Creating Flood Effects in Uncharted 3

Eben Cook
VFX Artist @ Naughty Dog

GDC 12
GAME DEVELOPERS CONFERENCE
SAN FRANCISCO, CA
MARCH 5-9, 2012
EXPO DATES: MARCH 7-9
2012

Me me me

- BA in Communication Design from UNT
  - Computer Science minor
- 11 years in the industry. EALA, Naughty Dog
- I’ve been:
  - Concept Artist
  - Texture Artist
  - Modeler
  - Technical Artist/VFX

We did lots of kinds of water

- Puddles

We did lots of kinds of water

- Puddles
- Waterfalls

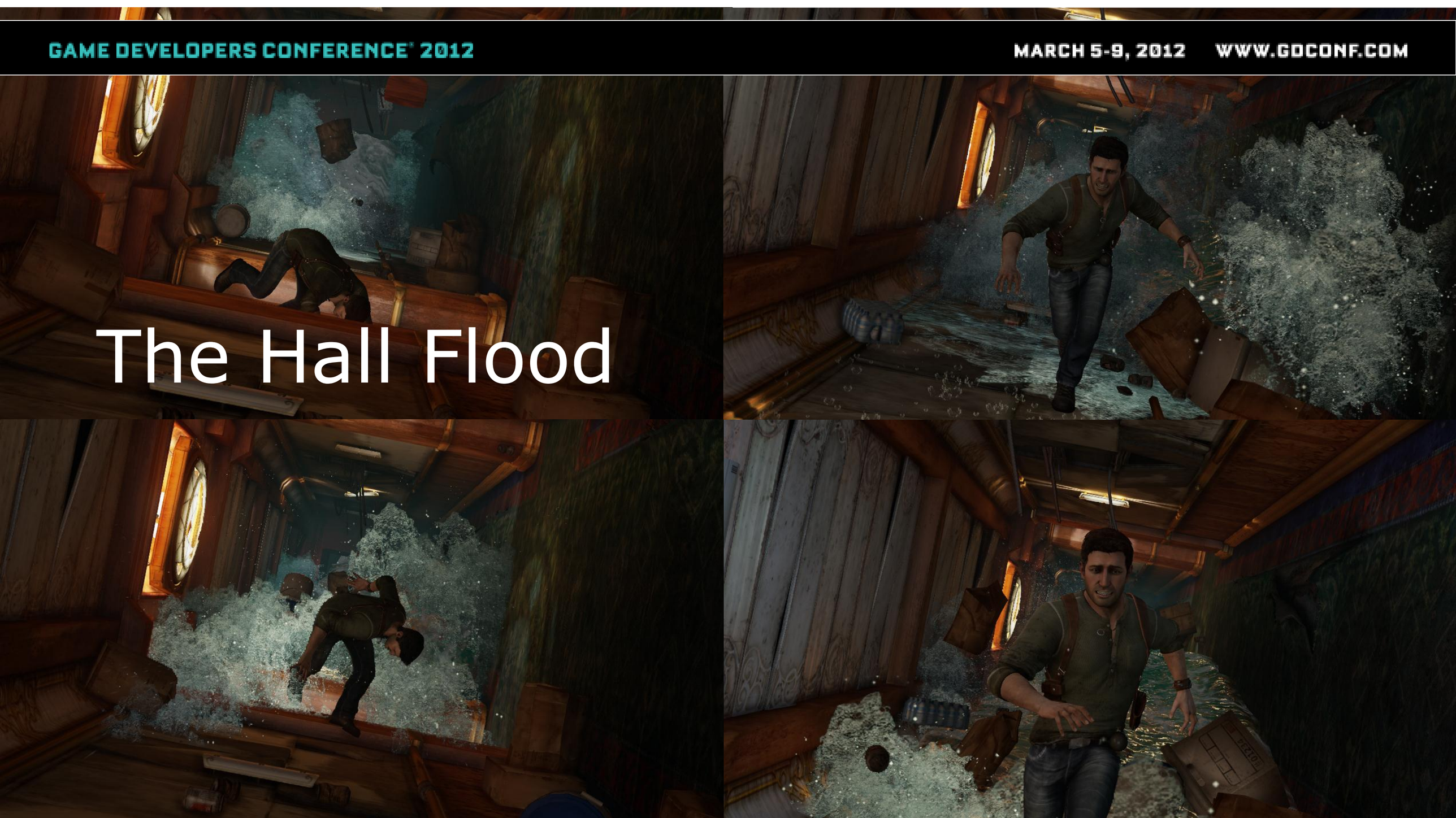
We did lots of kinds of water

- Waterfalls
- Puddles
- Oceans

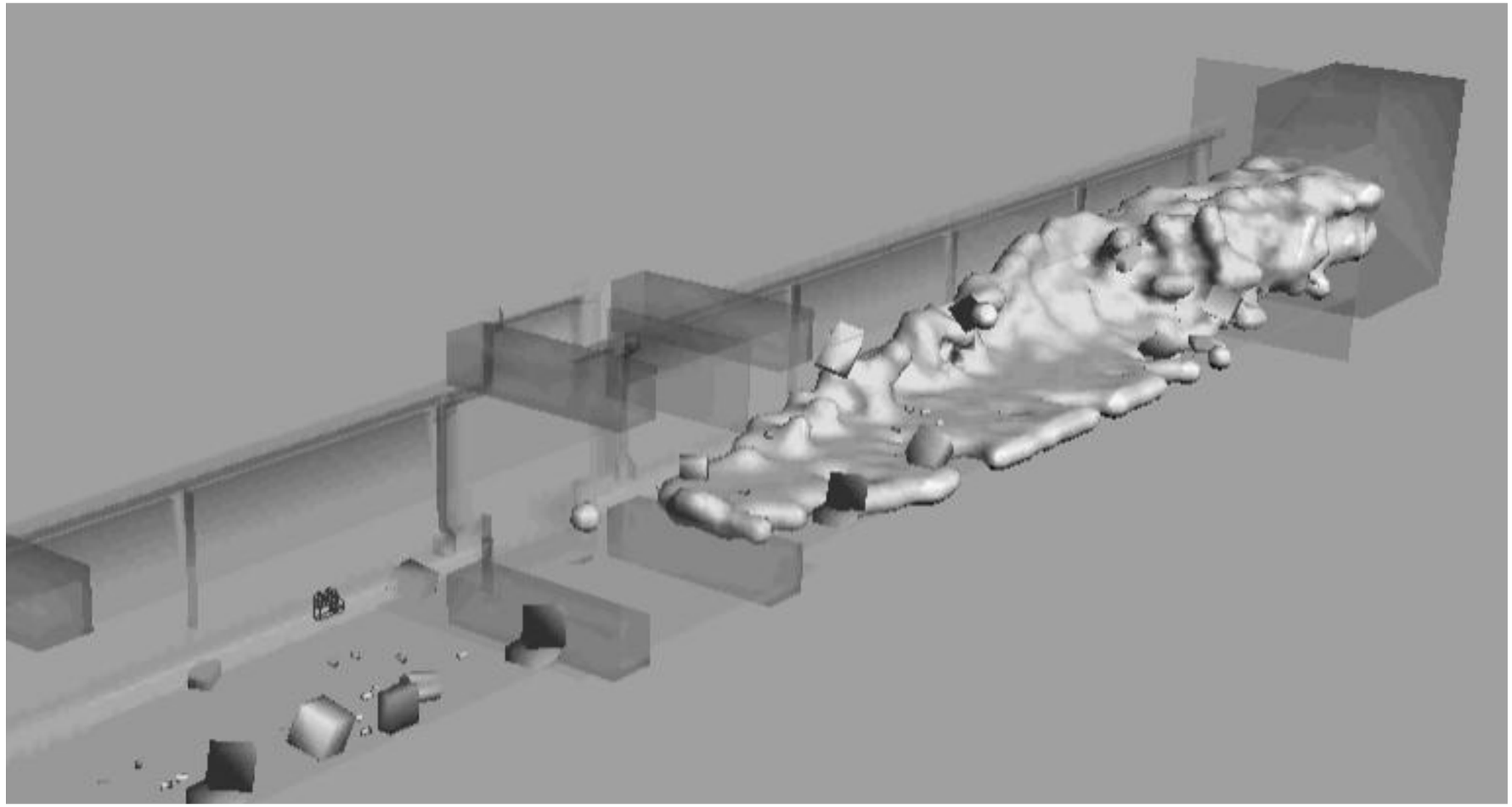
We did lots of kinds of water

- Waterfalls
- Puddles
- Oceans
- Floods

The Hall Flood



Fluid sim



Fluid sim

- Done in Houdini
  - Worked through ideas on movement and timing
  - Continued using throughout
    - Created in-game mesh from sim
    - Particles were inspired by sim
    - Rigid bodies were pushed by sim data

Creating the mesh

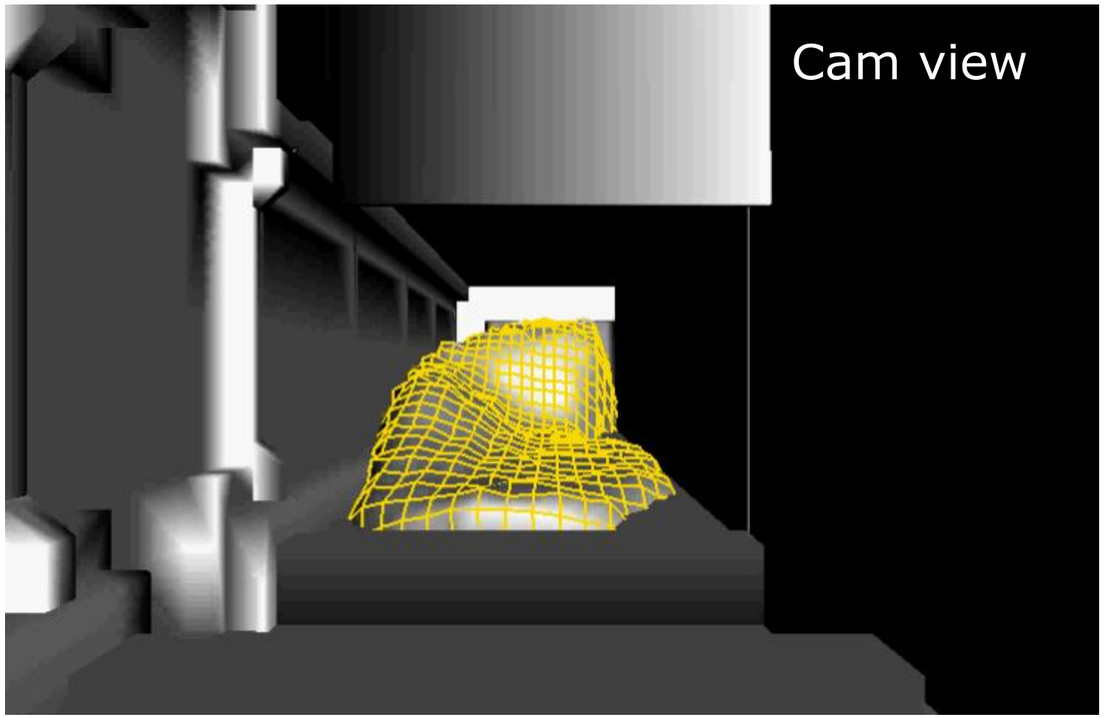
- Considered loading an animated point cache.
  - Would have required a lot of new tech and taken a lot of time.

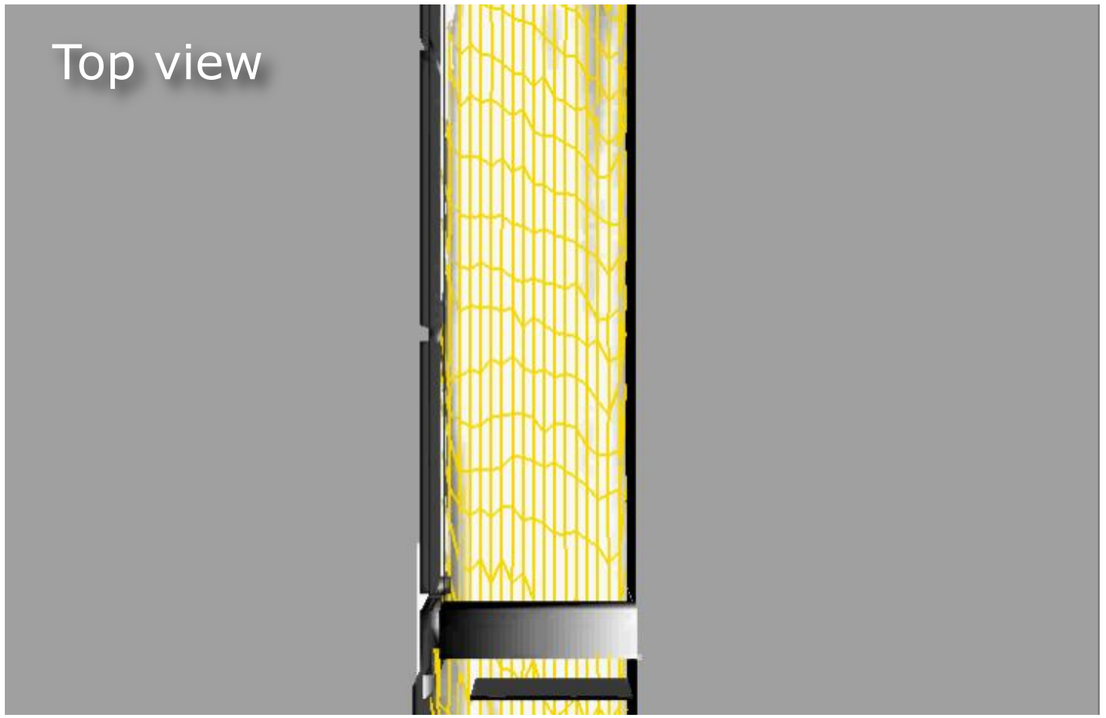
Creating the mesh

- 1st attempt: programmatically sliding waves across the surface.
  - Abandoned because it was hard to work with and didn’t look good.
- Decided to go with brute force method
  - Skeletal meshes with 1 joint per vertex

Creating the mesh

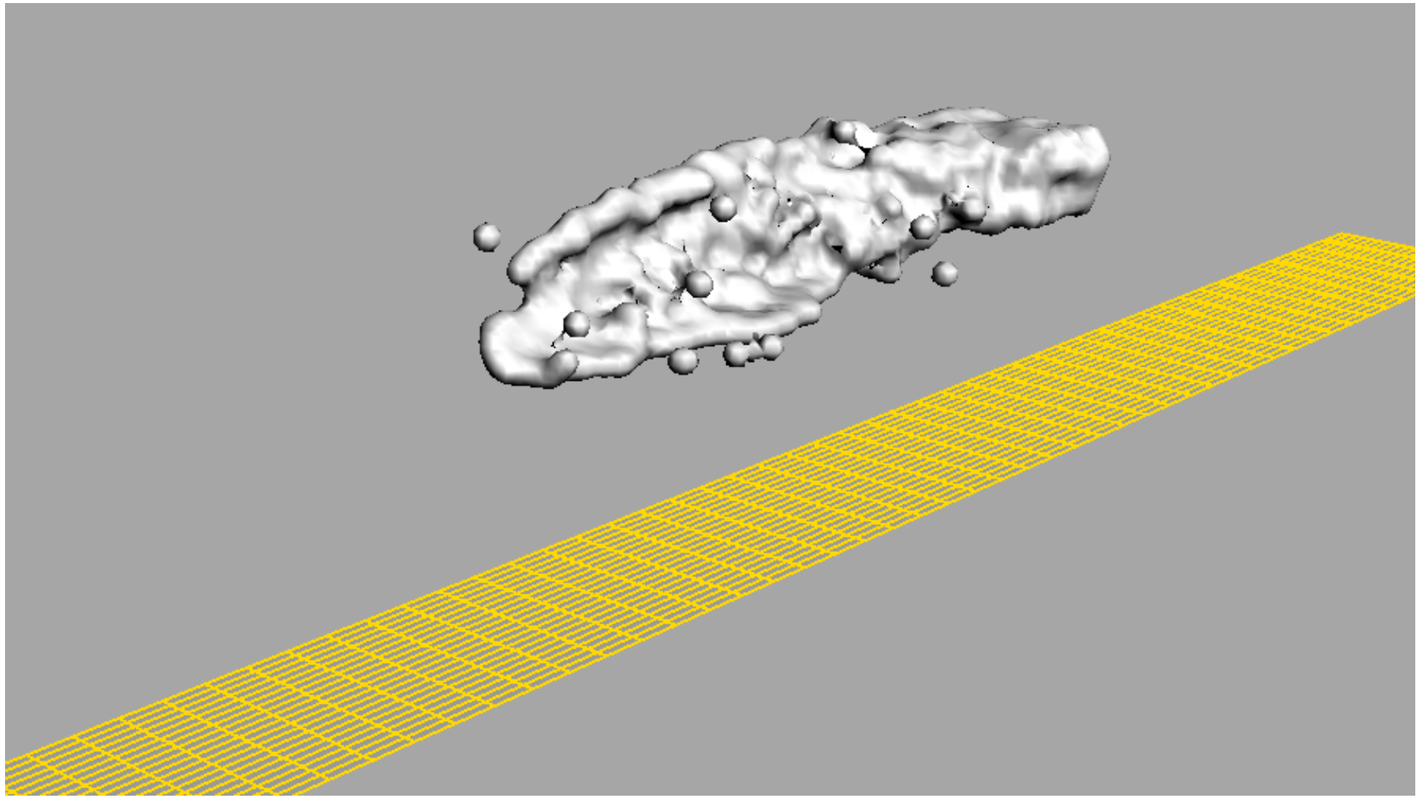
- Constrained camera angle allowed for optimization: rectangles instead of squares.





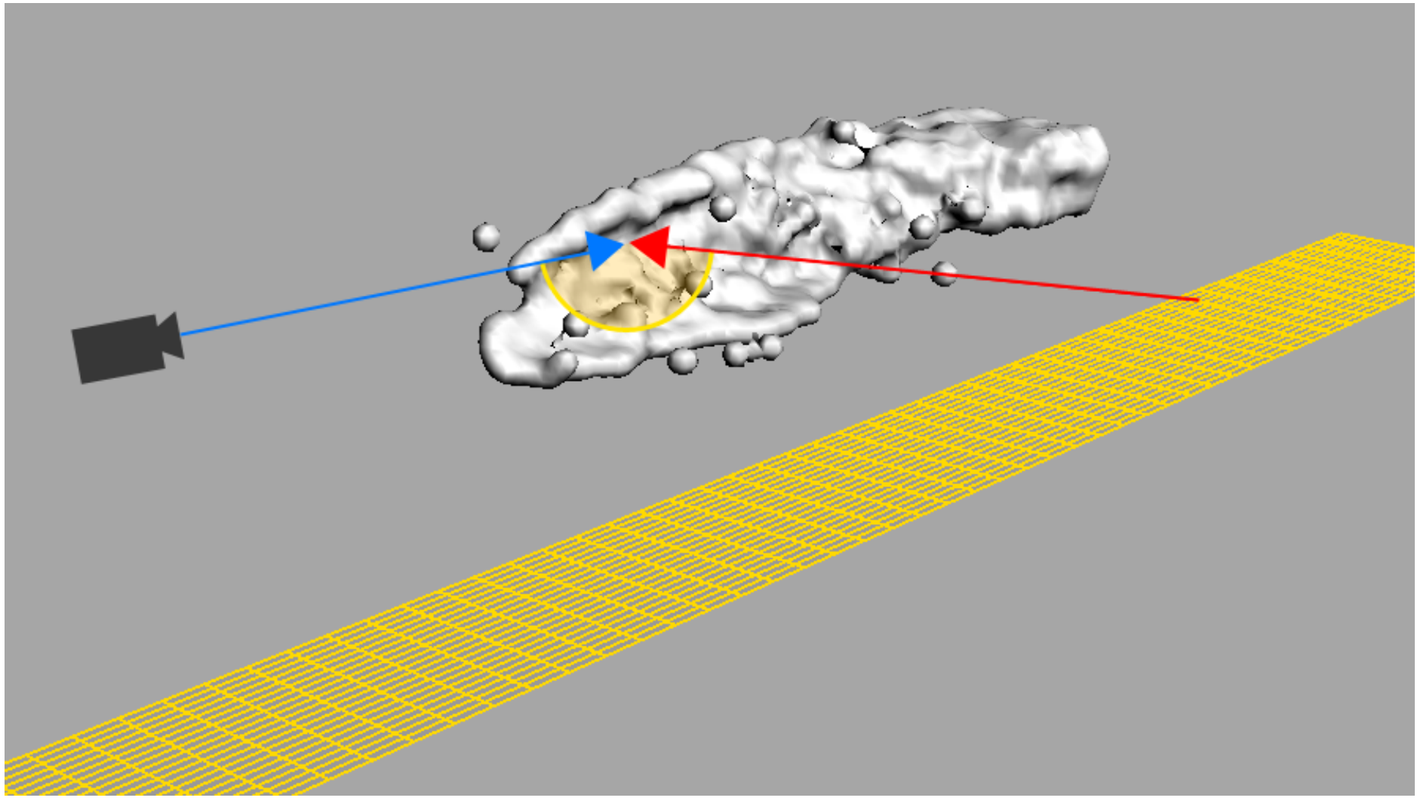
Creating the mesh

- Didn’t ray cast along surface normal



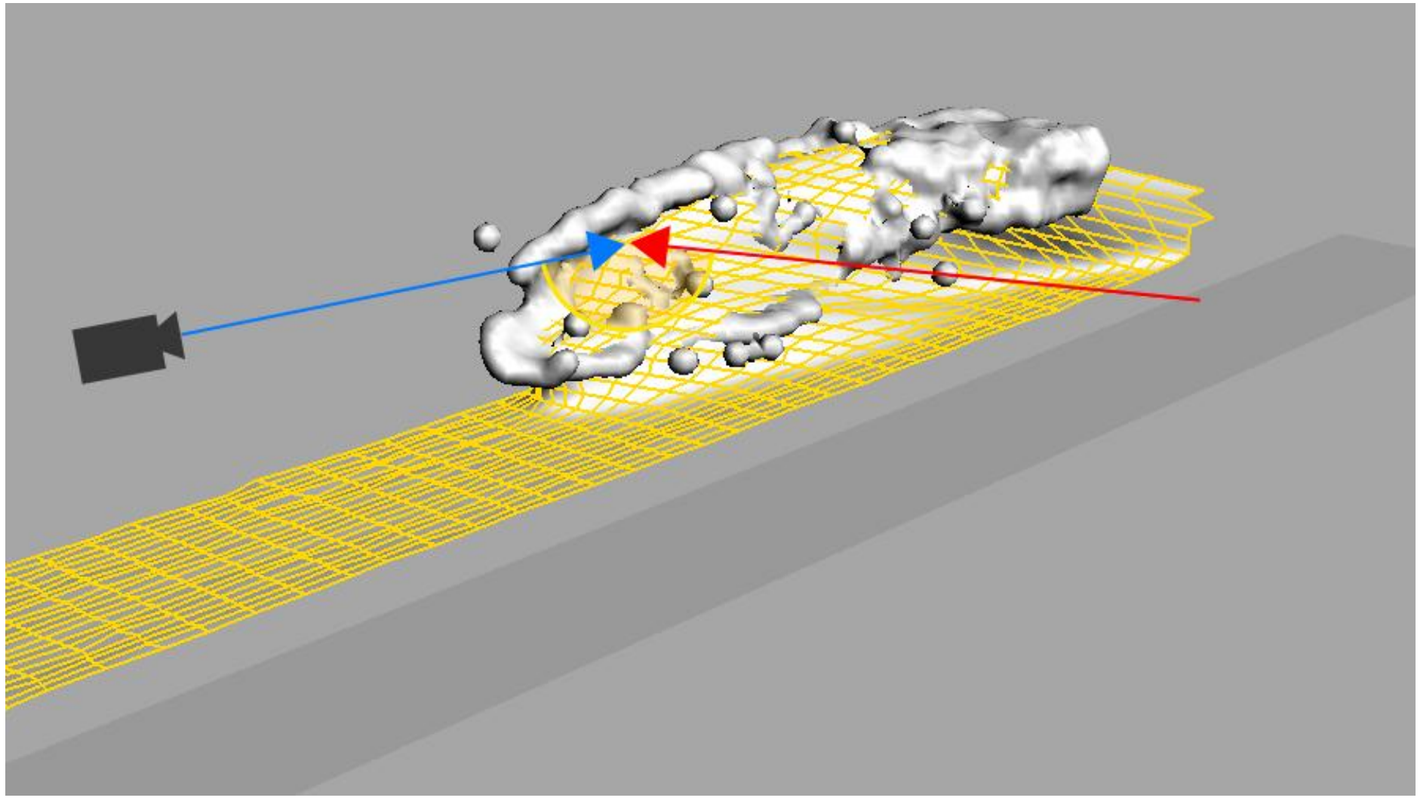
Creating the mesh

- Didn’t ray cast along surface normal
- Instead cast toward camera



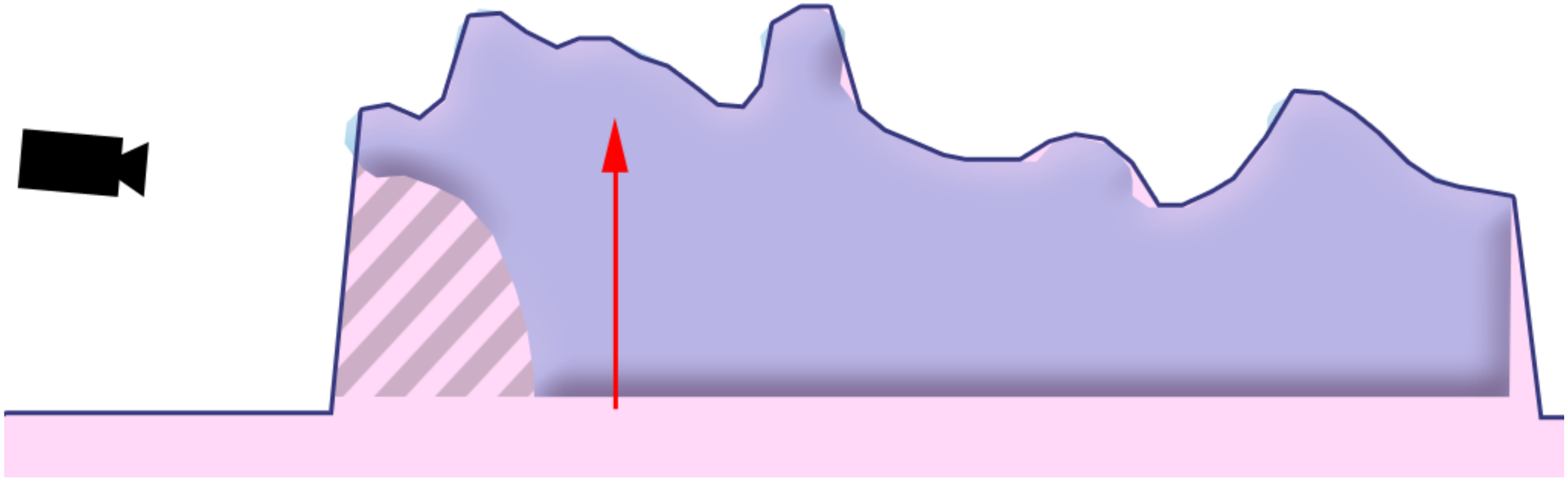
Creating the mesh

- Didn’t ray cast along surface normal
- Instead cast toward camera



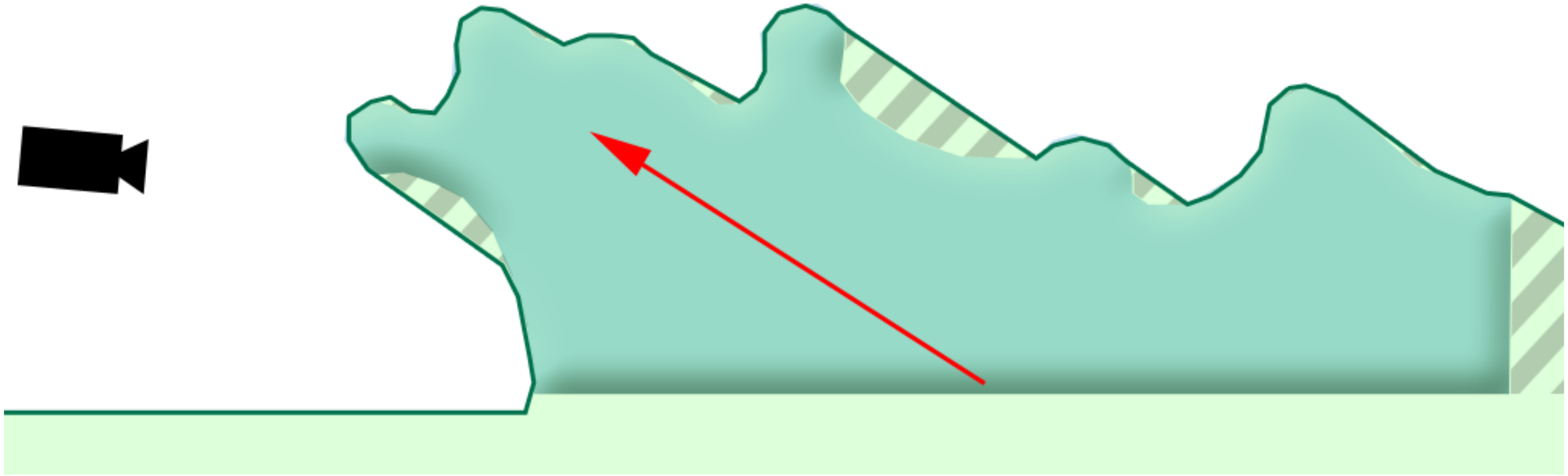
Creating the mesh

- Errors are very visible to the camera when casting along the surface normal



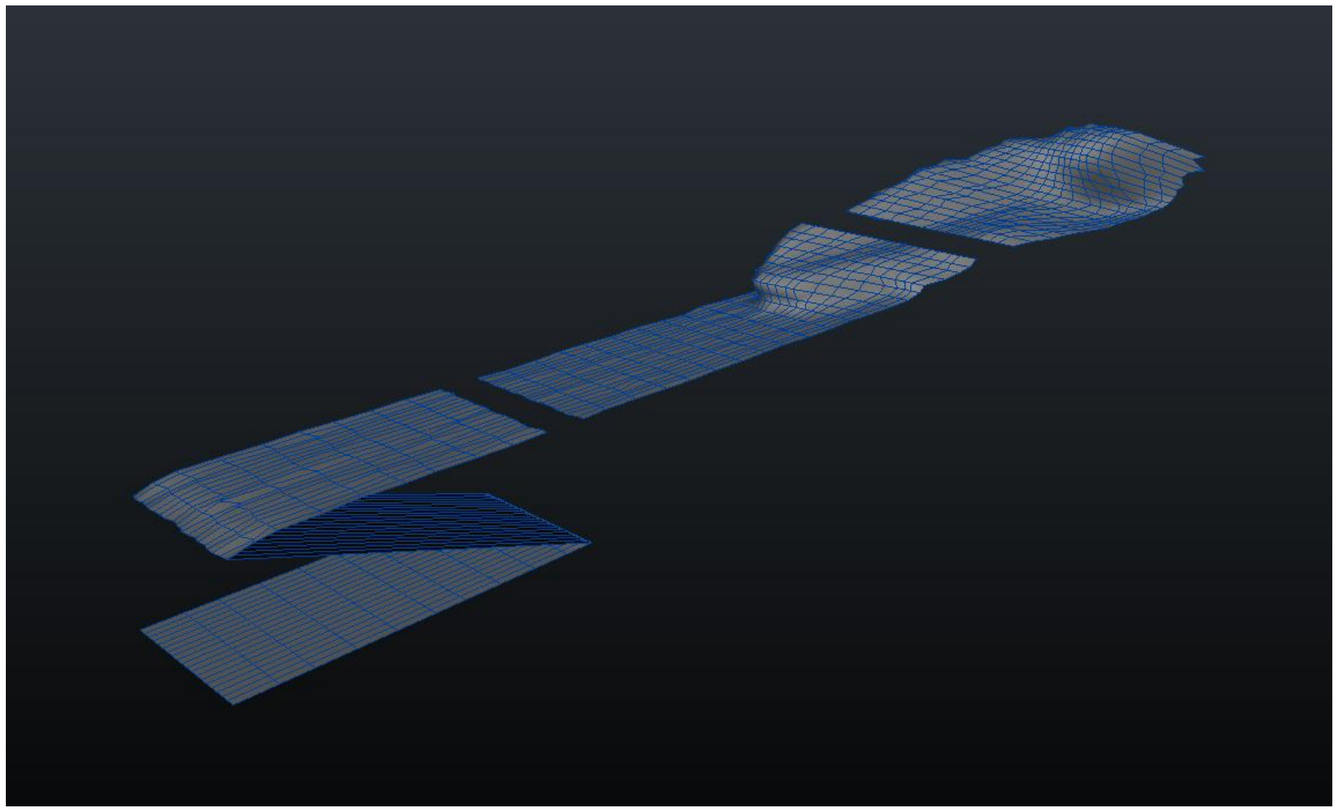
Creating the mesh

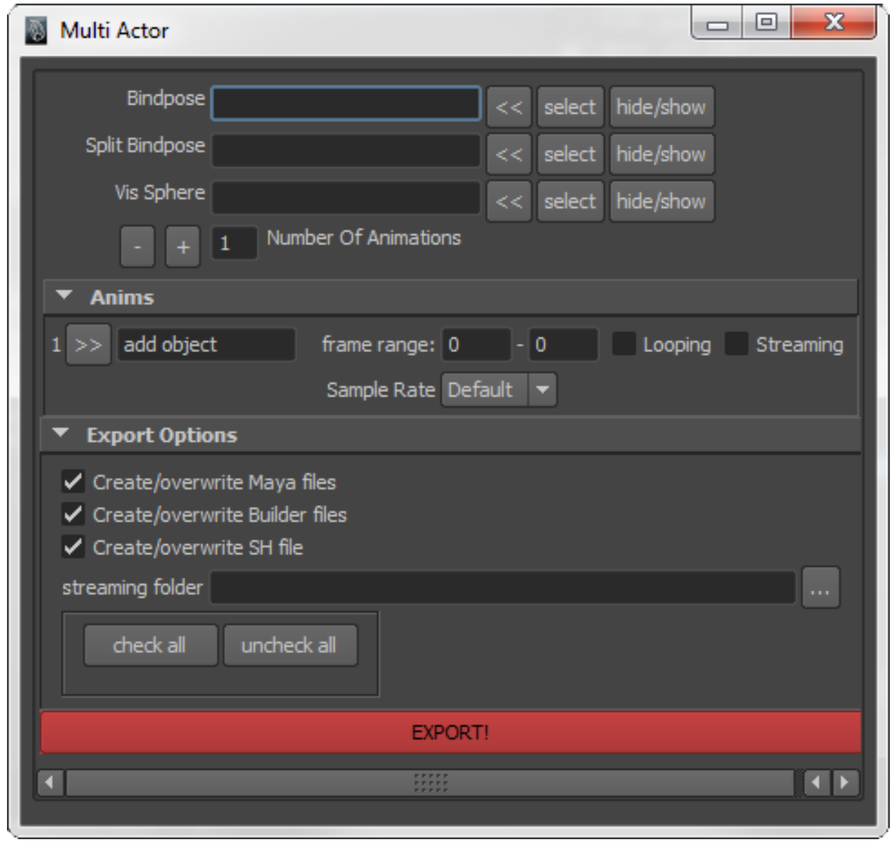
- Errors are minimized and hidden from the camera when casting toward the camera



Creating the mesh

- Only needed 3 actors with ~400 joints each





The Surface Mesh

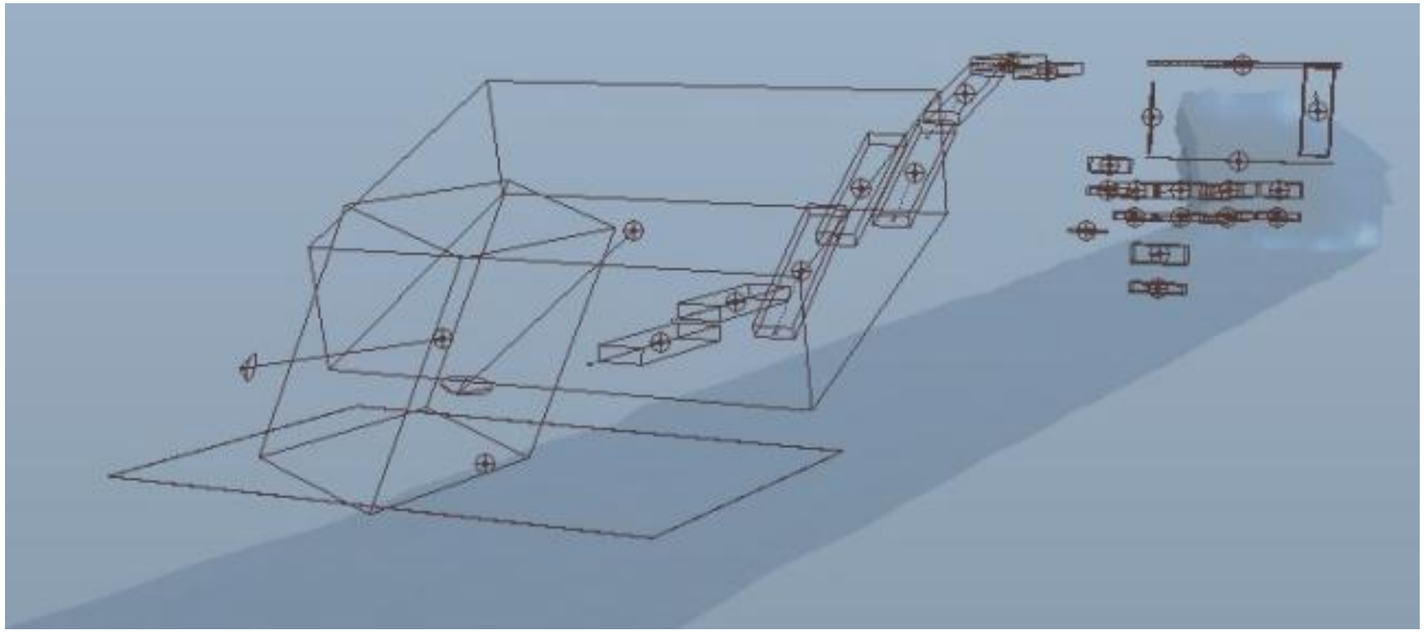
The Surface Shader

- Refraction
  - Distorts screen buffer based on depth and normal
  - Opacity based on depth
- Cube Map
  - With fresnel
- Foam

The Surface Shader

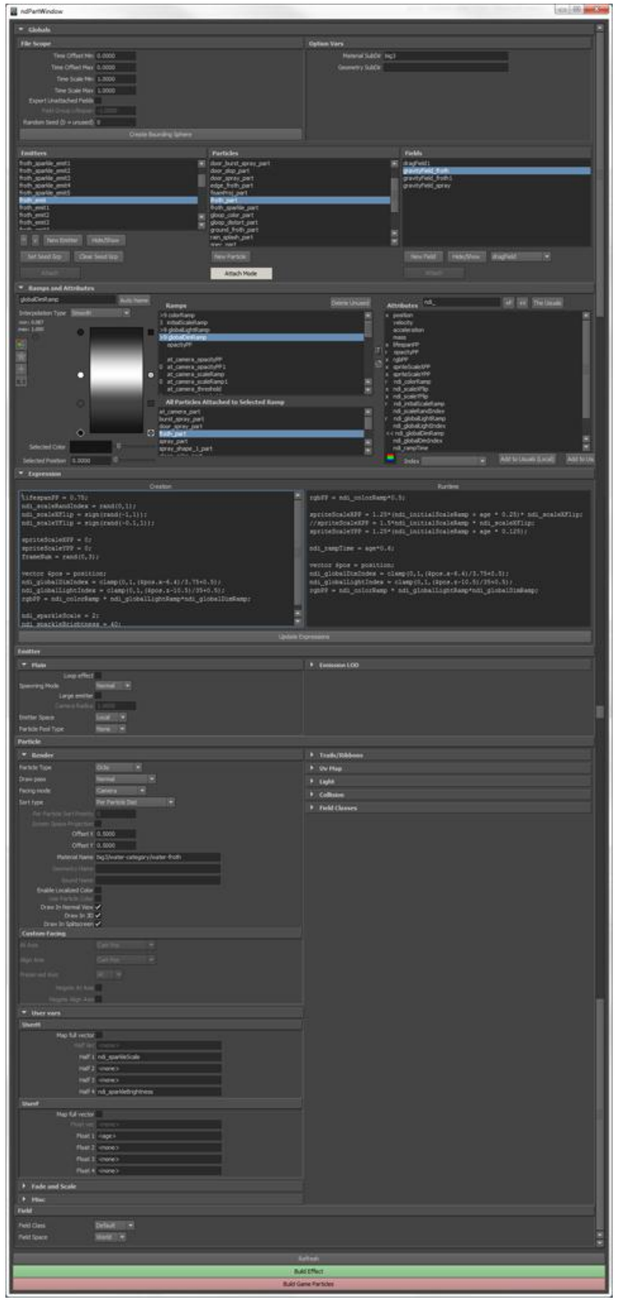
Particles

- 31 separate emitters
- 8 different particle definitions



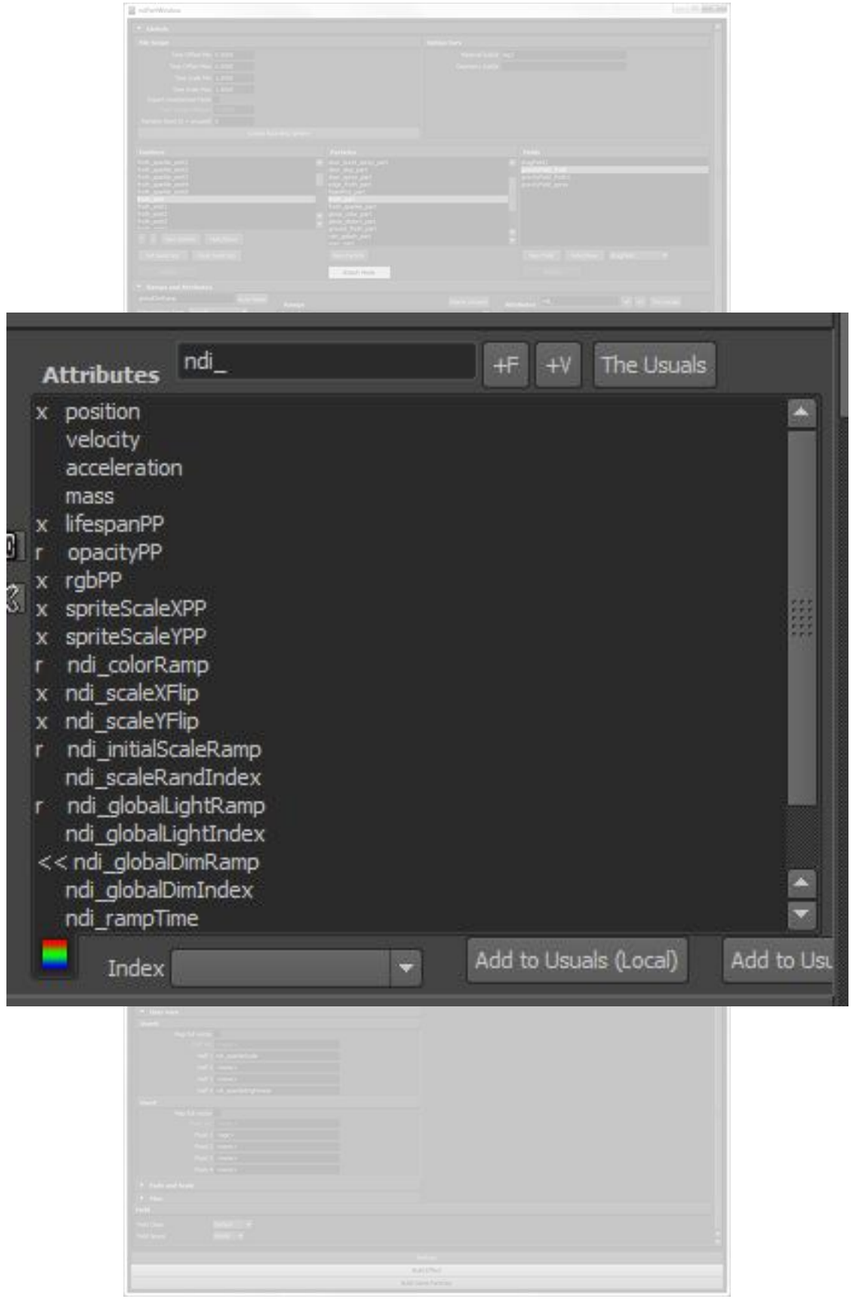
Particles

- Robust tools and runtime



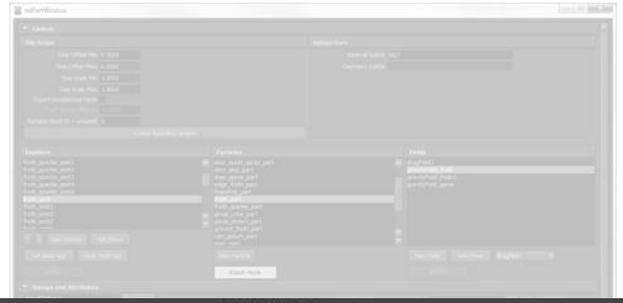
Particles

- Robust tools and runtime
  - Custom attributes



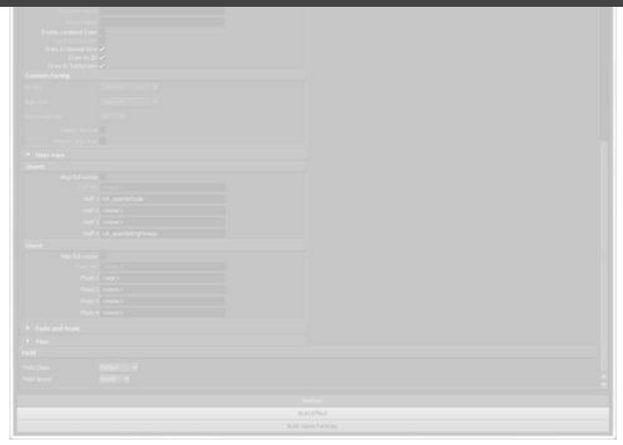
Particles

- Robust tools and runtime
  - Custom attributes
  - Expressions!!!



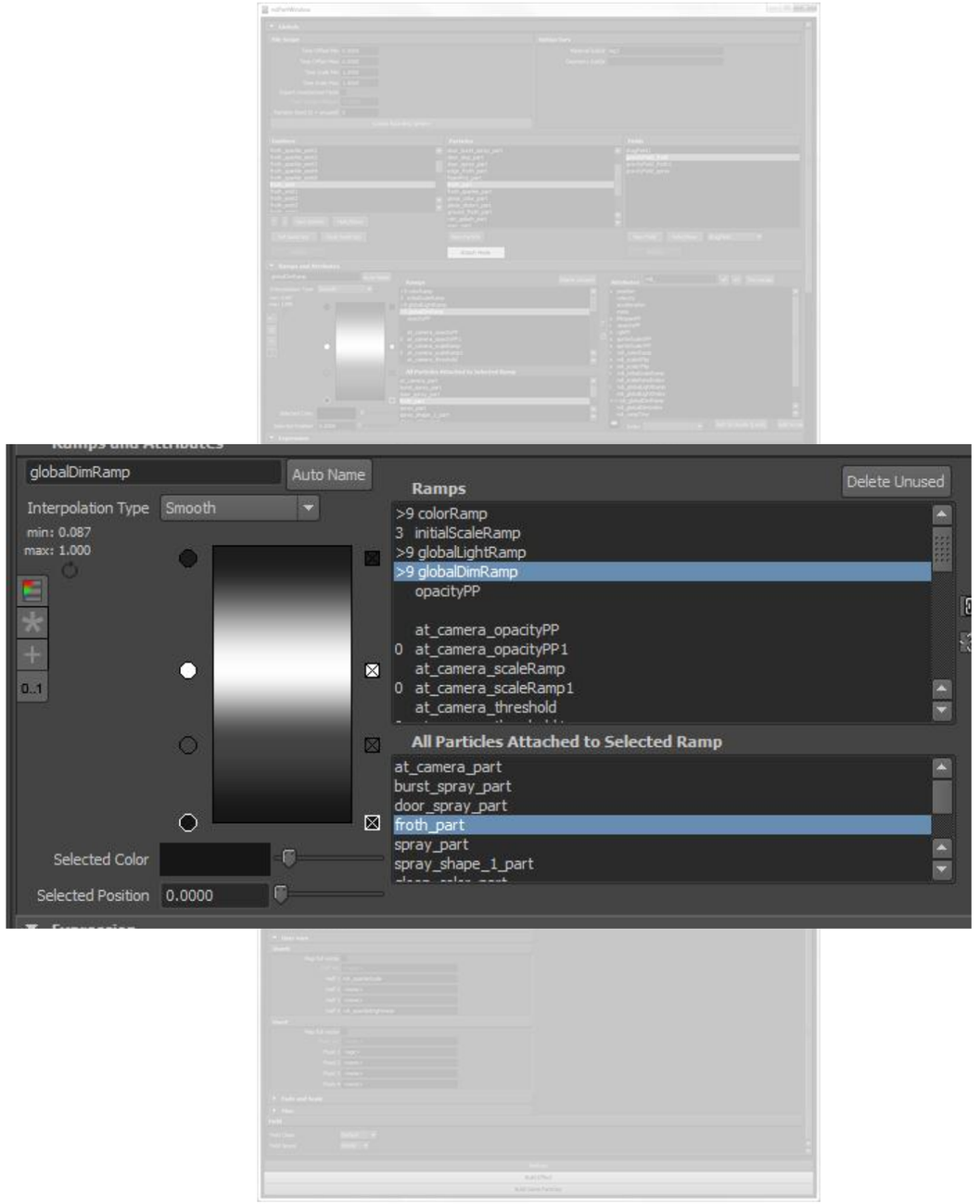
Runtime

```
rgbPP = ndi_colorRamp*0.5;

spriteScaleXPP = 1.25*(ndi_initialScaleRamp + age * 0.25)* ndi_scaleXFlip;
//spriteScaleXPP = 1.5*ndi_initialScaleRamp * ndi_scaleXFlip;
spriteScaleYPP = 1.25*(ndi_initialScaleRamp + age * 0.125);

ndi_rampTime = age*0.6;

vector $pos = position;
ndi_globalDimIndex = clamp(0,1,($pos.x-6.4)/3.75+0.5);
ndi_globalLightIndex = clamp(0,1,($pos.z-10.5)/35+0.5);
rgbPP = ndi_colorRamp * ndi_globalLightRamp*ndi_globalDimRamp;
```



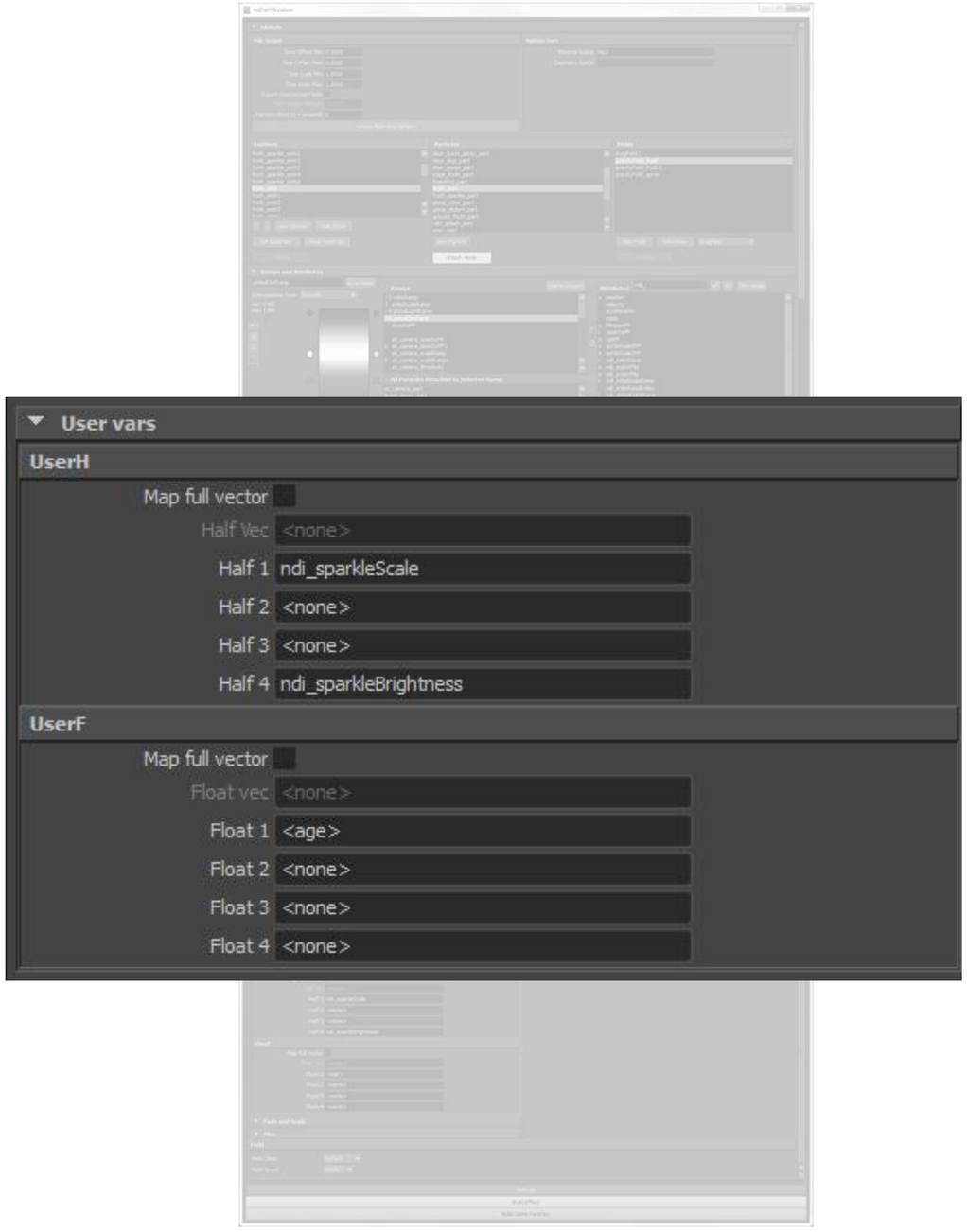
Particles

- Robust tools and runtime
  - Custom attributes
  - Expressions!!!
  - Ramps with custom inputs



Particles

- Robust tools and runtime
  - Custom attributes
  - Expressions!!!
  - Ramps with custom inputs
  - Send data to shader

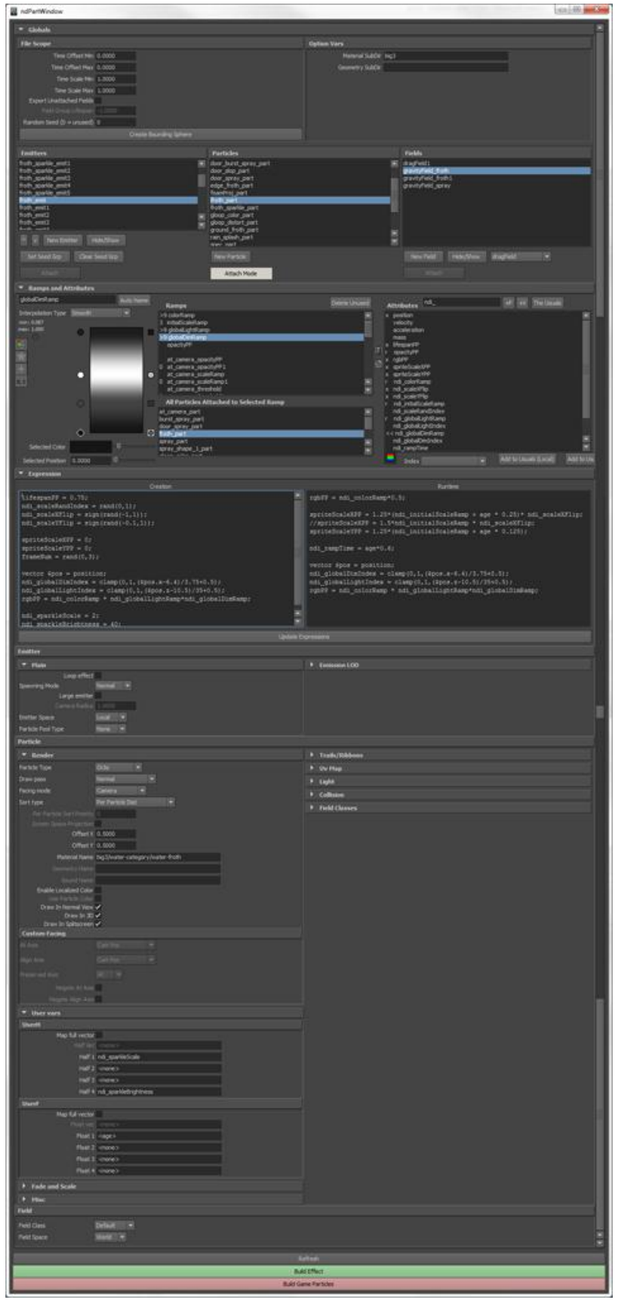


Particles

- Robust tools and runtime
  - Custom attributes
  - Expressions!!!
  - Ramps with custom inputs
  - Send data to shader
  - Custom orientation

Particles

- Robust tools and runtime
  - Custom attributes
  - Expressions!!!
  - Ramps with custom inputs
  - Send data to shader
  - Custom orientation
  - Much, much more
    (thank you Marshall Robin)

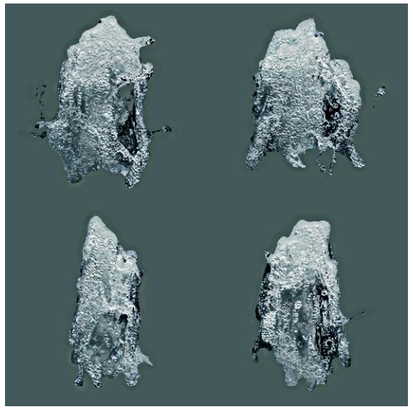


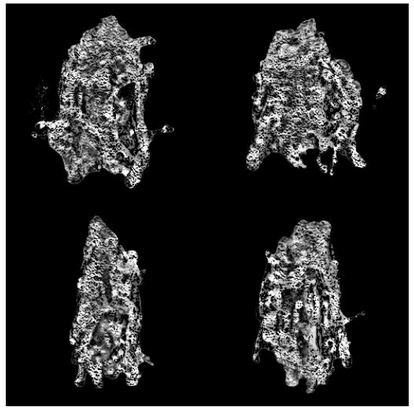
Particle optimization

- Particles were rendered to a 1/4 res buffer
- Octagons were used instead of quads

Particles

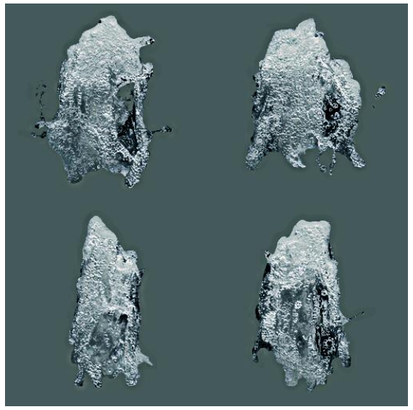
- Froth: main particle
  - 4 varieties

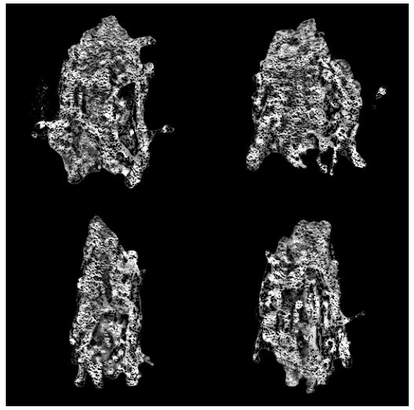


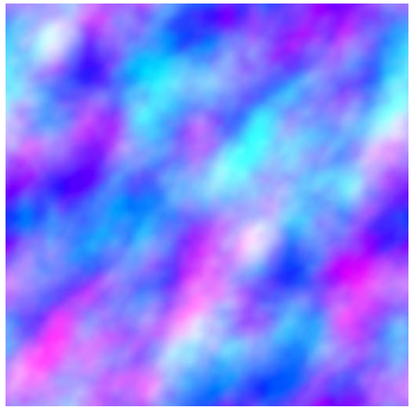


Particles

- Froth: main particle
  - 4 varieties
  - UV distortion: additional variety

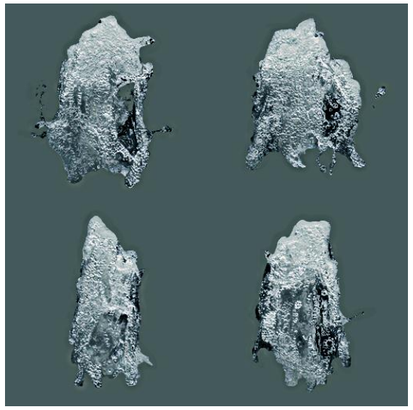
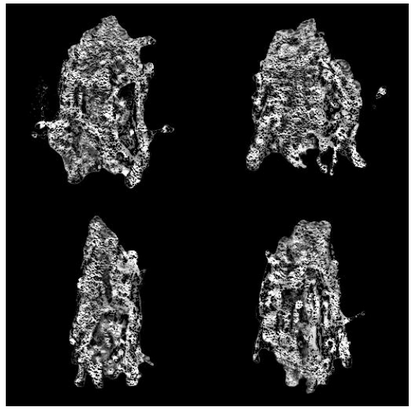
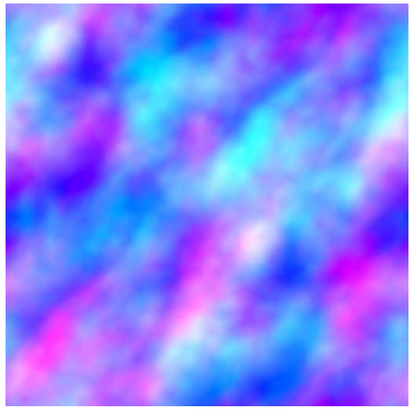
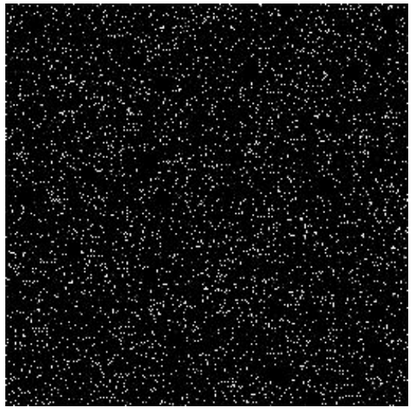
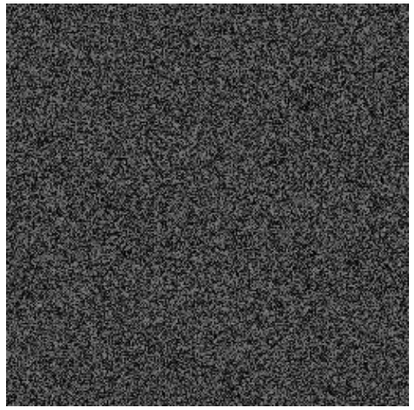






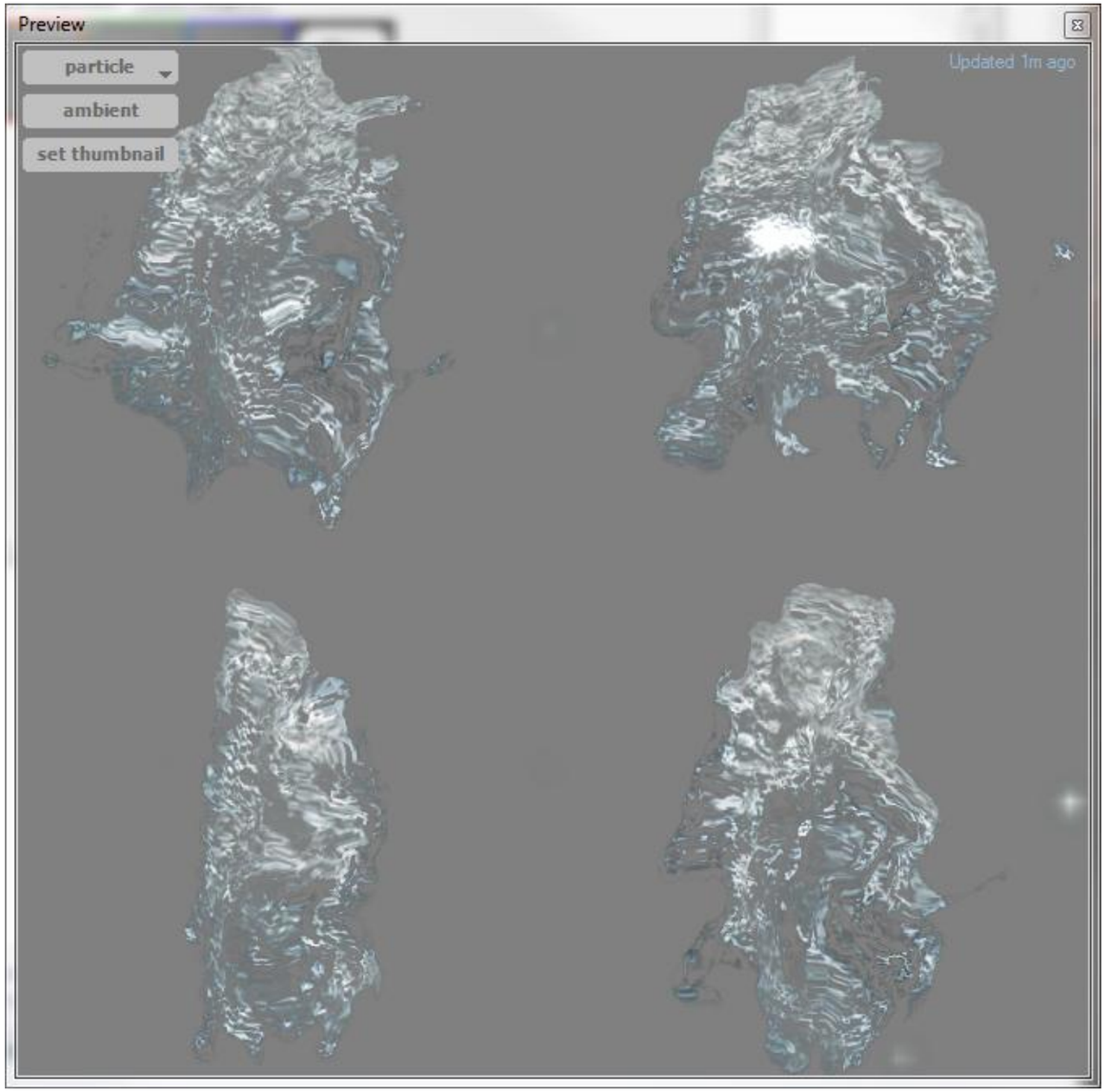
Particles

- Froth: main particle
  - 4 varieties
  - UV distortion: additional variety
  - Sparkles: sparse dots masked by uniform noise

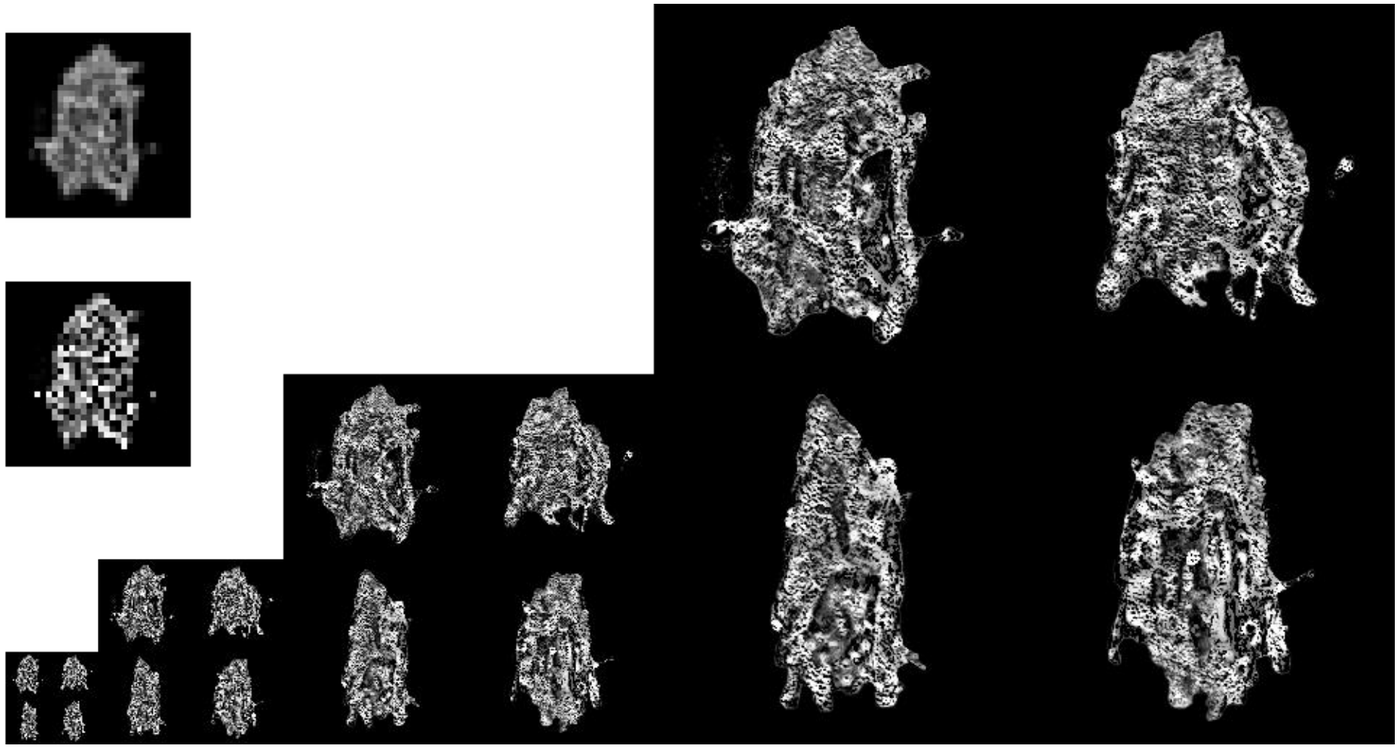
Particles

- Froth: main particle
  - 4 varieties
  - UV distortion
  - Sparkles



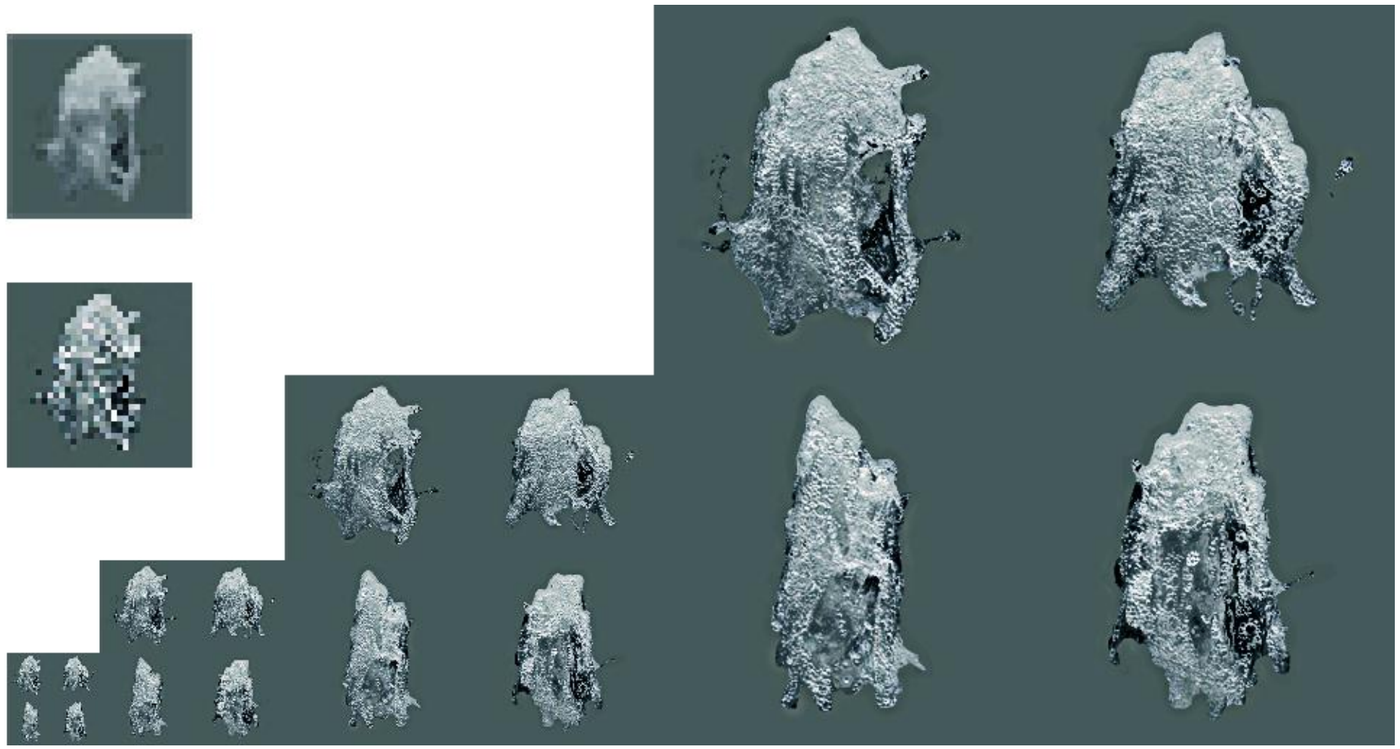
Particles

- Custom mip maps
  - Nearest neighbor sampling to retain crispness



Particles

- Custom mip maps
  - Nearest neighbor sampling to retain crispness



Particles

Particle Lighting

Particle Lighting

- It’s a hack!

Particle Lighting

- It’s a hack!
- Lit with 2 ramps

Particle Lighting

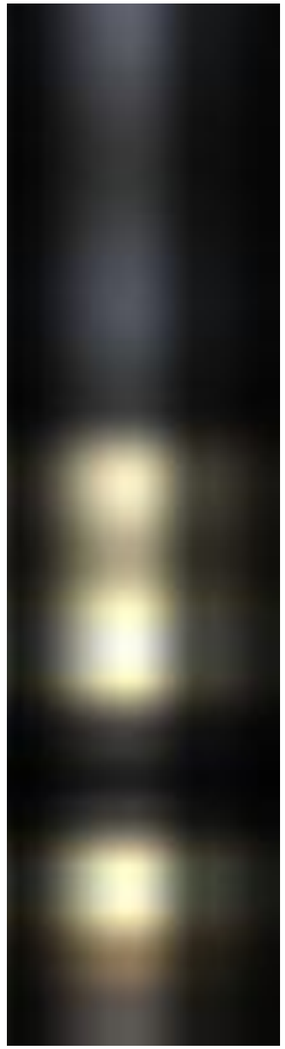
- It’s a hack!
- Lit with 2 ramps
  - 1 ran the length of the hall

Particle Lighting

- It’s a hack!
- Lit with 2 ramps
  - 1 ran the length of the hall
  - 1 crossed the width of the hall

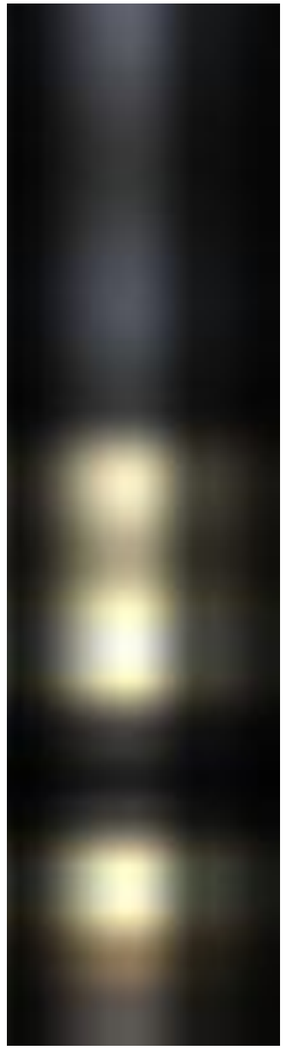
Particle Lighting

- It’s a hack!
- Lit with 2 ramps
  - 1 ran the length of the hall
  - 1 crossed the width of the hall
  - The two were multiplied together



Particle Lighting

- It’s a hack!
- Lit with 2 ramps
  - 1 ran the length of the hall
  - 1 crossed the width of the hall
  - The two were multiplied together
- Essentially a 2D projection



Lit Particles

Ta-da!