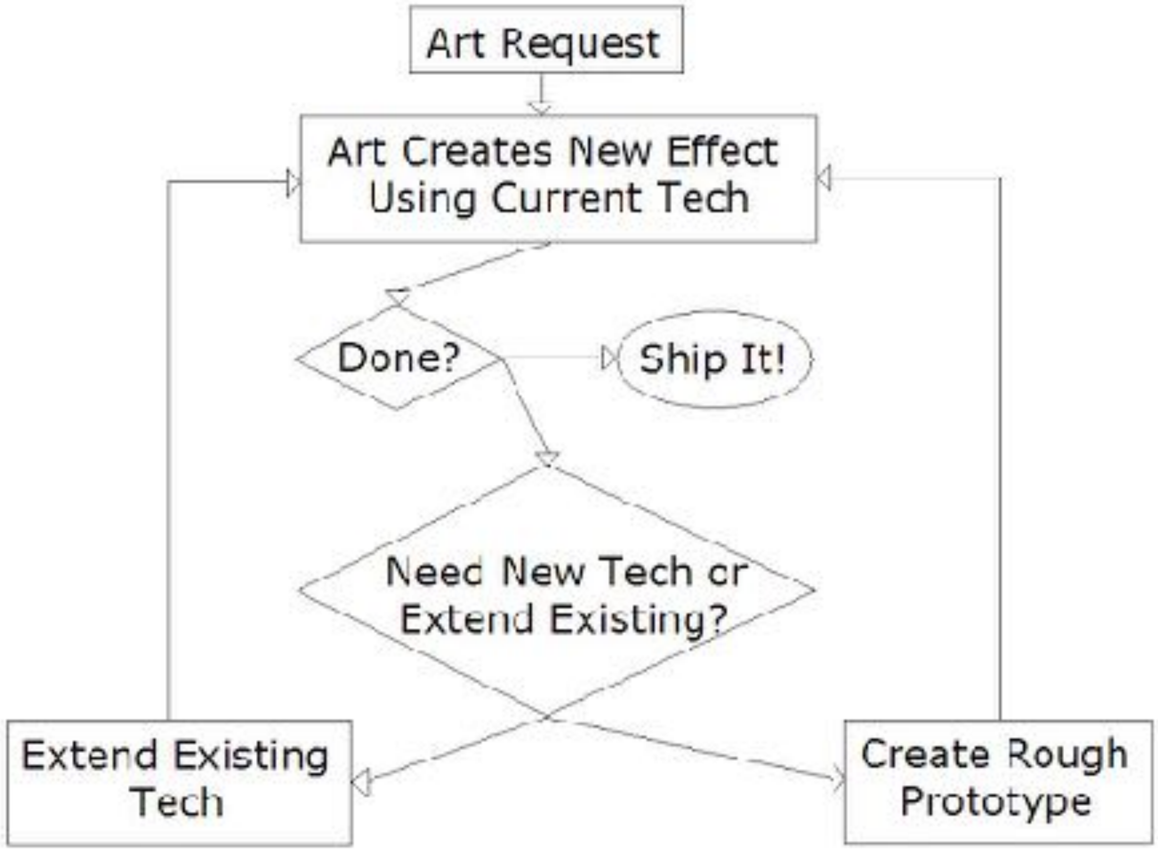
The Art and Technology Behind Bioshock's Special Effects

Presented by
Stephen Alexander (Effects Artist)
Jesse Johnson (Graphics Programmer)

Presentation Overview

- Iterative Workflow Between Art and Tech
- Lit Particles
- Bioshock’s Water: An Artist’s Perspective
- Technology Behind Interactive Waterfalls
- Water Caustics
- Interactive Rippling Water System
- Optimization Case Study: Bioshock’s Fire

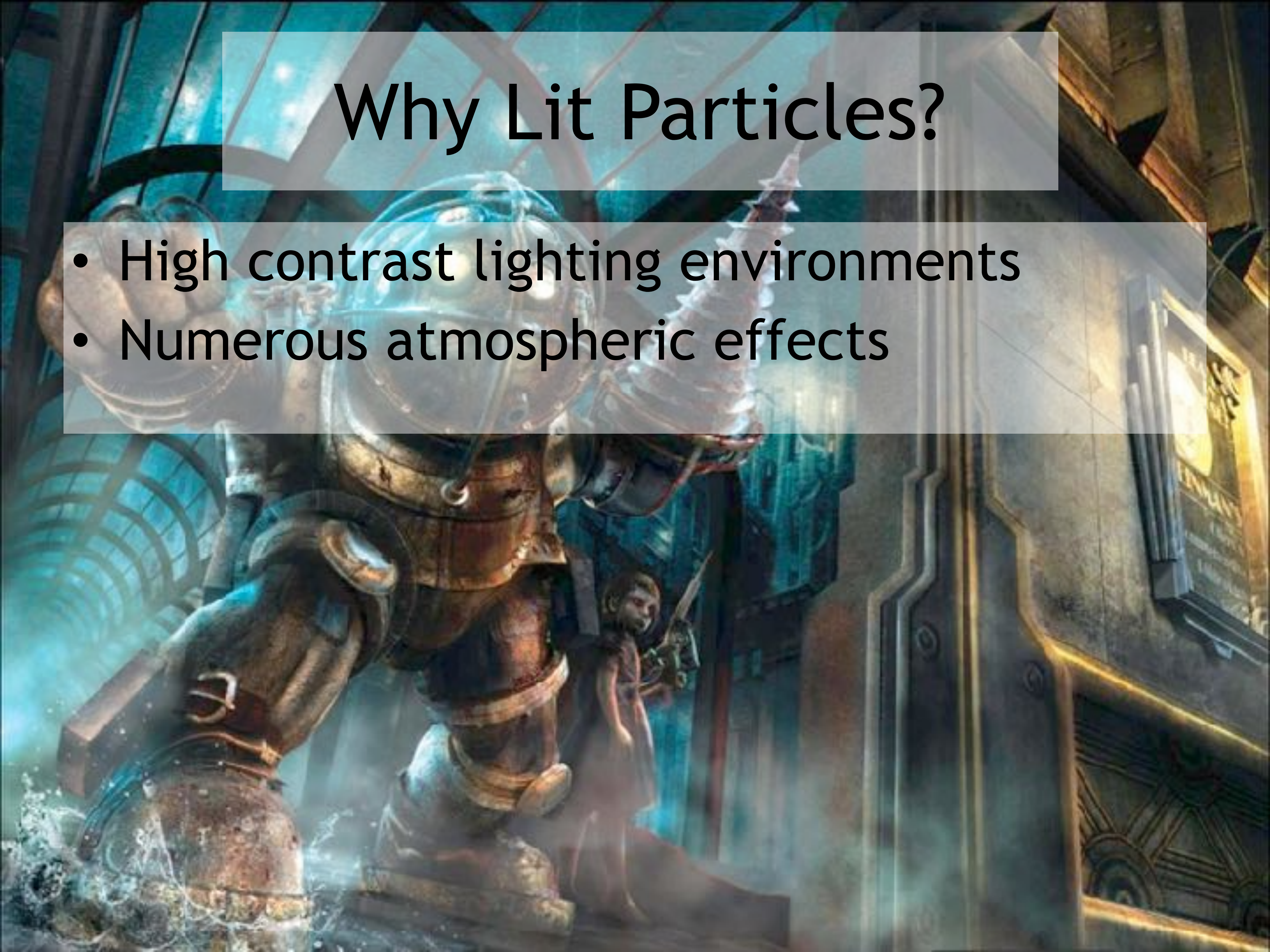
Art and Tech Iterative Workflow



The FX Closet

Why Lit Particles?

- High contrast lighting environments
- Numerous atmospheric effects




www.fraps.com
999
999
6-993
Pistol Rounds


www.fr
Electro Bolt

Technical Challenges With Lit Particles

- Must be rendered in a single pass
- Need to satisfy very high performance requirements
- Potential for shader permutation issues

Lit Particles: Iteration 1

- Light attenuation done per vertex
- Static branching used to accumulate multiple lights in vertex shader
- Attenuated color interpolated via single float3
- Per vertex approximation typically good for highly tessellated geometry

Lit Particles: Iteration 2

- Problems with iteration 1:
  - Particles too dark in total darkness
  - Particles too bight in bright environments
- Solutions
  - Used maximum of new ambient term, vertex lighting
    - Not ambient + lighting
  - Scale RGB down if any component over 1.0
    - newRGB = oldRGB / (1.0 + max(0.0, maxRGBChannelValue -1.0));


www.fraps.com
31 cur FP
30 avg FP
27 min FP
vsync on
9-959
Machine Gun Rounds


www.fraps.com
32 cur FP
30 avg FP
vsync on
999
999
29-959
Machine Gun Rounds

Lit Particles: Iteration 2

- Problems with iteration 1:
  - Particles too dark in total darkness
  - Particles too bight in bright environments
- Solutions
  - Used maximum of new ambient term, vertex lighting
    - Not ambient + lighting
  - Scale RGB down if any component over 1.0
    - newRGB = oldRGB / (1.0 + max(0.0, maxRGBChannelValue -1.0));

Lit Particles: Iteration 3

- Big need for per pixel lighting
  - Specular highlights
  - Avoid painting highlights into textures
- First prototype: 1 per pixel light
  - Remaining lights used per vertex lighting
  - Two new interpolators: per pixel direction and per pixel color

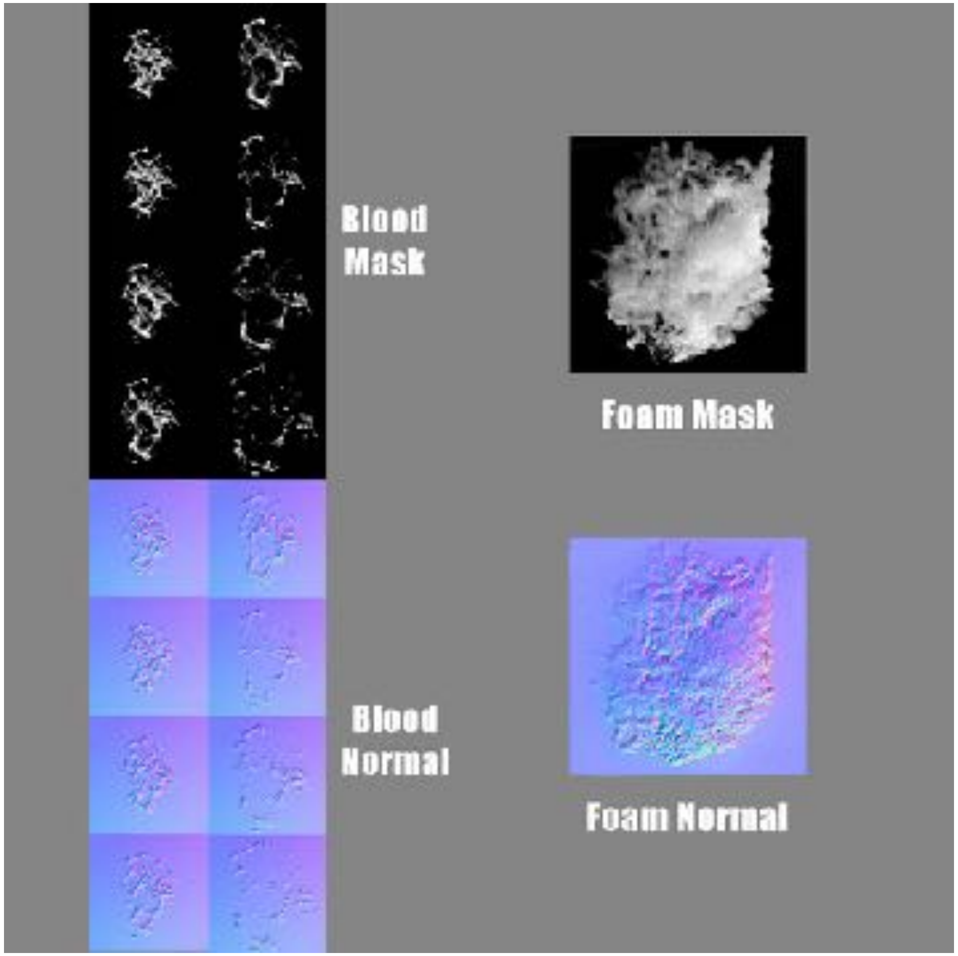

www.fraps.com
SMASH the DEBRIS blocking the door.


www.fraps.com
[DEBUG: Illumination will not affect vision]
Electro Bolt

Lit Particles: Iteration 4

- Numerous issues with iteration 3
  - Popping
  - Washed out lighting
  - Heavily interpolator limited
- New solution: Averaged per pixel lighting
  - Attenuated color same as before
  - Light direction for all sources accumulated per vertex
  - Weighted by:
    - Incident angle for each light
    - Magnitude of light's color and attenuation

Per-Pixel Blood And Foam




www.fraps.com
[DEBUG: Illumination will not affect vision]
33 cur FPS
avg FPS
27 min FPS
vsync on

Lit Particles: Iteration 5

- Particles would be dark depending on lighting direction
- Implemented fake light

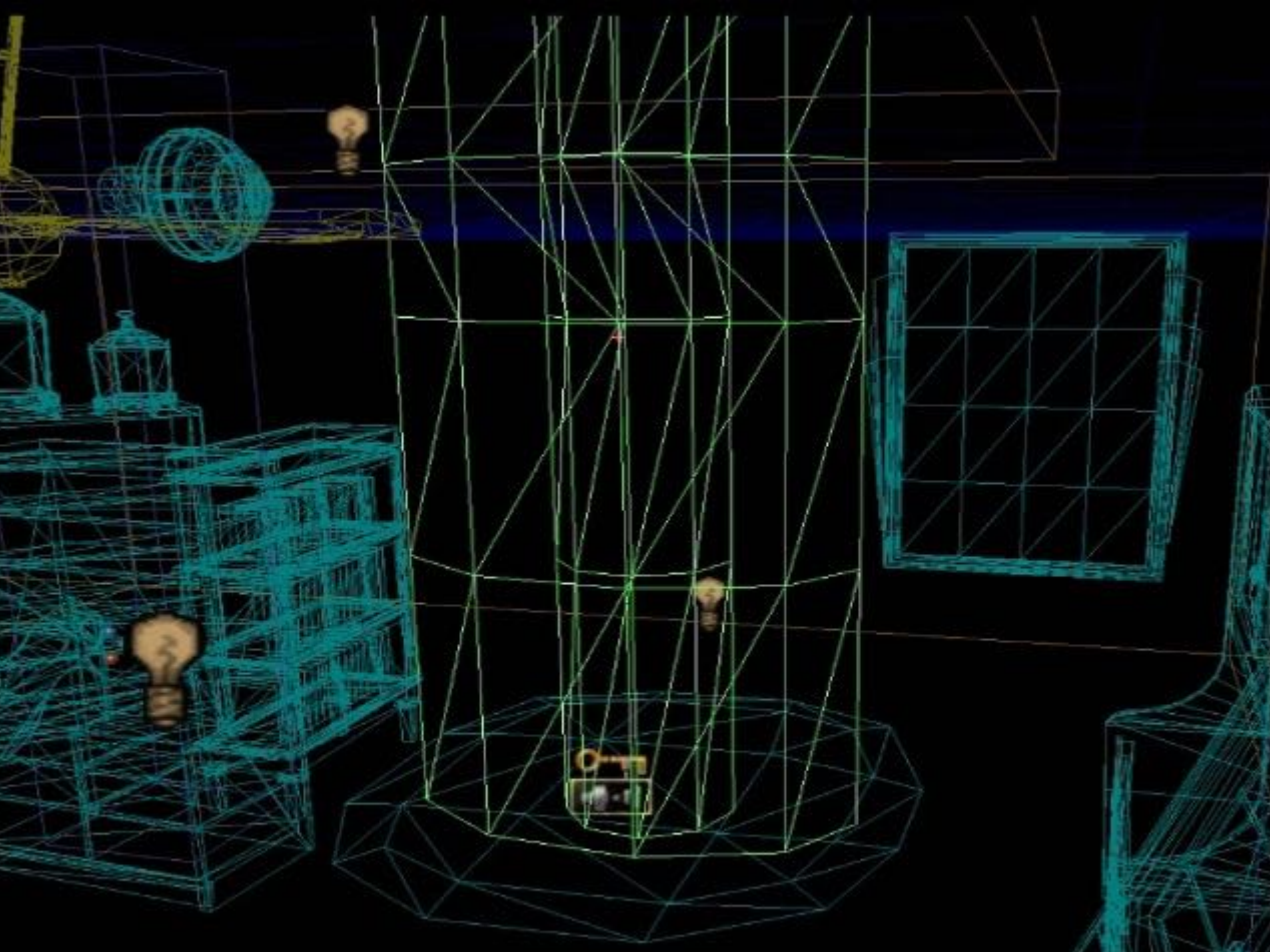

www.fraps.com
[DEBUG: Illumination will not affect vision]
Electro Bolt


www.fraps.com
4-1
Pistol Rounds


www.fraps.com
[DEBUG: Illumination will not affect vision]
4-0
Pistol Rounds

Lit Particles: Final Thoughts

- Performance limitations
- Performance benefits




www.fraps.com
[DEBUG: Illumination will not affect vision]


www.fraps.com

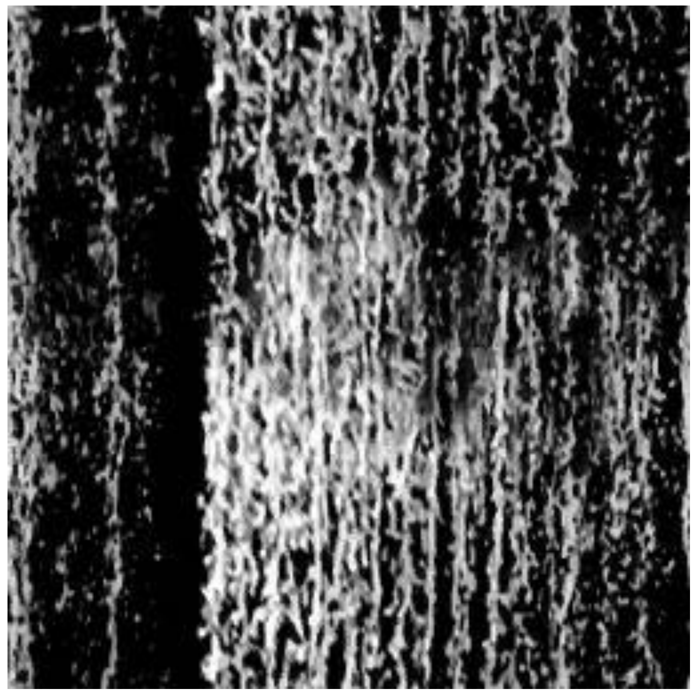
Water: An Artist's Perspective

- Underwater game? Probably need to make water look ok.
- Began using existing tech
- Limitations spawned a variety of art requests


www.fraps.com
OH-BOY WHAT A FLOOR SHOW
EVERY NIGHT


www.fraps.com
30 avg FP
vsync on

Iteration 1 Texture

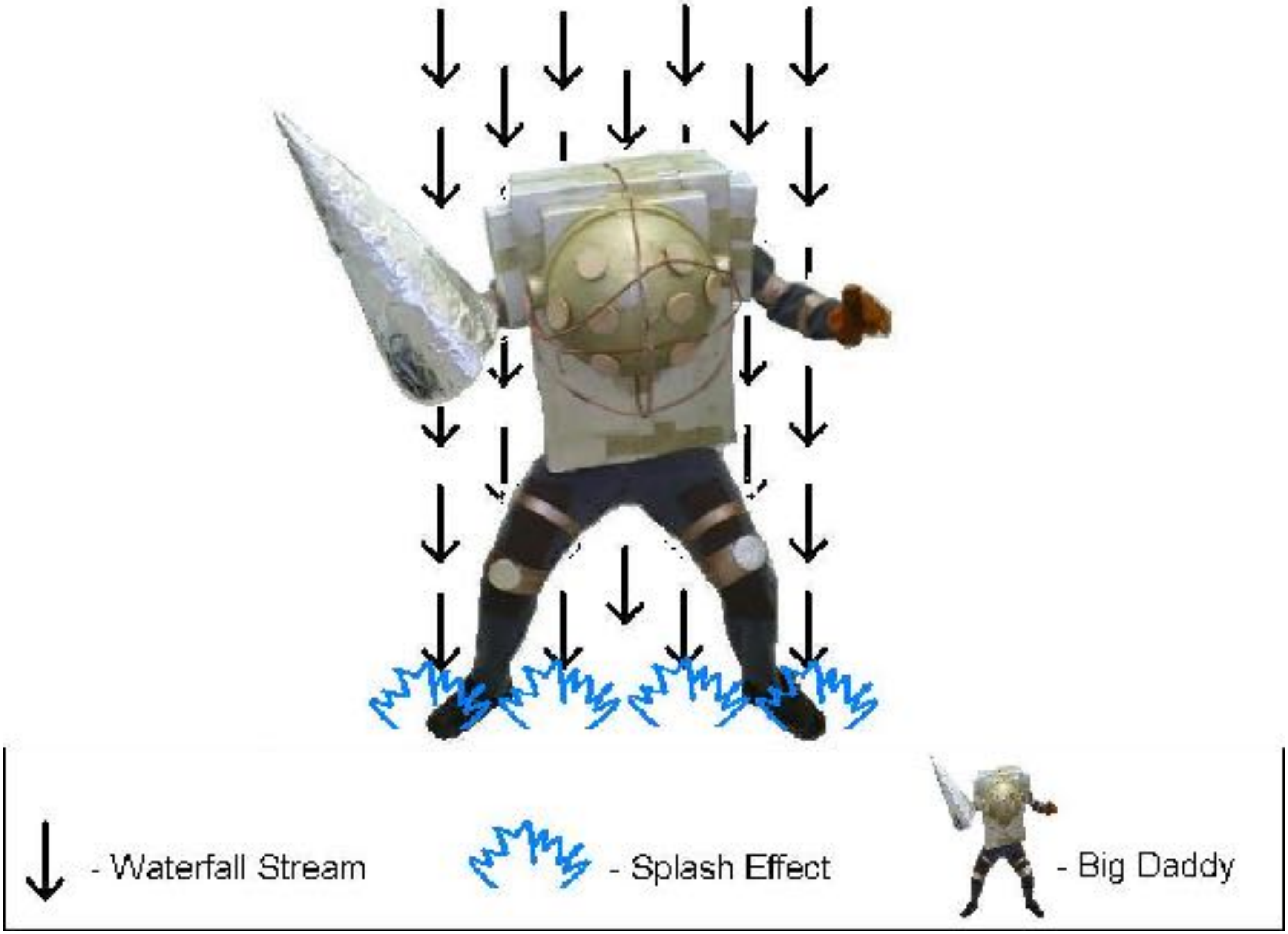



www.fraps.com
31 cur FP
30 avg FP
22 min FP
vsync on

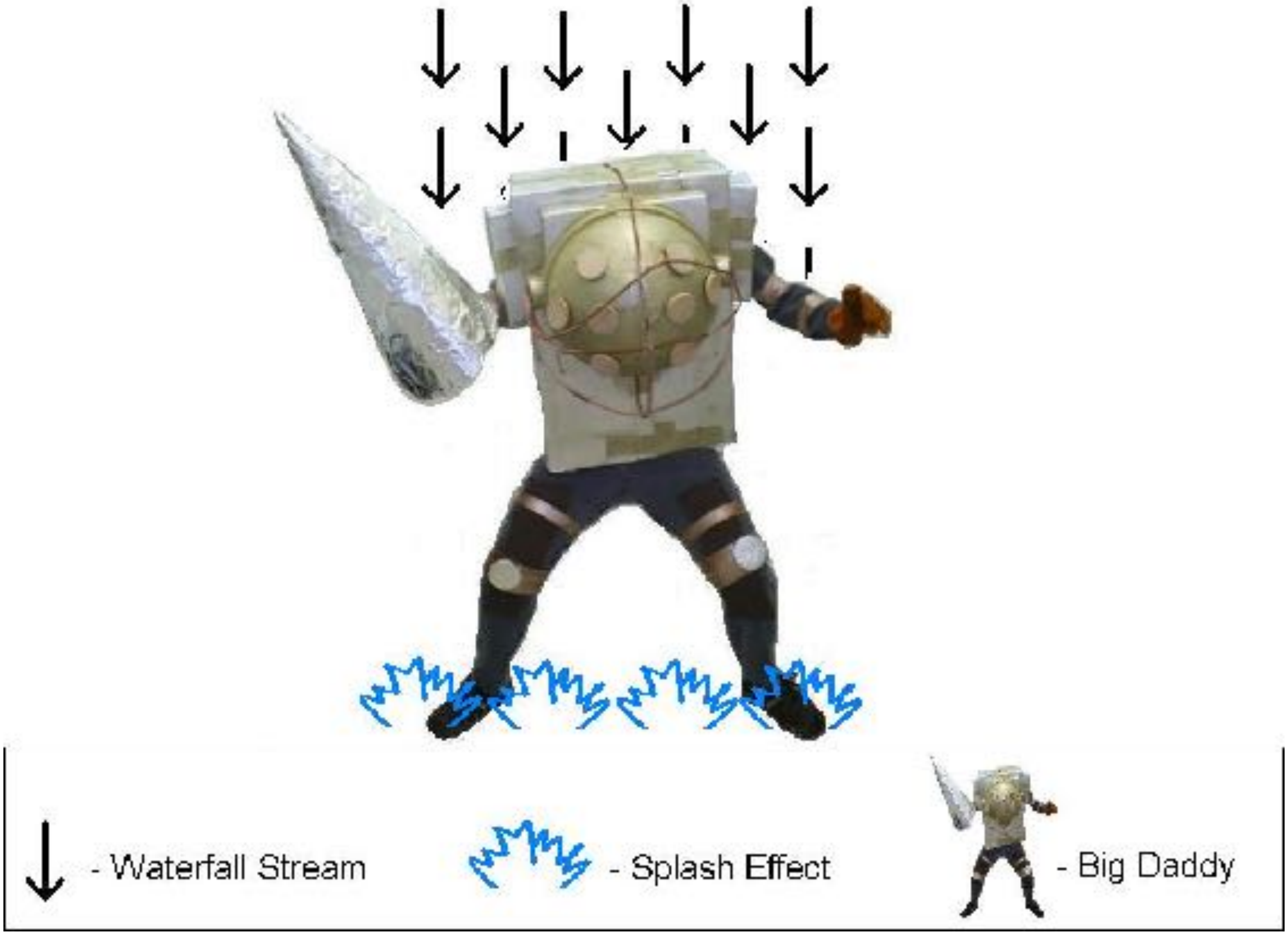
Final Implementation Textures


www.fraps.com

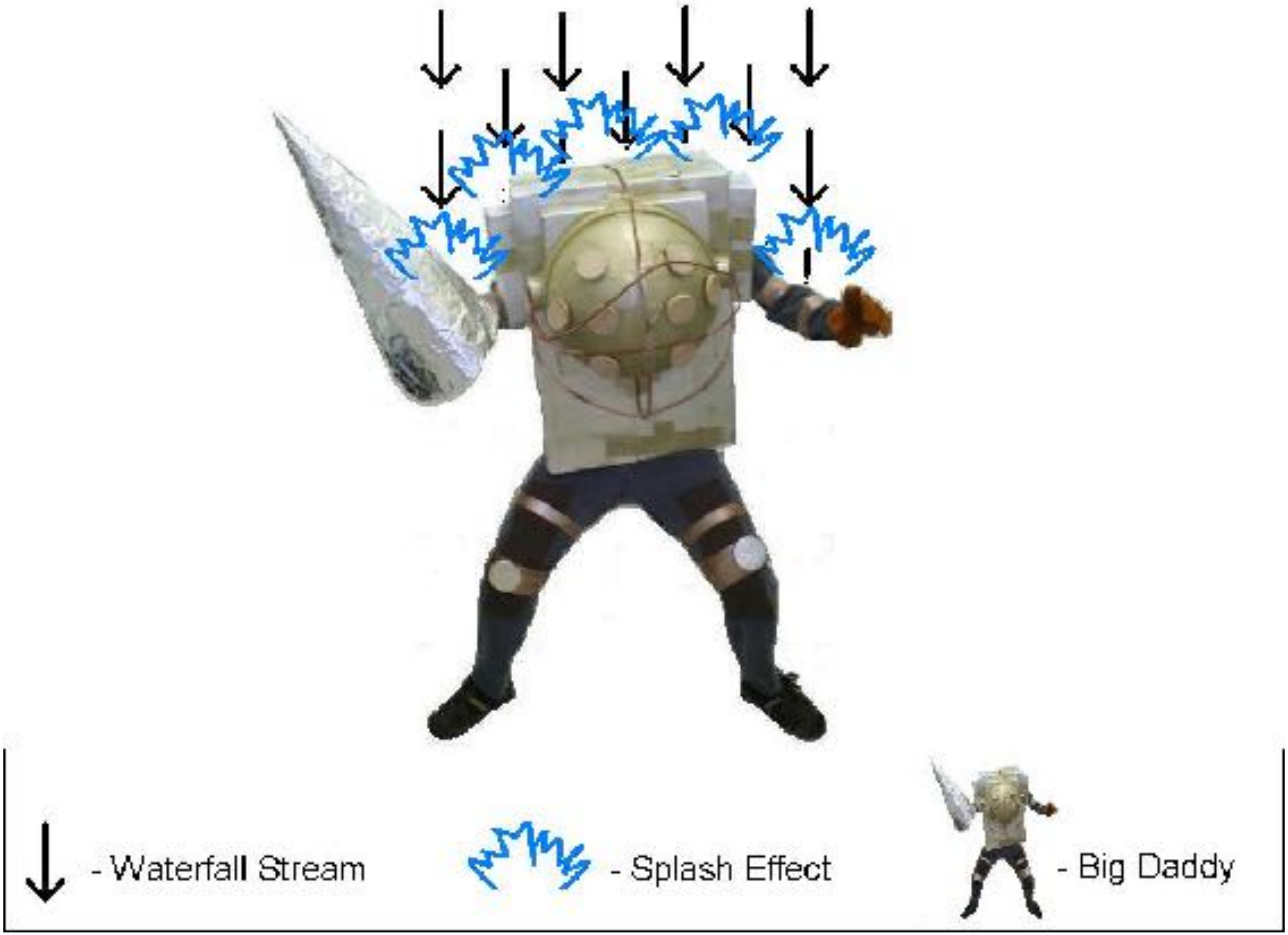
(Non) Interactive Waterfalls



(Semi) Interactive Waterfalls



Interactive Waterfalls



Interactive Waterfalls: Iteration 1

- Analogous to 1D shadow mapping
  - Camera is placed on top of waterfall, pointed down
  - Waterfall treated like 2D plane, depth values rendered into 1D row of 2D texture
- Use vertex texture fetch or ATI's R2VB to displace particle splash effects

E3 Footage

E3 Footage

Interactive Waterfalls: Iteration 2

- Added a second 1D shadow map, as seen from the bottom of the waterfall upward
  - Allowed outlines to be scissored into the waterfall
- Added a simulation step
  - Gravity!


www.fraps.com
30 cur FPS
30 avg FPS
23 min FPS
vsync on

Water Caustics

- Light that bounces off of water surfaces
- Important in creating water atmospherics

Caustics: Iteration 1

- Texture sequence
- Projectors/Decals used to place effect

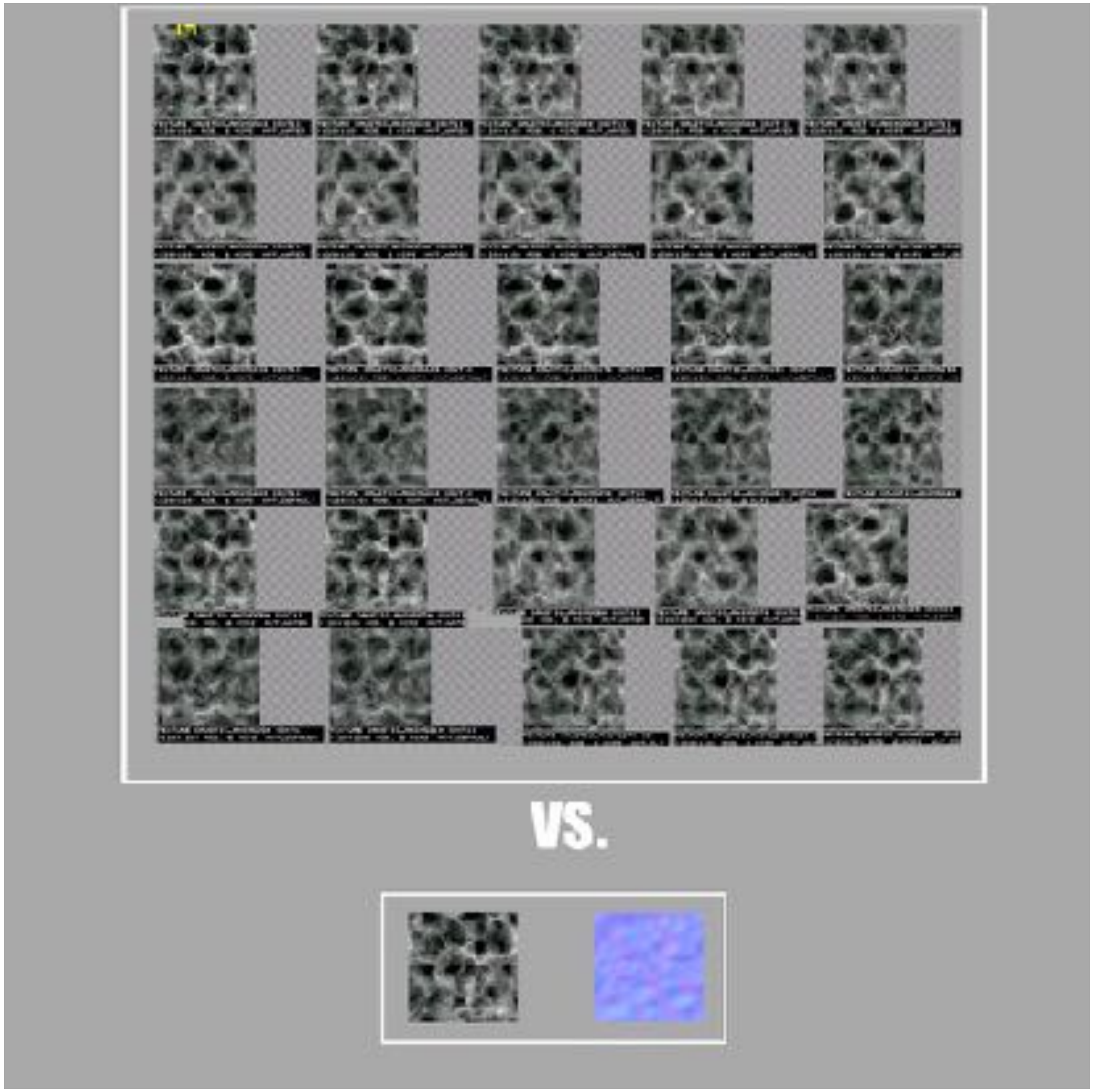

www.fraps.com
RAPTURE

Caustics: Iteration 2

- 1 Diffuse texture, sampled twice
  - .uv sample + swizzled .vu sample
- Diffuse texture UV’s offset by separate normal map

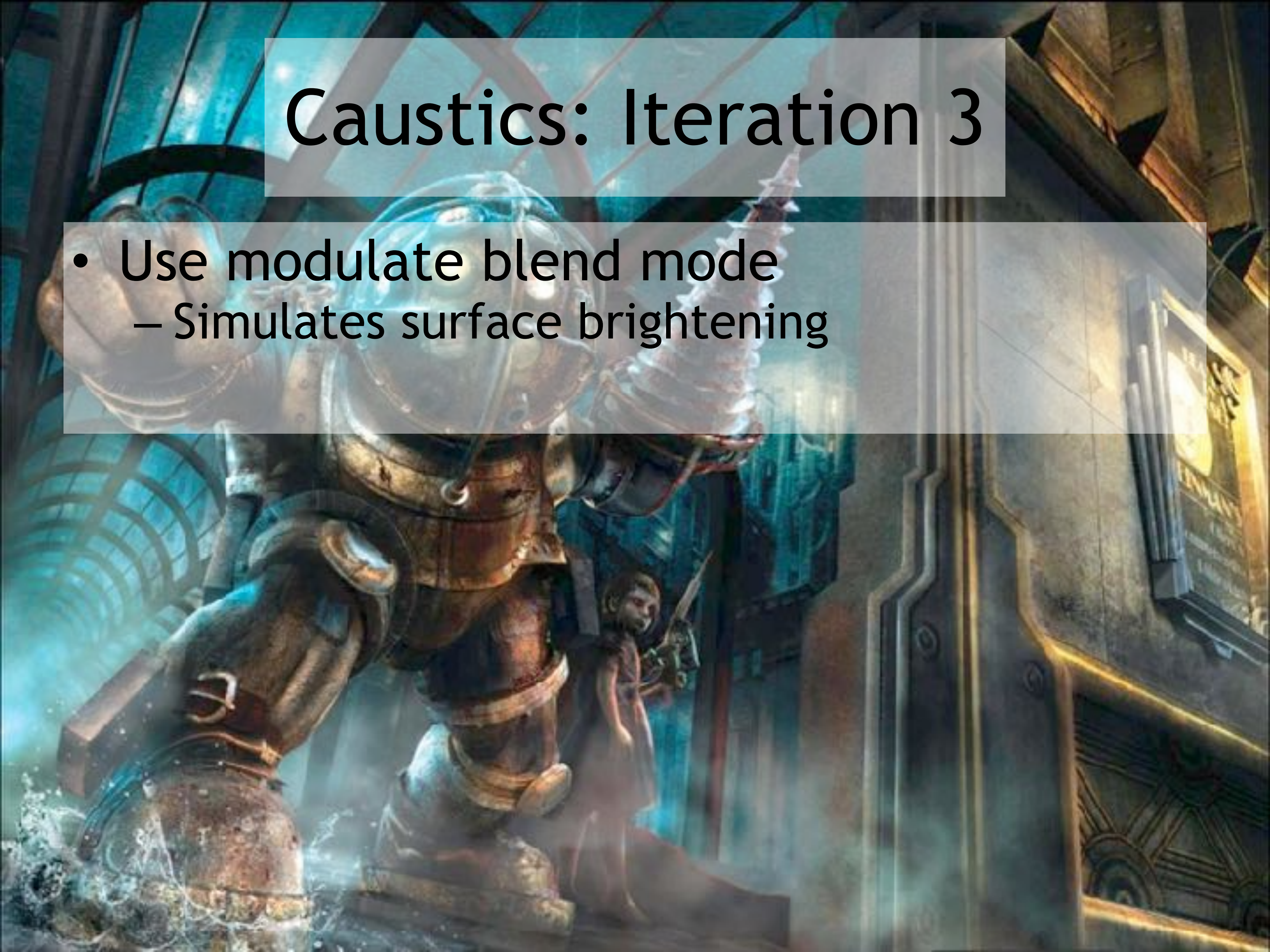

www.fraps.com

Memory Savings



Caustics: Iteration 3

- Use modulate blend mode
  - Simulates surface brightening




www.fraps.com
[DEBUG: Illumination will not affect vision]

Water = Lighting

- Controlling how light hits water surfaces is very important
- Special lighting system extension gave artists precise control over light contributions

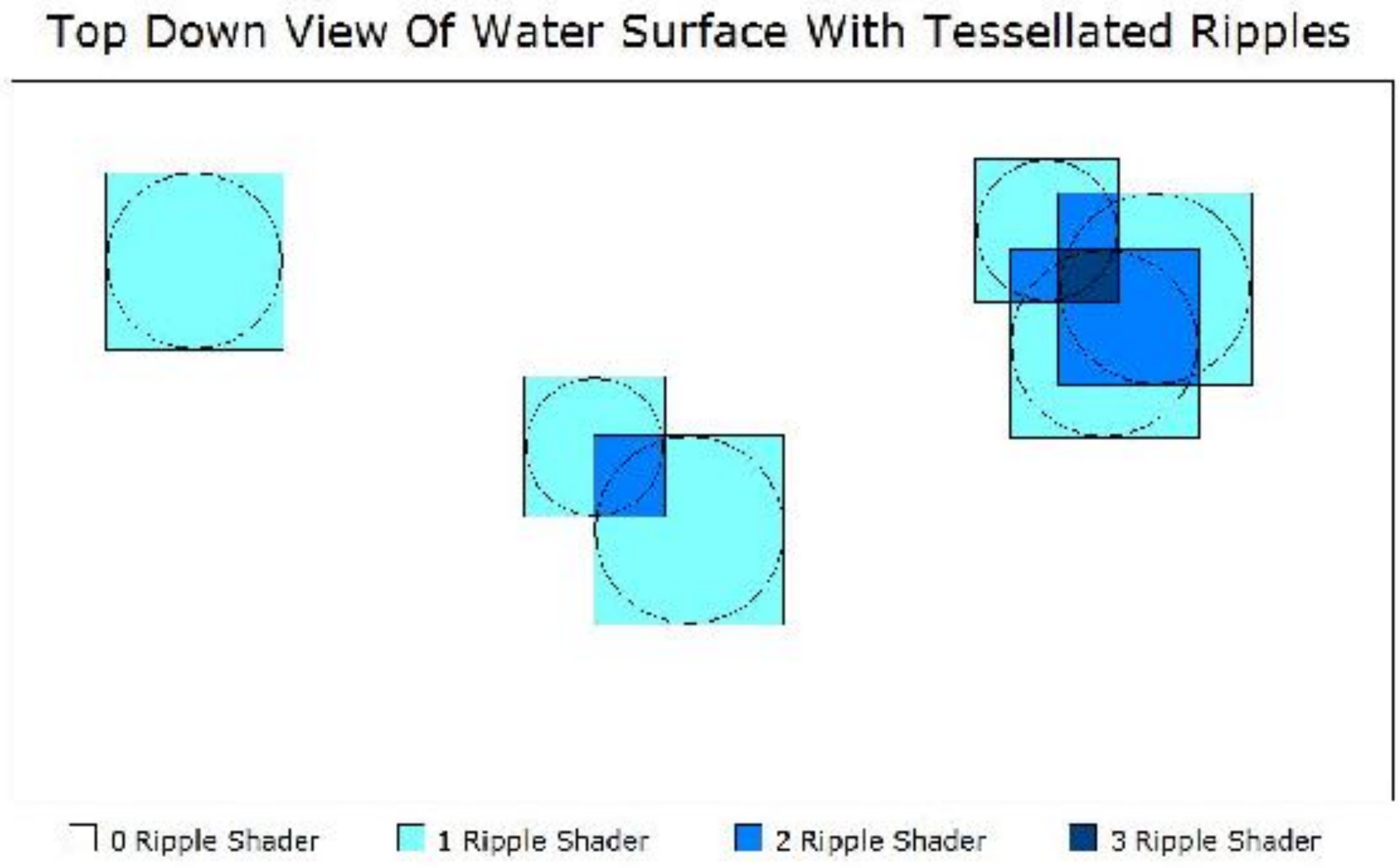
Interactive Rippling Water

- Considered four basic approaches
  - Exposing ripple locations to entire fluid surface shader
  - Dynamically tessellating fluid surfaces around ripple areas using Delaunay Triangulations
  - Dynamically tessellating fluid surfaces around ripple areas using real time 2D BSP trees
  - Deferred rendering approach

First Approach

- Bind an array of ripple locations to pixel shader
- For each pixel
  - For each ripple
    - Calculate UV's, sample ripple normal map
- Performance of every pixel bound by number of ripples on entire surface

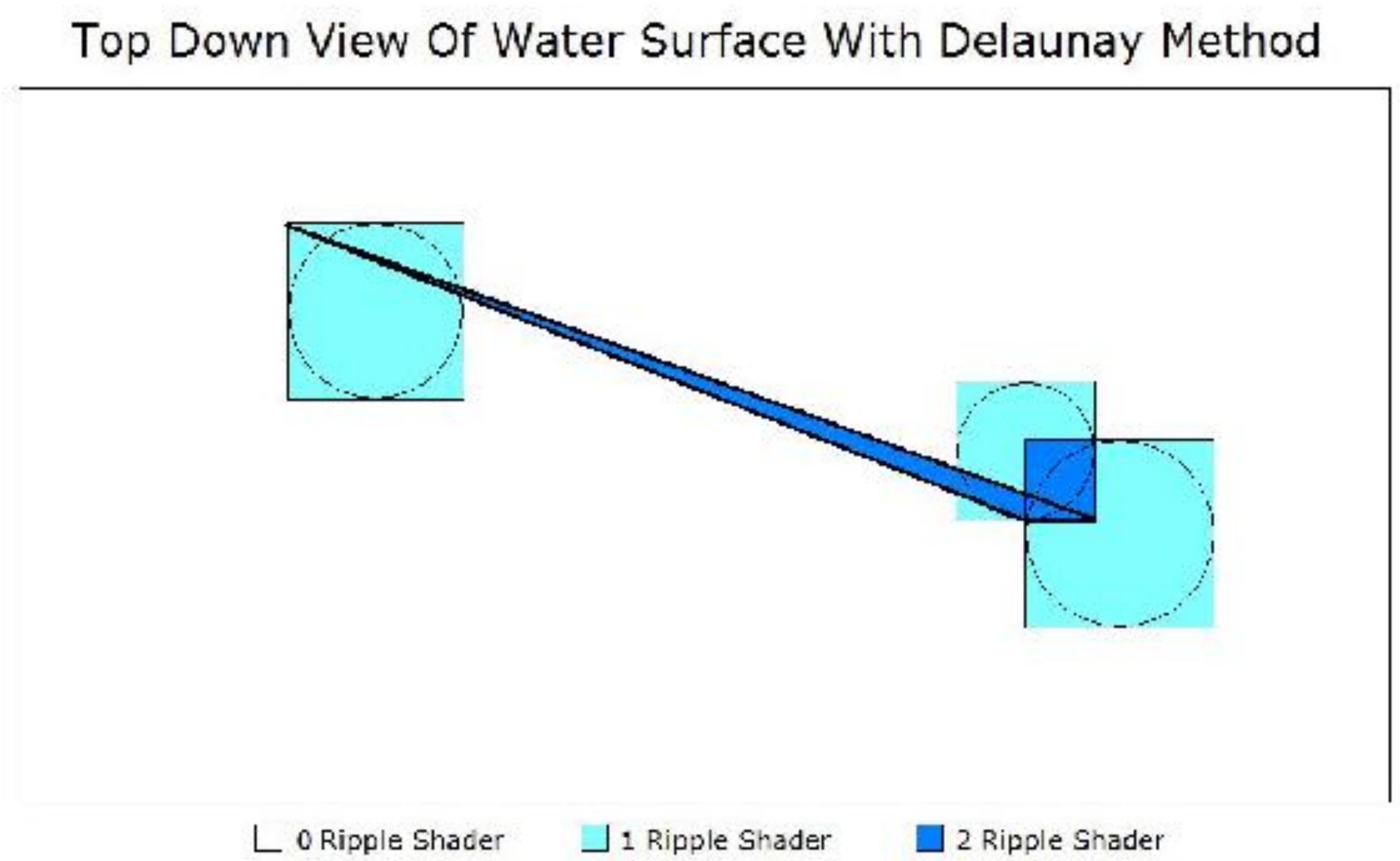
Tessellation Approach



Delaunay Triangulation Method

- Run Delaunay Triangulation using all ripple quad corners
- For each triangle generated
  - Generate UV's for all ripples the triangle intersects
- Batch triangles according to how many ripples they were affected by

Issues with Delaunay Method



BSP Method

- Build 2D BSP tree using edges of each ripple quad
- Last edge added from ripple quad is flagged to indicate child leafs are affected by that ripple
- Post build, tree could be traversed, adding new geometry for each BSP leaf.

Deferred Rendering Approach

- Render all ripple info into a ‘Ripple Buffer’
  - Same resolution as main color/depth buffers
  - Signed FLOAT16 buffer allows normal and diffuse to be stored without MRT
- Store normals in local space to the water surface
- Sample ripple buffer from fluid shaders in screen space


www.fraps.com
30 cur FPS
30 avg FPS
25 min FPS
vsync on


www.fraps.com
32 avg FPS


www.fraps.com
31 cur FPS
30 avg FPS


www.fraps.com


www.fraps.com
30 cur FPS
30 avg FPS

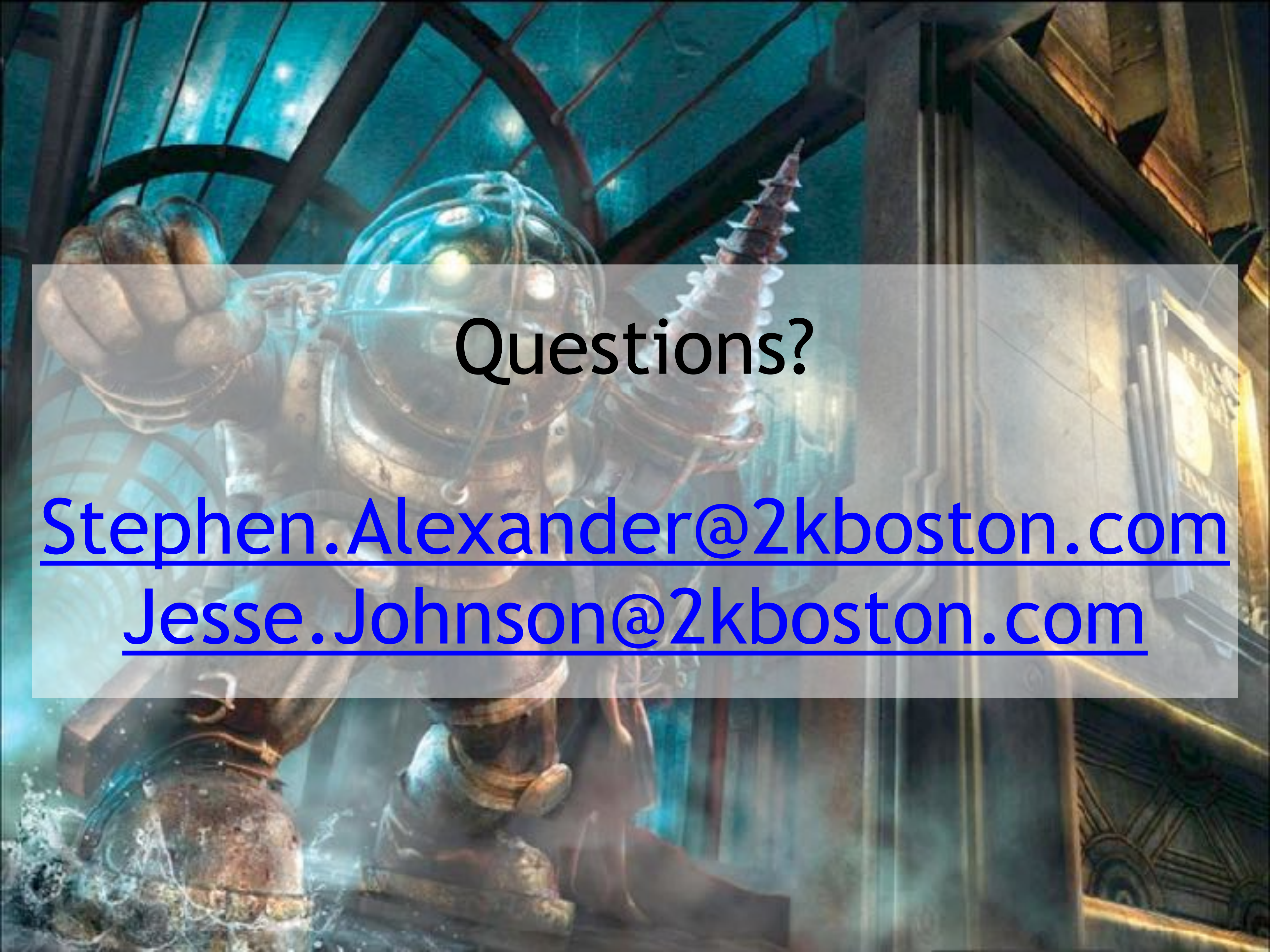

www.fraps.com
31 cur FPS
30 avg FPS

Fire Optimization

- Having art and tech on the same page during optimizations is priceless!!
- Fire optimizations:
  - Merged multiple particle emitters and lights into single emitters and lights
  - Scaled particle size, spawn rates depending on parent object size
  - Reduced particle spawn rates where possible


www.fraps.com
999
999
Incinerate!


www.fraps.com
999
999
Incinerate!



Questions?

Stephen.Alexander@2kboston.com
Jesse.Johnson@2kboston.com