MARCH 20-24
SAN JOSE, CALIFORNIA

WHAT'S NEXT
GDC:06

www.gdconf.com

GAME DEVELOPERS CHOICE AWARDS
INDEPENDENT GAMES FESTIVAL

GDC MOBILE
SERIOUS GAMES SUMMIT
GAME CONNECTION
Artist-Directable Real-Time Rain Rendering in City Environments

Natalya Tatarchuk
3D Application Research Group
ATI Research
Let Me Introduce… Myself

Natalya Tatarchuk
- Researcher
- Lead Engineer on ToyShop
- 3D Application Research Group
- ATI Research, Inc.

What we do
- Demos
- Tools
- Research
What’s in It for You?

- Share our lessons of developing an extensive environment for next generation
  - To help you in similar development goals
  - The new technology developed for this demo
- Learn what it means to render rain in city environments
  - Novel algorithms
  - Lighting and lightning
  - Raindrops and puddles
- Lots of eye candy!
The Plan

- Overview of the ToyShop demo
- Lightning system
- Wet Reflections
- Rain rendering
- Puddles and droplets on surfaces: dynamic water effects
- Conclusions
Imitating Reality:
Background Research

- **Goal**: Create a moment in a dark city, downtown, during a rainy night
- Fortunately had a lot of opportunities for research
  - Started on the concept in a middle of October in Boston
- Gigabytes of videos of rain, droplets, streams, … and here we are:

![Rainy night in Boston](image1)

![Rainy night in Toy Shop Town](image2)
Zen and The Art of Shader Writing

This demo is a study in combination of complex computations with inexpensive yet effective shaders for a variety of effects.

- Perspective-correct extruded surfaces with SM 3.0 parallax occlusion mapping

- GDC session: "Practical Dynamic Parallax Occlusion Mapping" presentation Friday 9-10am
- ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games paper "Dynamic Parallax Occlusion Mapping with Approximate Soft Shadows" (N. Tatarchuk)
Zen and The Art of Shader Writing

This demo is a study in combination of complex computations with inexpensive yet effective shaders for a variety of effects:

- Perspective-correct extruded surfaces with SM 3.0 parallax occlusion mapping
- Water and raindrop simulation effects
- A variety of post-processing effects for glow, blurry reflections and rain
- Many custom novel effects for rendering rain
- HDR illumination with lightning and shadows
  - GDC session “Shadow Mapping Tricks and Techniques” Friday 12-1 pm
- Many more…
  - GDC session: “The Ancient Temples, The Modern Cities and The Stars: GPU Journeys with the ATI Demo Team” – Wed 2:30 pm – 3:30 pm
Rendering Realistic Rain

- Frequently, rain is implemented as streaky particles in games
  - Attached to camera so that it follows the viewer
  - Doesn’t look very realistic
  - Doesn’t respond correctly to illumination in the scene
  - Requires too many particles for strong rainfall
- Simply rendering particle rain is not enough
  - Rain has a variety of visual cues, not just the streaky drops
  - Missing pieces can destroy the illusion of immersion
- We developed and combined a number of novel techniques to create this natural scene of a rainy city street at night
- All of our rain effects use our illumination model including the lightning
  - Responds dynamically and correctly to the lighting changes in the complex environment
What Does It Take To Render Rain?

- Composite Rain
- Raindrop splashes
- Wet reflections
- Raindrops off objects
What Does It Take To Render Rain?

- Water droplets on windows
- Misty objects in the rain
- Streaming water
- Water ripples and puddles
- Fog and Glow
Demo: The ToyShop
Lightning System
There’s No Thunder W’out Lightning

- Lightning and Thunder increases the feel of a rainy, stormy night
- Illumination from the lightning flashes needs to affect every object in the scene
- Uniformly aligned shadows are crucial
- At the same time, we are still using shadow mapping
- Computing lightning shadows for each additional lightning light can hurt performance
Lightning Challenges

- Lightning is a strong directional light that has to affect every object in the scene.
- Lightning effect would feel very repetitive if it only comes from one direction.
- The viewer can get very close to some of the lightning shadows…
  - Resolution has to hold up.
Solution: Lightning Maps

- Implemented lightning as 2 unique lightning lightmaps
- Encode scene lightning information into maps
- Artist editable intensity for custom mixing of these two maps
- Pre-rendered in Maya and stored as a 8-bit grayscale image
Integrating Lightning Illumination

- Every shaded pixel uses lightning illumination
- Our script propagates these to all object shaders
  - Use either or both
    - A lightning brightness parameter
    - Lightning lightmap sample added to the regular lightmap sample before tone mapping
  - Cost: 1 additional texture fetch plus a couple of ALUs
    - Cheap and effective
- All object materials respond to lightning illumination
- Translucency of water is affected by lightning strength
  - Used the lightning brightness value to adjust the pixel’s opacity as well for rain effects
Misty Glow in Bad Weather

- Water particles in the atmosphere during rain increase the amount of light scattering around objects.
- Appearance of glow around light sources in stormy weather is due to multiple scatter.
- We apply Kawase post-processing approach for rendering glows in our scene (4 feedback passes).
- Since we use 1010102 buffers for rendering, we only use 2 bits of alpha for glow amount.
  - This looks good in our demo.
- Attenuate light in shaders to simulate fog.
Wet Reflections
Wet World = Reflective World

- Wet environments display a great deal of reflectivity – without realistic reflections the illusion is broken
  - Convincing Reflections is a *must* for a rainy environment
- We render a variety of reflective surfaces in the ToyShop demo
  - Various wet surface materials (wet granite, pavement, stones, etc)
    - Since wet surfaces can easily display a lot of specular aliasing, attenuating both reflection *and* specular highlight at edges is important
Wet World = Reflective World

- Reflections within rain effects
- Planar glass reflections for the ToyShop glass windows
  - We reflected both the interior of the store and the environment outside
Wet World = Reflective World

- Reflective taxi was rendered using a dynamic cubemap
  - The contents of the cubemap were rendered only when the taxi was moving
- Stretchy warped water reflections in the street, puddles and other wet surfaces
Strong Reflections in a Rainy Scene

- Realistic streaky reflections increase the feel of rain on wet street and surfaces
  - Very prominent in any rainy scene
  - Appear to stretch toward the viewer
  - Much more saturated for brighter sources
Reflections Implementation Details

- Render the proxy geometry in separate reflection buffer
- Stretch objects in vertex shader view-dependently

![Rendered reflection objects](image1)
![Reflection objects as quads](image2)
HDR and View-Dependent Warping for Reflection Rendering

- Preserve brightest lights through an expanded dynamic range
  - Reflection buffer is also HDR (1010102)
  - Reflection buffer is half size of the back buffer
  - Preserves bright light sources and maintain contrast
- Make them appear blurry and misty
  - Use post-processing to streak in vertical direction
  - Reflections also distorted based on the normals of the surface they pass through
- Reflections are occluded by simple blocker geometry
Reflections Implementation: Result
Demo: Reflections
Rendering Rain
Post-Process Composite Rain

- Render a composite layer of falling rain as a full-screen quad
  - Rainfall is simulated with an 8bit texture
  - Render before other scene post-processing
  - Blur the rain layer to simulate strong mistiness of raindrops
- The artists specify the rain direction and speed in world-space to simulate different rainfall strength
- The difficulty lies in minimizing repeating patterns without an expensive shader
  - Remember – this is a full-screen pass
Composite Rain Rendering Algorithm

- Rain direction is moved into clip space
- Determine raindrop position using the current position (in clip space), rainfall speed and time
- Scroll a rainfall texture using rain direction and velocity

  - Scrolling is easy, but even with several texture fetches in different directions, repeating pattern becomes very obvious
Simulate Multiple Layers of Rain

- We simulate different layers of rain moving with different speeds at varied depths in a single layer rendering.

- Specify a rain parallax parameter – essentially defines the depth range for rain layers.
  - This allows us to simulate raindrops falling with different speed at different layers.

- Rain parallax value, multiplied by a random value, used as the \( w \) parameter for a projective texture fetch to sample from the rainfall texture.
  - This creates very good random streaking for the raindrops.
Composite Rain Illumination

- We use a normal map for falling raindrops to model illumination for each raindrop in the composite layer.
- Light the raindrops using the scene lights:
  - Compute full reflection / refraction approximating air-to-water transmission.
  - With Fresnel effect.
  - These are subtle but crucial effects for the soft look of rain.
- The rain must correctly respond to lightning illumination:
  - Use lightning brightness parameters to bias reflection and refraction for water effects.
  - Lightning brightness also adjusts the opacity and the glow amount.
Creating the Feeling of Strong Rain

- Realistic rain is very faint in bright regions of the scene and tends to appear stronger when light falls in a dark area.
  - If this is modeled exactly, the rain appears too faint.
- Instead, we simulate an old Hollywood trick for rain on film.
  - The film crew add milk to water to make rain appear stronger on film.
  - We do the same, by biasing rain color and opacity to appear whiter.
  - Although exaggerated, this creates a perception of stronger rainfall.
Turning Rain On and Off

- It’s important to turn rain rendering on and off.
- In game environment, very straight-forward:
  - Typically, there is an engine state that specifies which part of the environment the camera is in.
  - We can simply turn off rendering for the rain quad.
- If there is no notion of state, use “sky visibility” map approach:
  - Compute a boolean sky visibility lightmap for every point in the scene (true if sky is visible, false otherwise).
  - Then during rendering, use the look-at point to look into this lightmap.
  - This allows us to turn rain on and off.
Rendering Semi-Transparent Glowing Objects

- Controlling both opacity and glow with a single alpha blending setting can be tricky
- But we want to render transparent objects that glow, controlling each separately
- For that we used two sets of blending parameters to control blending for glow and for transparency
  - A little-known DX state for separate alpha blending: D3DRS_SEPARATEALPHABLENDENABLE
- This is a useful and efficient trick
- We used this for all rain effects (composite rain, raindrops, splashes)
To simulate raindrops falling off various objects in our scene, we used billboard particle systems.

Artist-placed and animated based on a ‘template’ system.

To render each particle:
- Stretch billboard based on velocity.
- Use a normal map for a droplet (a blurry version of a full raindrop’s normal map).
- Tangent space is defined by the view matrix.
- Only compute reflection and refraction to simulate air-to-water transmission.
- Droplets should appear more reflective and refractive when the lightning strikes.
- Biased lightning brightness value adjusts the refraction / reflection color.
Controlling Raindrop Transparency

- We attenuate raindrop opacity by distance
- Attenuate the opacity by Fresnel scaled and biased by artist-specified edge strength and bias
  - To make the raindrop appear less solid and billboard-like
- Observation: Raindrops should appear more transparent (like water) when the lightning strikes
  - Scaling the opacity by $1 - \frac{1}{2} \times$ lightning Brightness does the trick
  - The particles still appear their respective transparency when there is no lightning
  - They become more translucent-like when the lightning strikes
  - This was used for both raindrop particles and raindrop splashes to attenuate their transparency
Raindrop Splashes

- Raindrops splash when they hit solid objects.
- We simulated that effect with individual particles colliding with various objects:
  - In our pipeline, this was achieved with special collider objects.
  - In games or future engines, this should be done by directly colliding with objects.
- Used a filmed high-quality splash sequence for a milk drop:
  - We used just one splash sequence for thousands of particles:
    - The repetition can easily be noticeable.
    - To reduce that, randomly scale particle size and transparency.
    - Randomly flip $u$ texture coordinate based on pre-specified particle random color.
Illuminating the Splashes

- Splashes should appear correctly lit by the environment lights
  - If light sources were behind the rain splashes, rendered the splashes as brightened backlit objects
  - Otherwise just used simple ambient lighting
  - This worked particularly well when rendering under bright sources
  - Compute specular lighting in the vertex shader for all lights

- We used an overhead lightmap to simulate sky and street lamp lighting
  - Use the splash world-space position as coordinates to look up into this lightmap (with some scale and bias)
  - The overhead lightmap value modulated splash illumination
Misty Rain Halos on Objects

- Misty rain halos – fins with scrolling rain
- Shells – texture-based ripple effect (expanding rings)
- On the taxi and on the awning
- Requires extra geometry (fins) placed

Misty objects in the rain
Demo: Rain Effects
Puddles and Droplets on Surfaces: Dynamic Water Effects
Water Simulation for Street Puddles
Creating the Illusion of Rain: Water Effects

- Water droplets on windows
- Water swirling into a drain
- Water ripples and puddles
- Streaming water
Water Simulation and Puddle Rendering
Water Ripples in Puddles

- Goal: Dynamic realistic wave motion of interacting ripples over the water surface
- Water ripples are generated as a result of rain drops falling onto the geometry in the scene
  - This can be a direct response
- In our case, we approximate by seeding rain drops into a texture
  - Raindrop seeds are rendered as points into the water simulation texture
  - Additionally, object outlines can be projected onto the simulation texture to generate object response in water (i.e. wakes)
Simulate Water Interaction

- Real-life raindrops generate multiple ripples that interact with other ripples on the water surface
  - We implement the same model
  - Single rain drop generates ripples with damping for the duration of the splash life span. This is done with rendering a dampened sine wave when rendering each raindrop seed
Water Simulation Algorithm

- **Pass 1:** Render seeds into the first water simulation buffer
  - These seeds rendered as initial positions of water ripples
  - The seeds ‘excite’ ripple propagation
- **Pass 2 and Pass 3:** Perform integration on water surface simulation
  - Uses ‘ping-pong’ texture feedback approach
  - We only use two passes, but more will help with system stability if time step desired to be smaller
  - These passes generate water height field
- **Pass 4:** Generate water normals
  - Sample from the water normals texture when rendering an object with puddles
Water Surface Approximation

- Approximate water surface with a lattice of points
  - We render our surfaces with a 256 x 256 simulation
  - Each lattice contains information about water surface at that point
    - Current position as a height value
    - Previous time step’s position
- We simulate the water lattice entirely on GPU
  - Using a texture to store lattice positions and its attributes
  - Similar to “Interactive Simulation of Water Surfaces” by M. Gomez (Game Programming Gems)
  - However, there the lattice is approximated with vertices on the CPU

Water lattice heights: Current frame’s height in R channel and previous frame’s height in G channel
Water Surface Response

- Treat water surface as a thin elastic membrane
  - Ignore gravity and other forces
  - Only account for surface tension
- At every time step, infinitesimal sections of this surface are displaced
  - Due to tension exerted from their direct neighbors
  - Acting as spring forces to minimize space between them
Computing Ripple Heights

- Vertical height of each water surface point can be computed with partial differential equation:

\[
\frac{\partial^2 z}{\partial t^2} = v^2 \left( \frac{\partial^2 z}{\partial x^2} + \frac{\partial^2 z}{\partial y^2} \right)
\]

where \( v \) is the velocity of the waves traveling across the surface

- Solve this equation in real-time to determine water wave height for each point in the lattice

- PDE solution is computed with Euler integration in pixel shaders
Water Puddles Integration

- We render a single water simulation for the entire demo
  - All objects with water puddles sample from that (for example, streets, rooftop ledge, etc)
- Sample from water membrane simulation using current position
  - Use the (xz) world-space coordinates for look-up
  - Scaled by the artist parameter to vary by size of membrane simulation (and size of ripples)
  - To reduce visual repetition, we rotate these coordinates by a pre-specified angle (ex: 15°)
  - Angle is specified per-object
- No additional geometry is required for water puddles
Puddle Placement and Depth

- To render deep puddles, we use just the water puddle normal sampled as just described, along with color / albedo attributes of the object.
- However, in real environments, puddle depth varies greatly.
- To simulate that, we allow a puddle depth and location mask map.
- Adding puddles with ripples to objects:
  - Define scale parameter and sample ripple normals.
  - Sample puddle depth map.
  - Interpolate between the normal map for the object and the water surface normal based on the puddle depth value.
Creating Swirling Water Puddle

- Create an impression of water, swirling towards the drain, with ripples from the raindrops.
- To create ripples from raindrops, use the same approach as for puddles on the street.
- For water, swirling toward the drain, we used several ‘wake’ normal maps:
  - Swirling radially around the drain.
  - Concentric circles draining toward the drain.
Controlling Rooftop Puddle Depth

- Puddle depth is painted with the vertex color
  - Used to make the water level merge seamlessly with the material on the roof
  - Careful attention was paid to matching materials on the rooftop floor and the edges of the puddle
Demo: Puddles, Ripples and Swirls
Water Droplet Movement on Glass
Raindrop Movement on Glass Surfaces
Simulation of Raindrop Movement

- Modification of “Animation of water droplets moving down a surface” [Kaneda99]
- The surface of the glass is represented by a lattice of cells
- Each cell contains:
  - the mass of water in each cell
  - velocity in x and y
  - the droplet traversal within the cell.
- Convenient to pack into .rgba 16-bit per channel texture
Droplet Simulation: Forces

- **Gravity** and mass are used to compute the downward force on the droplet.

\[ F_{\text{down}} = M \times G \]

- **Static friction** for stationary droplets, and **dynamic friction** for moving droplets is used to compute the competing upward force.

\[ F_{\text{up}} = f_{\text{static or dynamic}} \]

- The static and dynamic friction varies over the surface of the glass.

- This resultant force is applied to the initial velocity to determine the new velocity value for the droplet.

\[ V_{\text{new}} = V_i + \frac{F}{M} \times t \]
Droplet Movement Simulation

- Droplets can flow into one of the 3 cells below.
- New cell to flow into is randomly chosen biased by velocity, friction based affinity and wetness of the target cell.
- Droplets have a greater affinity for wet regions of the surface.
- Velocity is updated based on target cell chosen.
Droplet Rendering

- After simulation, each cell contains a new mass value
- A bump map is derived based on this mass
  - This is used to perturb reflection & refraction vectors
- The droplet mass is also used to render dynamic shadows of the simulation onto the objects in the toy store
  - If the droplet mass is large enough, we render a ‘caustic’ highlight in the middle of the shadow for that droplet
Rain Effects - Windshield Wipers

- Wiper shader with droplets on glass of car
- Wiper parameter is passed into the shader
- Wiper maps are used to determine what regions have been recently swept clean
- Two separate wiper maps so wiped regions can overlap
Demo: The Water Droplet Simulation
ToyShop Revealed

- We presented you some of the technology that was developed for the ToyShop demos
- Rich, complex environments demand convincing details
- Many custom shaders allowed us to create a thorough illusion of a dark, rainy night in the city
  - Different approaches for reflection rendering
  - A variety of rain effects
  - Water and raindrop movement simulation effects
- We hope to see even better environments in next gen games
  - The secrets are out: Use all of our technology to your benefit!
The ToyShop Team

Lead Artist  Lead Programmer
Dan Roeger  Natalya Tatarchuk
            David Gosselin

Artists
Daniel Szecket, Eli Turner, and Abe Wiley

Engine / Shader Programming
John Isidoro, Dan Ginsburg, Thorsten Scheuermann and Chris Oat

Producer  Manager
Lisa Close  Callan McInally
More ToyShop Secrets at GDC

- Session “Practical Dynamic Parallax Occlusion Mapping”
  Programming Track
  Friday 9-10am

- Session “Shadow Mapping Tricks and Techniques”
  Programming Track
  Friday 12-1pm

- Session “The Ancient Temples, The Modern Cities and The Stars: GPU Journeys with the ATI Demo Team”
  Programming Track
  Wednesday 2:30pm – 3:30pm

- Session “ToyShop – The Devil is in the Details”
  Artist Track
  Wednesday 10-30am-11:30am
### Other Interesting Sessions by ATI

#### Wednesday

<table>
<thead>
<tr>
<th>Time</th>
<th>Session</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:00 – 10:00AM</td>
<td>Squeezing Performance out of your Game with ATI Developer Tools</td>
</tr>
<tr>
<td>10:30 – 11:30AM</td>
<td>ToyShop – The Devil is in the Details (Art Track)</td>
</tr>
<tr>
<td>12:00 – 1:00PM</td>
<td>Efficient Tricks That Will Impress Your Friends! (Programming Track)</td>
</tr>
<tr>
<td>2:30 – 5:00PM</td>
<td>The Ancient Temples, the Modern Cities, and the Stars: GPU Journeys</td>
</tr>
</tbody>
</table>

#### Friday

<table>
<thead>
<tr>
<th>Time</th>
<th>Session</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:00 – 10:00AM</td>
<td>Practical Parallax Occlusion Mapping (Programming Track) (ToyShop)</td>
</tr>
<tr>
<td>12:00 – 1:00PM</td>
<td>Shadow Mapping Tricks and Techniques (Programming Track)</td>
</tr>
</tbody>
</table>
Reference Material

- [www.ati.com/developer](http://www.ati.com/developer)
  - All of these presentation and related materials
Questions?

سائل؟

natasha@ati.com