Acoustics improves immersion

- We understand our surroundings audio-visually
- Can hear behind head, around corners: persistent awareness
- Need good acoustics in games

VR/AR –

Positional Audio = Directional Audio + Environmental Acoustics
Game acoustics today

• Compared to lighting, CPU/RAM are limited...

...but sound’s behavior is more complex: wave effects crucial

• Mix of approximate methods w/ pitfalls and manual tagging

• Designer: Tedious work, limited time for creative expression

• Why not follow the path of game lighting?

Reliable physics + design
Part I: Accurate acoustic simulation = robust results for dynamic sources & listener. But expensive!

@GDC 2011: Real-time wave acoustics: full core, GBs of RAM, custom audio engine.
Raghuvanshi, et. al., SIGGRAPH 2010, “Precomputed wave simulation for real-time sound propagation of dynamic sources in complex scenes”

Part II: How to design acoustic data & implement expressive audio?

Triton is the first shipped system to meet both challenges
Triton’s solution

Part I: Bake **accurate physics** on scene geometry

Acoustic data

Part II: Implement and **express** audio design goals

**Goal:** Combine *fast* wave acoustics with *expressive* audio design to bring game spaces to life
Wave acoustics

- Diffraction: Sound waves bend, we hear around corners.
- Reverberation: We hear sound bouncing around 100 m.
Important acoustic effects

1. Obstruction
2. Occlusion
3. Wetness ratio
4. Reverb decay rate

...
Effects 1 & 2: Obstruction & occlusion

Obstruction ~ initial (direct) sound energy
Occlusion ~ initial (direct) + reflected (indirect)

Strong obstruction, little occlusion

Strong obstruction, strong occlusion
Effect 1: obstruction (initial energy)

Existing approximate method: Shortest path length

Issue: Even initial energy needs many paths!

Upshot: Risk of making sound too weak

wave scattering
Effect 2: Occlusion?

Fast approximate method does NOT exist!

Need wave diffraction!

Ignore diffraction → sound too weak
Missing paths → glitches

Fast approximate method does NOT exist!
Effect 3: Wetness ratio (sense of distance)

Wetness \sim \text{ratio of reflected and initial energy}

Common method: Designer specifies reverb distance roll-off curves

Room size and effect of geometry are ignored!

High wetness = "sound is in adjoining room"
Triton OFF

You do NOT have debug menu enabled

1. Gear/MainMenuEntry_D
2. GF_Form_B_D
3. Gear/MainMenuEntry_D
4. GF_Form_B_D
5. Gear/MainMenuEntry_D
6. GF_Form_B_D
7. Gear/MainMenuEntry_D
8. GF_Form_B_D
2. checkpoint reloads
Initial Energy only $\rightarrow$ Obstruction

Reflected Energy only $\rightarrow$ Wetness ratio

Total energy $\rightarrow$ Occlusion
Effect 4: Decay rate

Reverb volumes are tedious to draw.
Effect 4: Decay rate

- Decay rate also depends on both source and listener locations!
Consistent reverb: source *and* listener
Acoustics conveys location
Important acoustic effects

1. Obstruction
   - Depend on *both* (moving) source and listener location

2. Occlusion
   - Depend on scene geometry

3. Wetness ratio

4. Reverb decay rate
   - Triton can model them
Triton = baked wave simulation

- Wave simulation: **Accurate & reliable** results on complex scenes
- Runtime = lookup + interpolation. **Light on CPU.**
- Need *dynamic* source & listener: **large RAM!**
- Baking is restricted to static geometry
  - Feasible first step
  - Dynamic scenes (doors/destruction): could layer heuristics on top, like lighting
Illustrated pipeline: Scene

Offline bake tools with input –

• 3D map geometry as FBX
• Per-triangle material name
• Nav-mesh geometry
Start baking: Voxelize!
How to guide player sampling? Boxes?
Nav mesh to the rescue!
Auto-layout adaptively-sampled player probes
Triton map bake =
For each player probe, 3D wave simulation

200× slomo, horizontal slice

100 m
Main Idea: Response \(\rightarrow\) 4 perceptual parameters
[for each source & listener location pair]

- Initial ("direct") Energy
- Reflections Energy
- Reflections decay rate
- Reverb decay rate

(time, loudness)
Per-player-probe data: loudness

Initial energy (0-10ms)

Reflected energy (10-210ms)
Per-player-probe data: early sound decay

after 250ms
50 TB → 100 MB
Baking for moving sources & listener is costly

• 100 machines $\rightarrow$ ~4 hours
  – ~10-20 minutes per player probe
  – ~1000-1500 player probes per Gears campaign map

• Trivially parallel: double machines, half time

• Bake tool runs on PC/Xbox (latter as “bake game”)
Runtime: Interpolated lookup

- For each parameter: Decompress slice (Zlib) & do interpolation
- Fixed caches, 20MB overhead
- Worker thread on shared core
- Updated at visual frame rates [30/60FPS in single/multi-player]
- Averages ~100μs per query, spikes of ~200μs on cache miss
- 32 sources served per-frame ~10-20% of shared Xbox core
Game integration

Baked acoustic data

QueryAcoustics()

Triton lib

Game

Perceptual Data
- initial energy
- reflections energy
- reflections decay rate
- reverb decay rate

Wwise reverb sends, occlusion/obstruction

Interpret/design

Source pos.  Player pos.
Wave baking + perceptual compression = practical acoustics

<table>
<thead>
<tr>
<th>Technical</th>
<th>Efficient</th>
<th>Robust</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU: $\sim 100 \mu s$ for acoustic lookup</td>
<td>Smooth results on complex level geometry</td>
</tr>
<tr>
<td></td>
<td>RAM: $\sim 100$MB for Campaign, $\sim 20$MB for Multiplayer</td>
<td></td>
</tr>
<tr>
<td>Design</td>
<td>Automation</td>
<td>Expressivity</td>
</tr>
<tr>
<td></td>
<td>No geometry to clean, no volumes to draw</td>
<td>Interpret Triton data via Unreal blueprint and Wwise</td>
</tr>
</tbody>
</table>

Paper reference: Raghuvanshi & Snyder, SIGGRAPH 2014, “Parametric wave field coding for precomputed sound propagation”
Triton’s design

Part I: Bake **accurate** physics on scene geometry

Acoustic data

Part II: Implement and **express** audio design goals

Combine **fast** wave acoustics with **expressive** audio design
to bring game spaces to life
How does a Sound Designer work with this?
What Triton Is

• Listener and Source Positions ➔ Occlusion/Obstruction, Reverb Wetness, and Reverb Decay Rate values.
• This data is inert and must be interpreted.
• The RAM requirements are next-gen.

What Triton Isn’t

Triton is a Data-Set //not a ‘Reverb Plug-in’
Triton Implementation Challenge

The ideal Triton implementation with no CPU/RAM limitations

- Per-Sound Impulse Responses Updated Every Frame
  no middleware support, over RAM budget [10-100GB] 😞

- Per-Sound Convolution Reverb
  ~100 instances; way over CPU budget 😞

Not practical for video game audio...
Implementation Progression

- Triton Lookup (per 3D Sound)
- Occlusion
- RoomVerb [Large Spaces/~3s dcy]
- RoomVerb [Medium Spaces/~1s dcy]
- RoomVerb [Small Spaces/~0.5s dcy]
- Obstruction / Dry Gain
- Main Mix

Yellow circle = Triton controlled

Send levels = variable reverb decay per 3D Sound!
Implementation Progression

V1.0 – Summary:

- Triton Reverb Data controls levels sent 3 instances of Wwise RoomVerb.
- Occlusion/Obstruction uses Wwise built in architecture.


- Weaknesses: No solution for outdoor reverb, RoomVerb sounds crowded when stacked, unpredictable behaviour ~25% of the time. Not yet shippable.
Occlusion: firefight
**V2.0**
+ Convolutions
+ Outdoor reverb
+ ER/LR decay

**Triton Lookup Results**  
(per 3D Sound)

**Indoor**
- Reflections conv Indoor, small
- Reflections conv Indoor, medium
- Reflections conv Indoor, large

**Outdoor**
- Reflections conv outdoor, small
- Reflections conv outdoor, medium
- Reflections conv outdoor, large

**Occlusion**
- Reverb conv Indoor, small
- Reverb conv Indoor, medium
- Reverb conv Indoor, large

**Dry Gain / Obstruction**
- Reverb conv outdoor, small
- Reverb conv outdoor, medium
- Reverb conv outdoor, large

= Triton controlled

Ø Dynamic Triton-controlled send levels

ER = Early Reflections  
LR = Late Reverb tail

Main Mix
V2.0 – Summary:

• Triton Data controls levels to 12 Wwise Convolution Reverbs.

• Strengths: Indoor/Outdoor threshold blending is excellent (video example), dynamic reverb decay response

• Weaknesses: Setup and calibration, bugs were difficult to stomp
Triton Lookup Results (per 3D Sound)

- **Indoor**
  - Reflections conv Indoor, small
  - Reflections conv Indoor, medium
  - Reflections conv Indoor, large

- **Occlusion**
  - Reflections conv outdoor, small
  - Reflections conv outdoor, medium
  - Reflections conv outdoor, large

- **Outdoor**
  - Reflections conv outdoor, small
  - Reflections conv outdoor, medium
  - Reflections conv outdoor, large

- **Dry Gain / Obstruction**

- **Main Mix**

- **V2.0** + Convolutions + Outdoor reverb + ER/LR decay

ER = Early Reflections
LR = Late Reverb tail

= Triton controlled

= Dynamic Triton-controlled send levels
Mixing Indoor/Outdoor Reverbs Through Thresholds

Problem: This concept is not binary.

Solution: Triton pre-computes a smooth “Outdoorness”
Solution: \[ \text{Outdoorness} = \frac{\text{Energy reaching sky}}{\text{Total Energy shot}} \]

Outdoorness = 0.13...
Outdoorness changes **smoothly** as player approaches transition

Outdoorness = 0.72...
Unforeseen Uses: Threshold Blending for Rain and Wind

- Triton uses this for determining sends for inside/outside reverb blending
- **Exposed as RTPC:** Driving sound of Wind, Rain, etc.
V2.0 Generated some uncomfortable questions:
Triton Reverb worked well sometimes—why?
Time investment for shippable quality?

Storytime...

One day, we were given a deadline by Rod Fergusson (Gears Co-Creator/Studio Head) and Chuck Osiega (Creative Director):

“Get Triton Reverb sounding shippable on one map in 2 weeks or we switch to a proven conventional reverb solution.”
Change of Strategy

1. Reduce number of sounds going into Triton Reverb.
2. Reduce the number of reverb outputs.

Simplification revealed root causes of the problems.

Triton-calculated Occlusion/Obstruction remained enabled for almost all 3D sounds.
V2.5

+ Convolutions
+ Outdoor reverb
+ ER/LR decay

Triton Lookup Results (per 3D Sound)

Inside

- Reflections + Reverb, small
- Reflections + Reverb, med
- Reflections + Reverb, large

Occlusion

Outside

- Reflections + Reverb, small
- Reflections + Reverb, med
- Reflections + Reverb, large

Dry Gain / Obstruction

= Triton controlled

Ø Dynamic Triton-controlled send levels

Main Mix

ER = Early Reflections
LR = Late Reverb tail
Root Causes of our Triton Challenges

1. Physical dynamic range vs. Aesthetic dynamic range
2. Dry Gain added to point source sounds
3. Physics-based vs. Aesthetics-based reverb
   wetness/decay times: Emotionally Interpreted Spaces

A Shift in Thinking:
How “Real” is “Too Real” for Game Acoustics?
A Shift in Thinking: How “Real” is “Too Real” for Game Acoustics?

Triton datasets are formed with reality based calculations. BUT: *Not all physical aural phenomena are creatively desirable all the time!*

Lesson 1: Dynamic Range

Triton’s ‘real world’ dynamic range needs interpretation into a smaller gameplay range.

Sound designers in film and games have ALWAYS made this interpretation to create a mix. Realizing that we needed to apply this same interpretive process to Triton-Reverb inputs was crucial to shipping.
<table>
<thead>
<tr>
<th>Decibel/SPL Range Chart</th>
<th>Real World Sounds</th>
<th>Gears 4 Sounds in &quot;real world dB range&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>0dB</td>
<td>Threshold of human hearing</td>
<td></td>
</tr>
<tr>
<td>10dB</td>
<td>Breathing</td>
<td></td>
</tr>
<tr>
<td>20dB</td>
<td>Whispering</td>
<td></td>
</tr>
<tr>
<td>30dB</td>
<td>Room tone/Light ambient tones</td>
<td>Quiet Level Ambience</td>
</tr>
<tr>
<td>40dB</td>
<td>Typical World Sound (Fire, Trees in Light Wind… etc.)</td>
<td>Squad Speech</td>
</tr>
<tr>
<td>50dB</td>
<td>Conversation</td>
<td></td>
</tr>
<tr>
<td>60dB</td>
<td>Busy street</td>
<td></td>
</tr>
<tr>
<td>70dB</td>
<td>Vacuum, noisy restaurant</td>
<td></td>
</tr>
<tr>
<td>80dB</td>
<td>Kitchen Blender</td>
<td>Loud World Sound (Large Machinery… etc.)</td>
</tr>
<tr>
<td>90dB</td>
<td>Lawn Mower</td>
<td></td>
</tr>
<tr>
<td>100dB</td>
<td>Helicopter 30m above</td>
<td>Kestrel</td>
</tr>
<tr>
<td>110dB</td>
<td>Loud Rock Concert</td>
<td></td>
</tr>
<tr>
<td>120dB</td>
<td>Thunderclap (close range)</td>
<td></td>
</tr>
<tr>
<td>130dB</td>
<td>Gunshot</td>
<td>Lancer</td>
</tr>
<tr>
<td>140dB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>150dB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>160dB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>170dB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>180dB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>190dB</td>
<td>Grenade (close range)</td>
<td></td>
</tr>
<tr>
<td>200dB</td>
<td>Eardrum Rupture</td>
<td></td>
</tr>
</tbody>
</table>

~35dB of usable Dynamic Range
Final adjustment of occlusion/obstruction curves to account for game-relative dynamic range.

Physics based dynamic range: 100dB
Game based dynamic range: 25dB
A Shift in Thinking: How “Real” is “Too Real” for Game Acoustics?

Lesson 2: Dry Gain

Small enclosed spaces can amplify dry sound in reality.

When simulated in gameplay, this can sound unexpected.

**Solution:** The ceiling of this value was originally +12db - we clamped this gain to +3dB
A Shift in Thinking: How “Real” is “Too Real” for Game Acoustics?

Lesson 3:

Triton simulations can violate expectations:

• small spaces can generate very long decay times.
• large spaces can generate short decay times.

In games, we have emotional requirements for reverb to inform storytelling.

Solution: Scriptable designer-based interpretations of the Triton data. AKA The “Triton Tweak Acoustics node”.

Solution: Scriptable designer-based interpretations of the Triton data. AKA The “Triton Tweak Acoustics node”.
Designer crafted spaces: reverb/rain transition
Designer crafted spaces: reverb/rain transition
Reflections & Reverb conv
Indoor, small

Indoor
Occlusion

Dry Gain (< 3dB) / Obstruction

Main Mix

[ER = Early Reflections, LR = Late Reverb tail]

Final
+ Convolutions
+ Outdoor reverb
+ Reverb decay

Triton Lookup Results (per 3D Sound)

Outdoor

Dynamic Triton-controlled send levels

Reflections +Reverb conv
Outdoor, small

Reflections +Reverb conv
Outdoor, large

Reflections +Reverb conv
outdoor, medium

Reflections +Reverb conv
outdoor, large

Reflections +Reverb conv
Indoor, medium

Reflections +Reverb conv
Indoor, large

Reflections +Reverb conv
Indoor, small

[Ø] = Triton controlled
Gears 4 Triton Implementation: Final

Lessons Learned; Summary

• Tuning and Testing of this unprecedented system was our biggest risk.
  – Reduce number of Wwise parameters Triton is driving. Simplify! Limit Triton Reverb to a sensible dB range.
• Focus on the *emotionally* motivated “Cinematic Experience” instead of the reality-inspired “Realistic Experience”.

Our Trip to The Uncanny Valley...

We've only scraped the surface of what's possible with Triton.
ACT IV - CHAPTER 2
NO DETOURS
Conclusion

• Baked wave acoustics for dynamic sources & listener is now practical
• Designed occlusion, obstruction, reverb wetness, and decay times in production game levels
• Key idea: Do accurate physics $\rightarrow$ perceptual data $\rightarrow$ game
• Future
  – Next: directional initial sound and early reflections, outdoor echoes, streaming 100$\rightarrow$25MB RAM, faster bakes
  – Portability
  – Longer term: dynamic geometry e.g. doors/destruction
Special Thanks...

John Morgan (Audio Director),
Mike Rayner (Technical Lead),
Rod Fergusson (Studio Head),
Chuck Osieja (Creative Director)
...for believing we could pull it off.

Jimmy Smith (Lead Audio Programmer),
Clarence Chu (Audio Programmer),
Jess Kube (Tools Programmer),
Michael Oliver (Intern Programmer)
...for their indispensable work.

John Snyder (Principal Researcher),
...for his steadfast collaboration on
Triton research for the last 9 years

Chakravarty Reddy Alla Chaitanya
(Research Intern)
...for helping invent adaptive sampler

Kristofor Mellroth (Head of Audio),
...for bridging Research and Games
Questions?

Slide deck with videos and papers available @www.nikunjr.com
WE’RE HIRING!

Email – andrewgl@Microsoft.com
Visit – http://thecoalitionstudio.com/#Join-Us