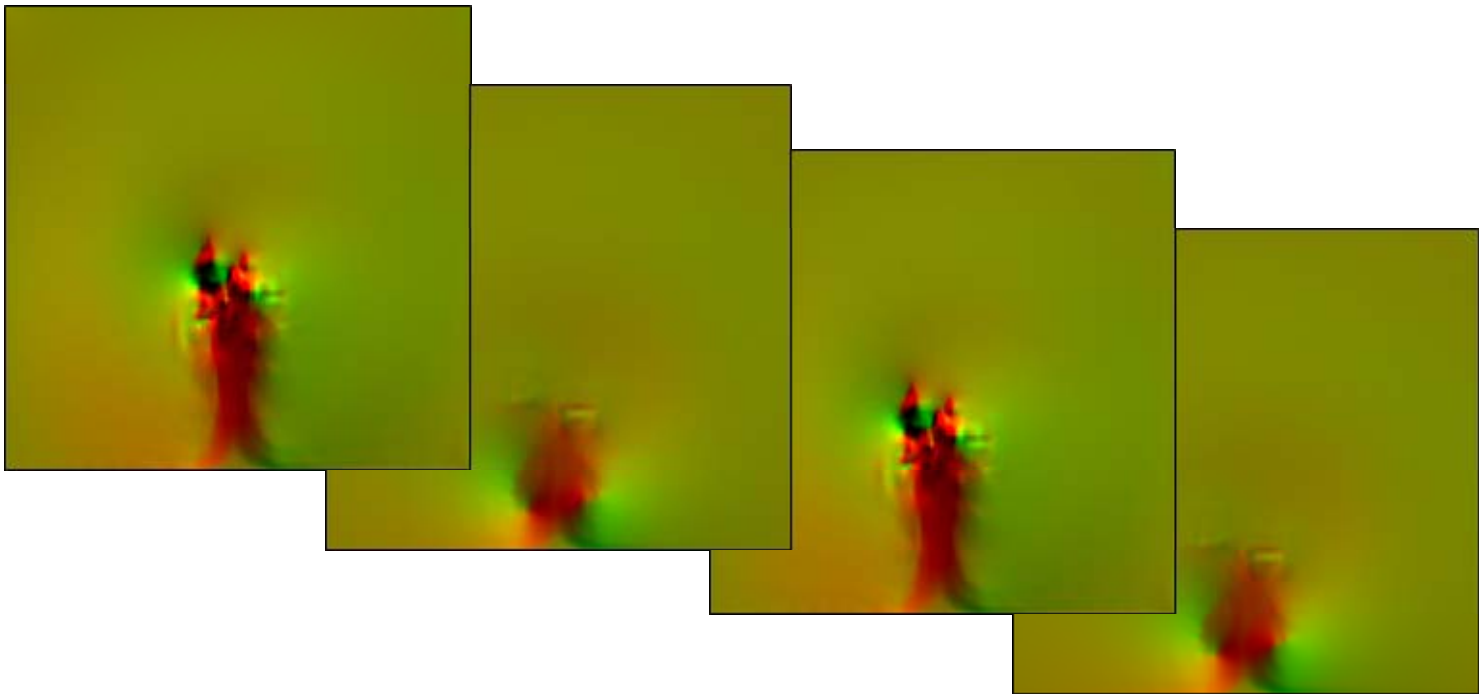
- Interpolation (x100)
- SIMD Integration (x20)
- SM 3.0 Particle Rendering (x20)

ATI X800 XT vs. Pentium 4 3Ghz



GPGPU

100+ fps Smoke with GPU Particles I

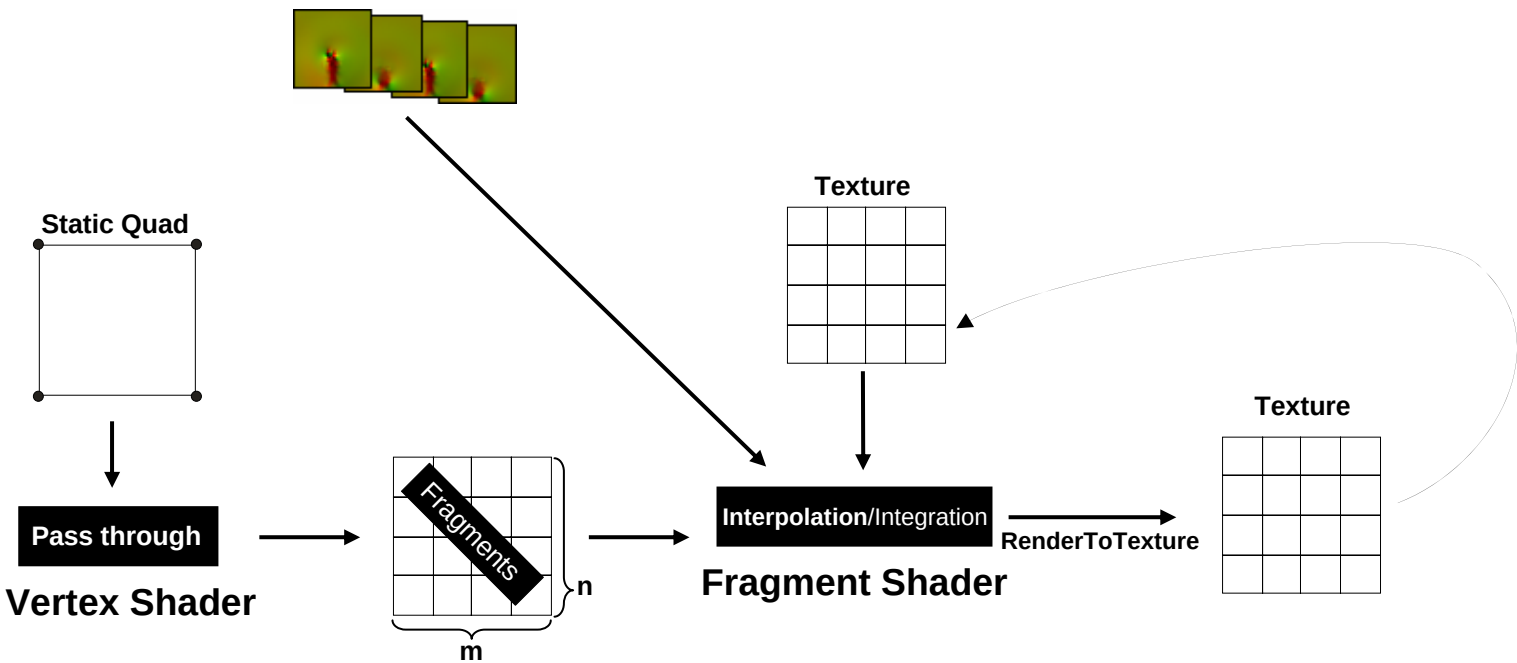


Step 1:

- **do a couple of independent 2D fluid simulations**
- **run at a few hundred FPS on current GPUs**

GPGPU

100+ fps Smoke with GPU Particles II



Step 2:

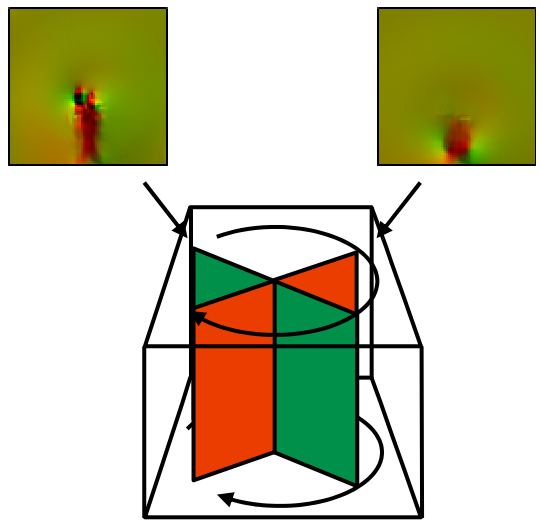
- Bind the output of the simulation as vector field textures for the particle advection

GPGPU

100+ fps Smoke with GPU Particles III

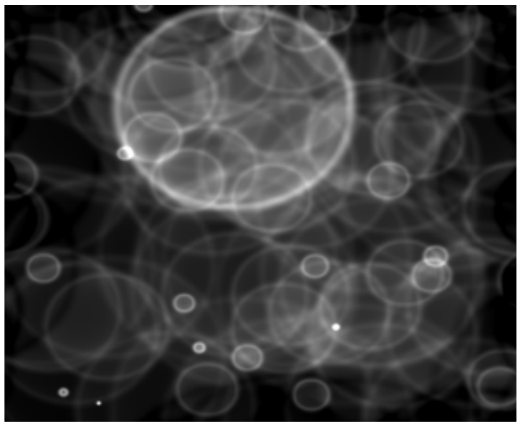
Step 3: Let the interpolator compute the revolution

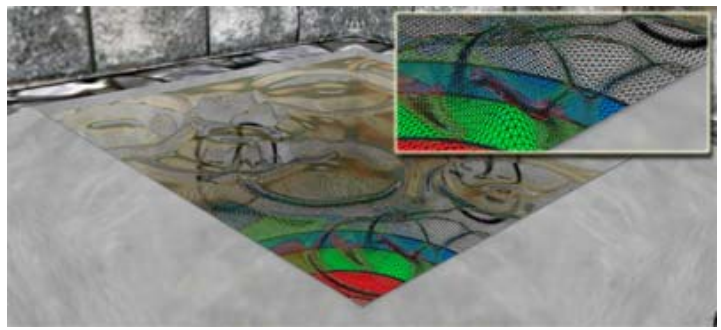
```
float3 sample2DNSExtr(float3 position) {
  float4 translPosition;
  translPosition.xy = position.xz*2-1;
  translPosition.zw = float2(-translPosition.y,translPosition.x);
  float l = length(translPosition.xy);
  float4 f4TexCoords = float4( (sign(translPosition.x)*l+1.0f)/2.0f, position.y,
                               (sign(translPosition.z)*l+1.0f)/2.0f, position.y);
  float2 f2SimulationA = tex2D(sVFieldA,f4TexCoords.xy).xy;
  float2 f2SimulationB = tex2D(sVFieldB,f4TexCoords.zw).xy;
  translPosition  /= l;
  float3 f3SimulationA, f3SimulationB;
  f3SimulationA.y  = f2SimulationA.y;
  f3SimulationA.xz = translPosition.xy*f2SimulationA.x*sign(translPosition.x);
  f3SimulationB.y  = f2SimulationB.y;
  f3SimulationB.zx = translPosition.zw*f2SimulationB.x*sign(translPosition.z);
  f3SimulationB.z = -f3SimulationB.z;
  float3 f3Simulation = lerp(f3SimulationA,f3SimulationB,asin(abs(translPosition.z))/1.57f);
  return f3Simulation;
}
```

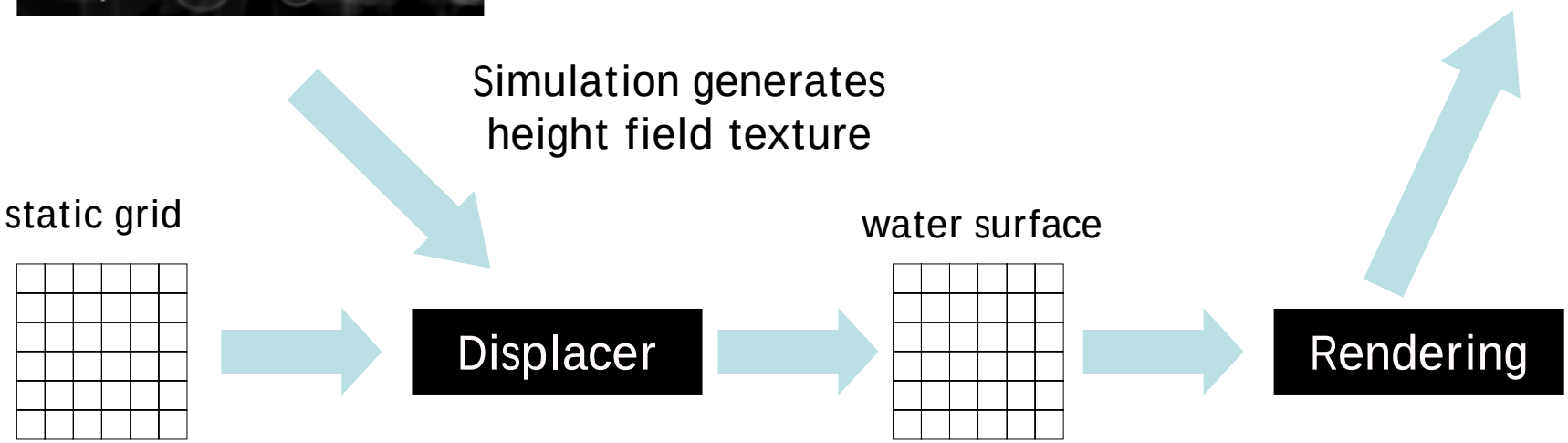


GPGPU

Example 2, Water Surface

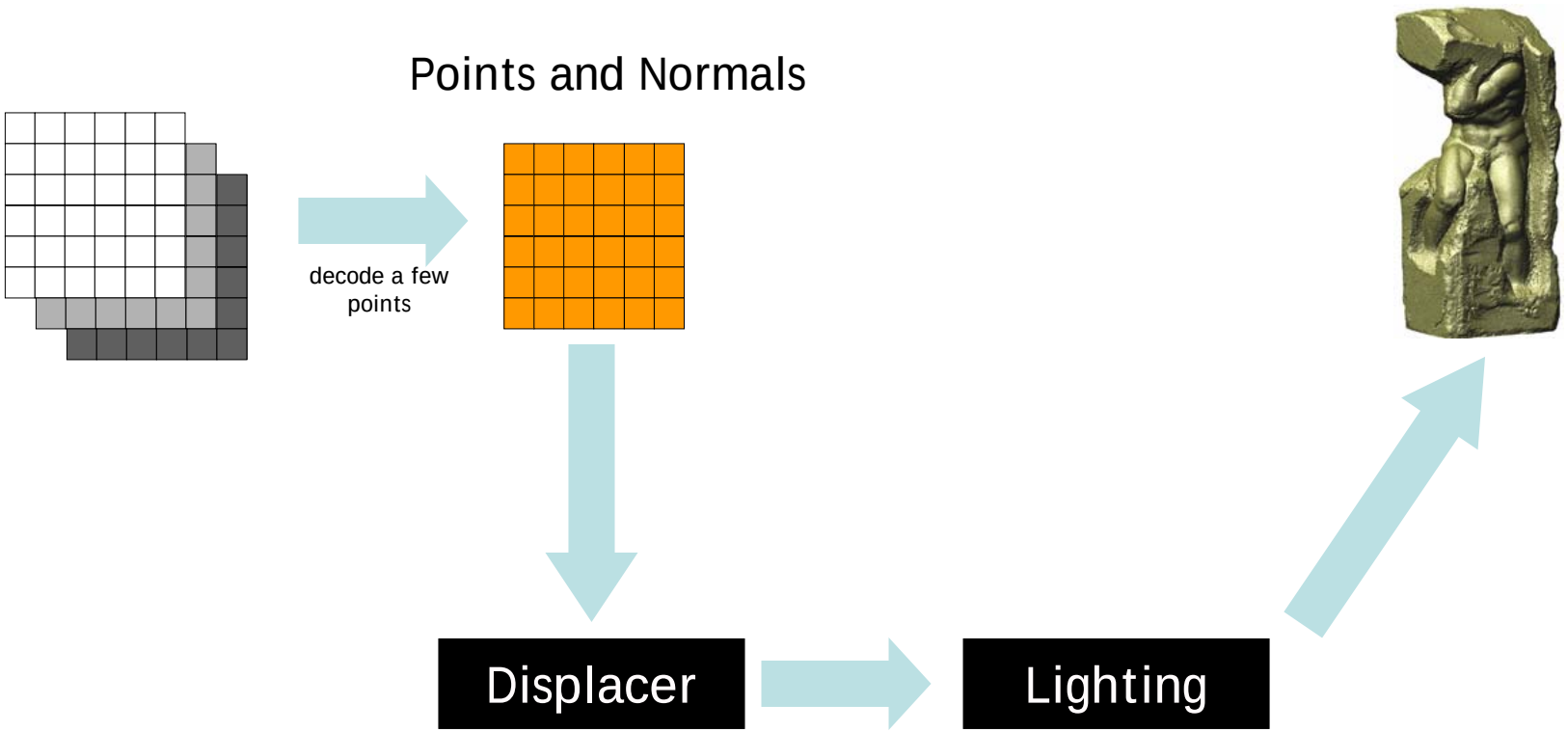




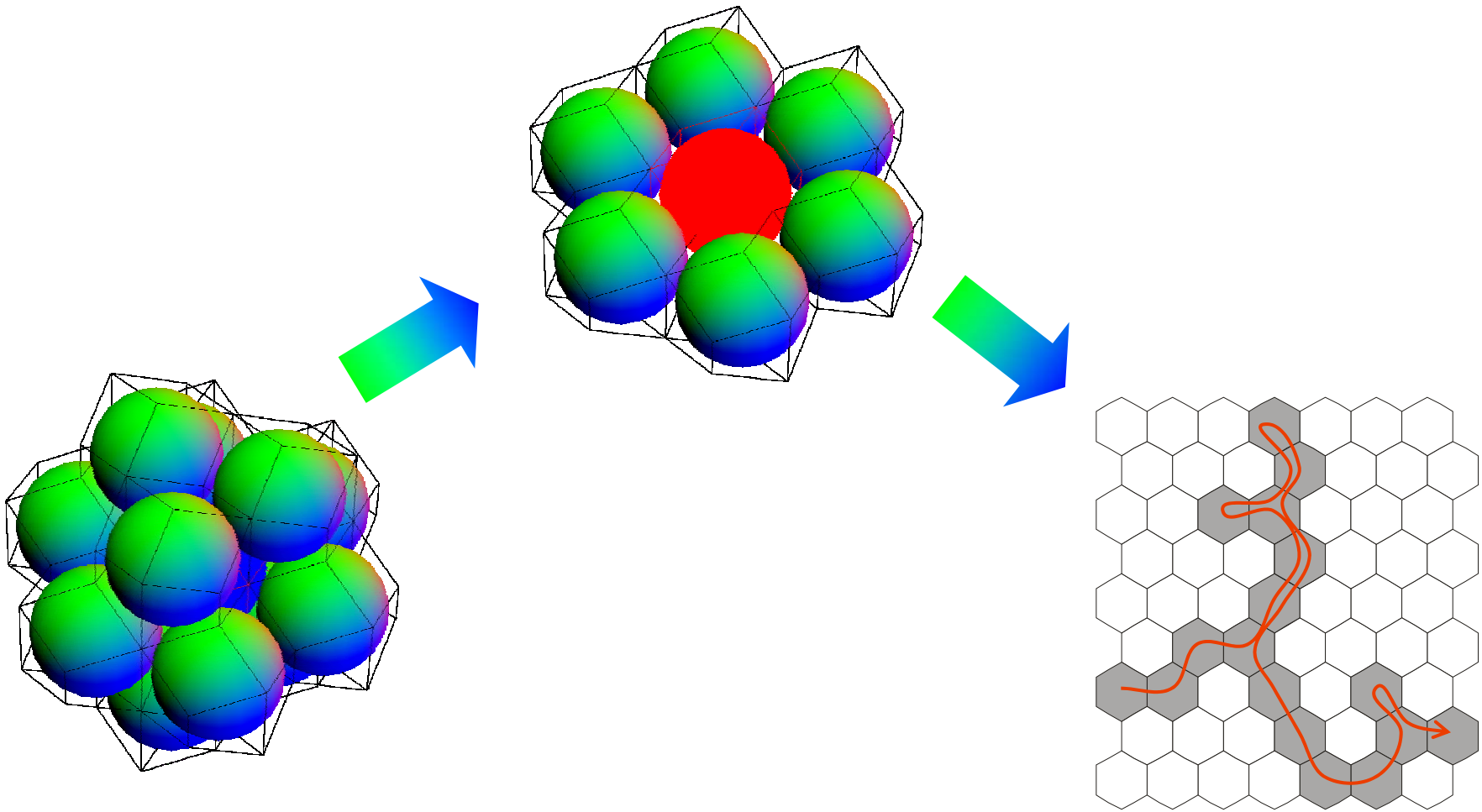


GPGPU

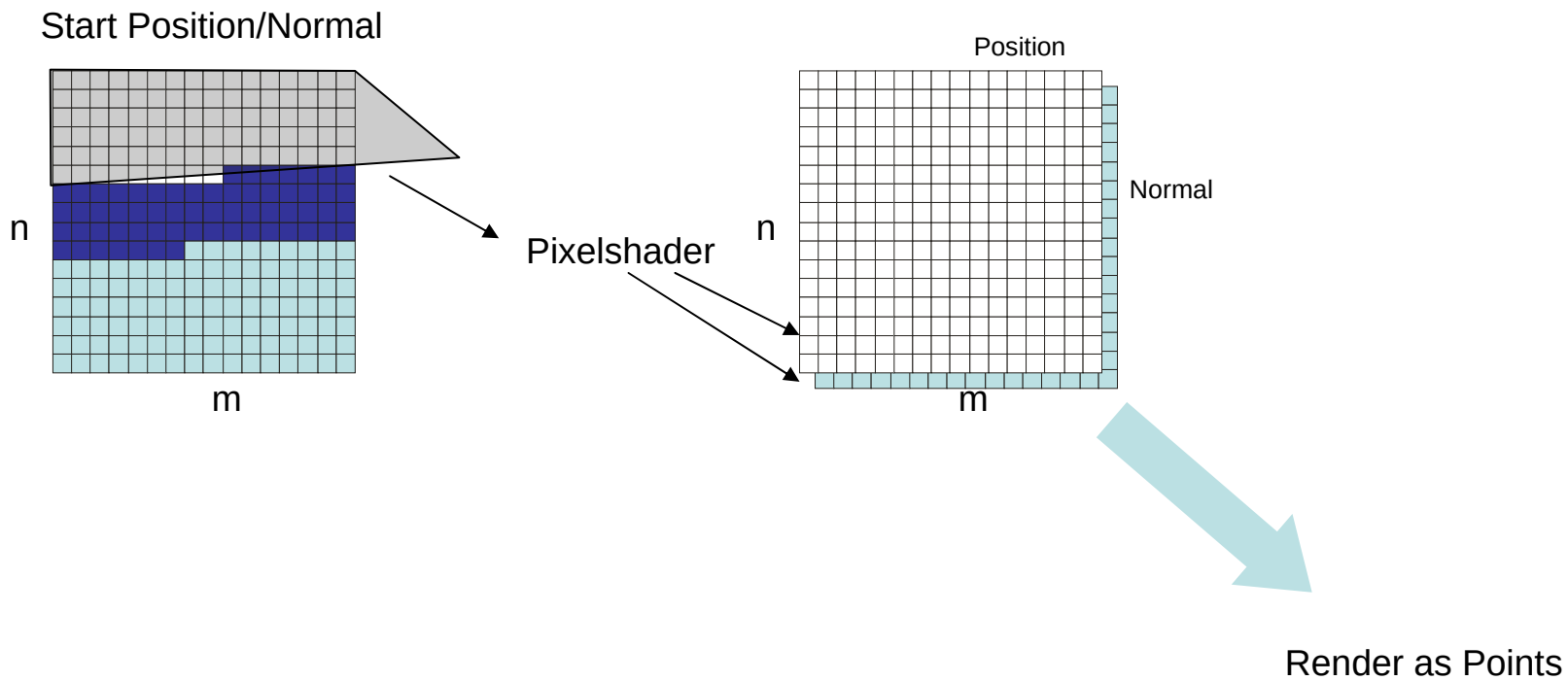
Example 3, Point Compression



Example 3, Point Compression (cont.)

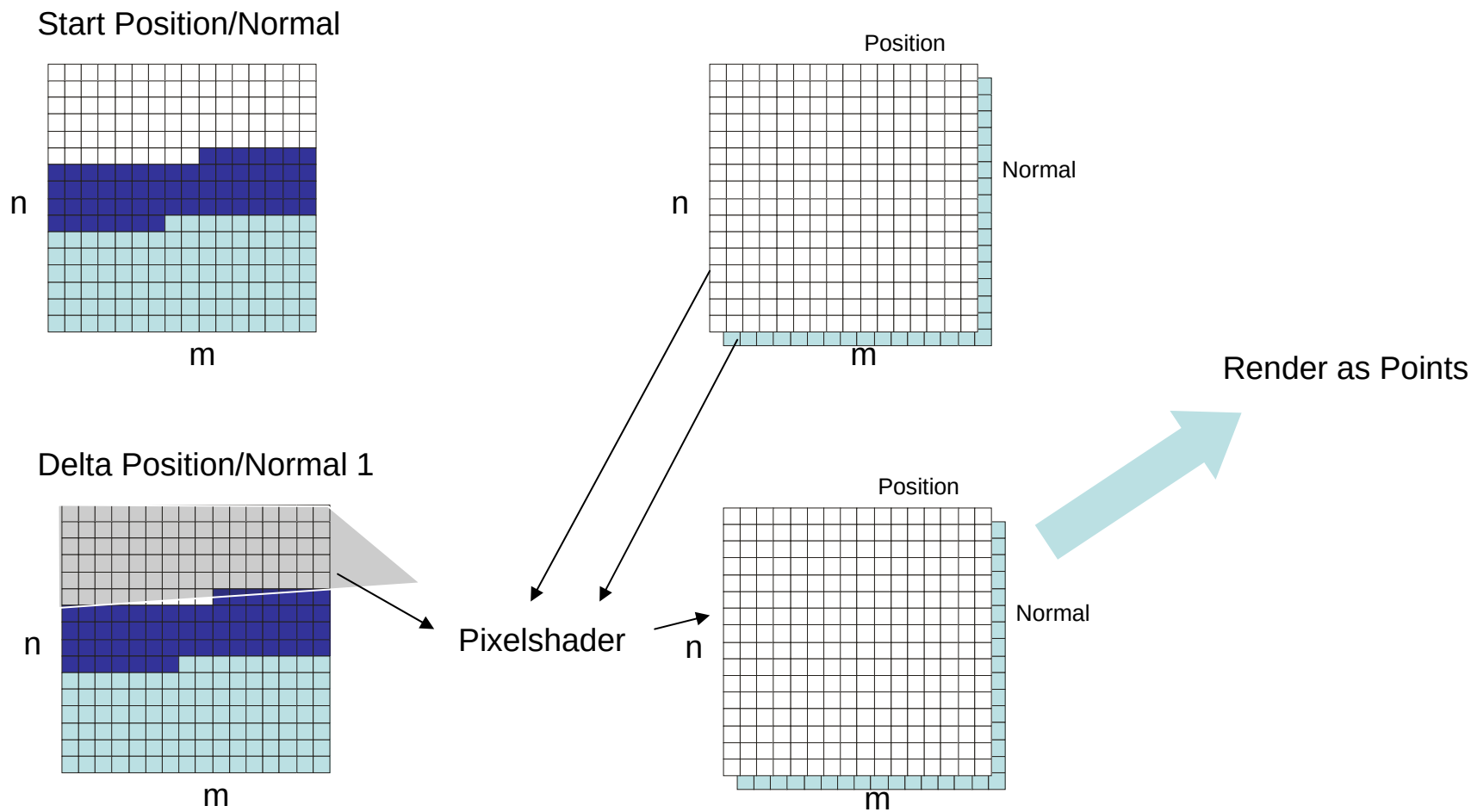


Example 3, Point Compression (cont.)



GPGPU

Example 3, Point Compression (cont.)



GPGPU

Selected References

- Lutz Latta et al., *Building a Million Particle System*, Game Developers Conference 2004 & Graphics Hardware 2004, http://www.2ld.de/gdc2004/
- P. Kipfer, R. Westermann, *UberFlow: A GPU-Based Particle Engine*, Graphics Hardware 2004, http://wwwcg.in.tum.de/Research/Publications
- J. Krüger, R. Westermann, *GPU Simulation and Rendering of Volumetric Effects for Computer Games and Virtual Environments* , Eurographics 2005, http://wwwcg.in.tum.de/Research/Publications
- J. Krüger, P. Kipfer, P. Kondratieva, R. Westermann, *A Particle System for Interactive Visualization of 3D Flows,* IEEE Transactions on Visualization and Computer Graphics, http://wwwcg.in.tum.de/Research/Publications
- NVIDIA, *Cloth Simulation*, I3D 2005 Presentation, http://developer.nvidia.com/object/i3d_2005_presentations.html
- Krüger, J. Westermann, R. *Linear algebra operators for GPU implementation of numerical algorithms*, In Proceedings of SIGGRAPH 2003, ACM Press / ACM SIGGRAPH, http://wwwcg.in.tum.de/Research/Publications
- Bolz , J., Farmer, I., Grinspun, E., Schröder, P. *Sparse Matrix Solvers on the GPU: Conjugate Gradients and Multigrid*, In Proceedings of SIGGRAPH 2003, ACM Press / ACM SIGGRAPH, http://www.multires.caltech.edu/pubs/
- Hillesland, K. E. *Nonlinear Optimization Framework for Image-Based Modeling on Programmable Graphics Hardware*, In Proceedings of SIGGRAPH 2003, ACM Press / ACM SIGGRAPH, http://www.cs.unc.edu/~khillesl/nlopt/
- Fedkiw, R., Stam, J. and Jensen, H.W. *Visual Simulation of Smoke.* In Proceedings of SIGGRAPH 2001, ACM Press / ACM SIGGRAPH. 2001, http://graphics.ucsd.edu/~henrik/papers/smoke/
- Stam, J. *Stable Fluids.* In Proceedings of SIGGRAPH 1999, ACM Press / ACM SIGGRAPH, 121-128. 1999, http://www.dgp.toronto.edu/people/stam/reality/Research/pub.html
- Harris, M., Coombe, G., Scheuermann, T., and Lastra, A. *Physically-Based Visual Simulation on Graphics Hardware*.. Proc. 2002 SIGGRAPH / Eurographics Workshop on Graphics Hardware 2002, http://www.markmark.net/cml/

GPGPU

The End

Websites:

http://www.gpgpu.org

http://wwwcg.in.tum.de/Research/Publications/Compression

http://wwwcg.in.tum.de/Research/Publications/Windtunnel