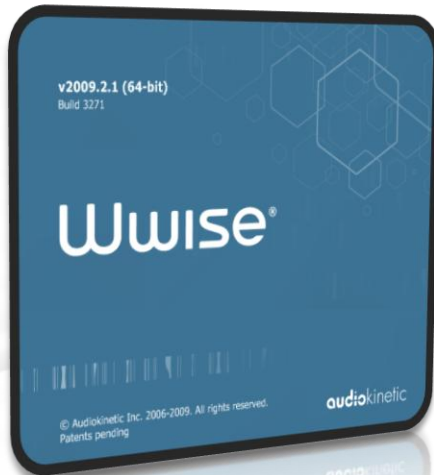


Tornado Outbreak + Wwise = Love



Scott Bilas
Wwise Tour 2009, Seattle

audiokinetic®



(Cell Phones?)

About This Talk

- Introductions (Hi!)
- About our game and its audio requirements
 - Why did we choose Wwise?
 - What makes Wwise different?
- How did we integrate Wwise into our game?
 - What did we screw up and have to redo?
 - A few quick “gotchas”

About Tornado Outbreak

- “Vacuum cleaner” game
 - Destroy, suck up, grow
 - Vaguely platformer-ish
 - Missions, power-ups, chain combos, mini-games, boss battles, split-screen coop
 - (Ok, fine, “it’s similar to Katamari”)
- Shipped September 2009
 - Wwise 2009.1 Patch 3
 - Small team, custom engine
 - Xbox 360, PS3, Wii
 - Wii caused the majority of our audio problems due to limited memory and CPU (imagine that!)



Demo

Wwise Tour 2009, Seattle

Audio Requirements

- The usual suspects...
 - Simple streamed looping music track with different levels based on "excitement level"
 - Tornado sounds, powerup effects
 - Some ambient and distant objects
 - Dialog for in-game cutscenes
 - Movies (rendered from Flash, played by Bink)
- Levels loaded all at once, no world-streaming
 - All game objects available up front
 - This simplified many parts of the game
- (Criminally) small audio memory budget on Wii

Audio Requirements: The Crazy Part

- Object-interactive and idles
 - Alert, panic, break, throw
 - Ambients
 - ...for hundreds of object types
- Scaling up of audio during level
 - Start out small and calm, but steadily grow
 - Very quickly approach a sonic wall of destruction
 - Must sound good at all sizes (of course)
 - Iconic sounds must still pop out of the sonic wall

(Most of this talk is about the crazy part)

Choosing Wwise

- “Uh oh, next milestone needs audio” ~2 yrs ago
- Evaluated the usual suspects: CRI, Miles, FMOD
 - CRI is the furthest along of those three
- Wwise was obvious choice, after price dropped 😊
- No porting from another audio engine involved
 - Audio was built from ground up with Wwise
 - This is a lot easier, of course...

Why Wwise?

- Wwise is an *audio management engine*
 - Audio guys have near total control
 - Less engineering involvement (engineers are slow)
 - Less engineering responsibility (I like this)
 - Heavy tool support ensures hitting CPU/mem budgets
- Others are simple “play samples + DSP” libraries
 - Low level, do-it-yourself
 - Channels, callbacks, fopen/fread()
 - Not game-oriented
 - Wwise is a total separation of “something happened” from “soundscape has changed”
- Awesome dev and support teams

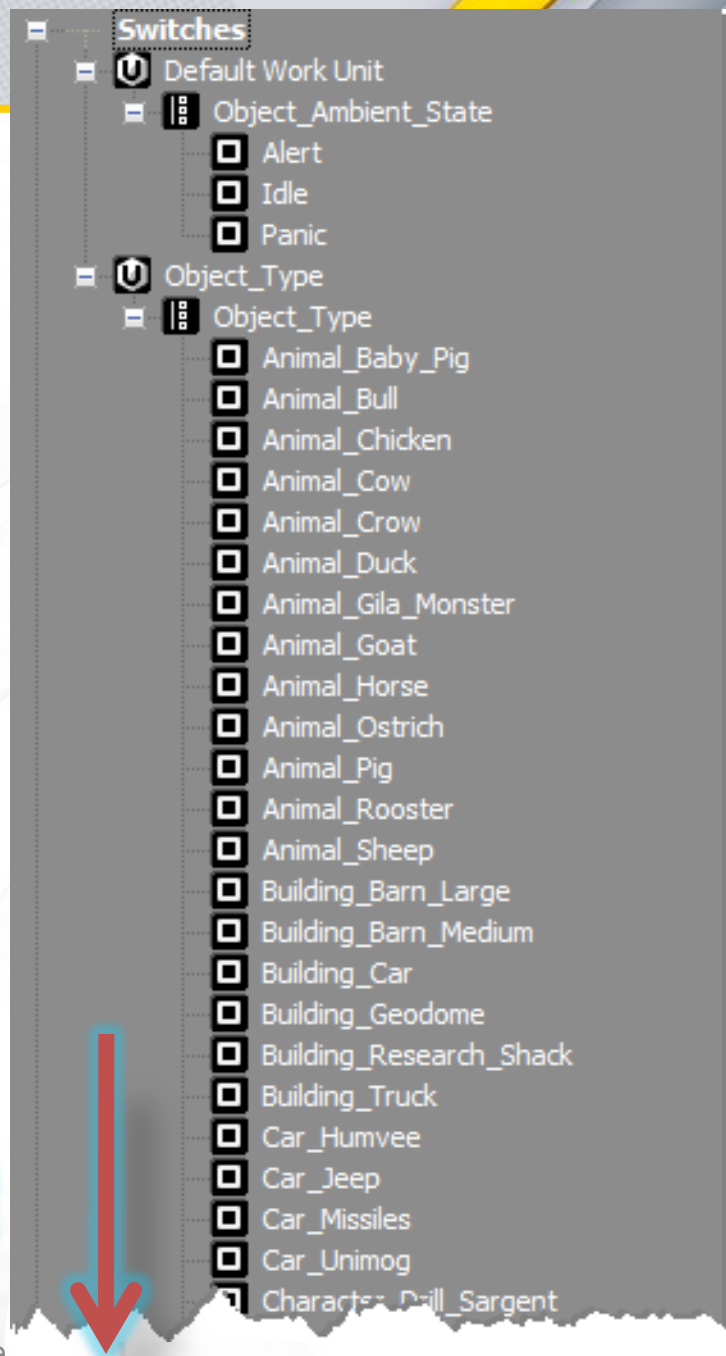
Wwise Is *Different*

- Focus is on audio creators, not engine integrators
 - No “set volume”, “stop” calls
 - Wwise actor-mixer objects can do almost *anything*
 - Game developers need to use ~5 API calls total
- I think of Wwise as “audio rigging”
 - Similar concept to character rigs in Maya
 - Requires significant up front prototyping and design
 - Significant learning curve, but worth it
- It took three tries for us to get it right
 - “Try” being a major reorg of rig or engine integration
 - Easy: GUI, ambient, tornado, music, etc.
 - Big trouble: the general event sounds

First Attempt

- Reduction of game object template types to “audio object type”
 - Example: grass1, grass2, grass_blue -> object_grass
- Initially implemented as switch in Wwise
 - *This got big very fast, into the hundreds of types*

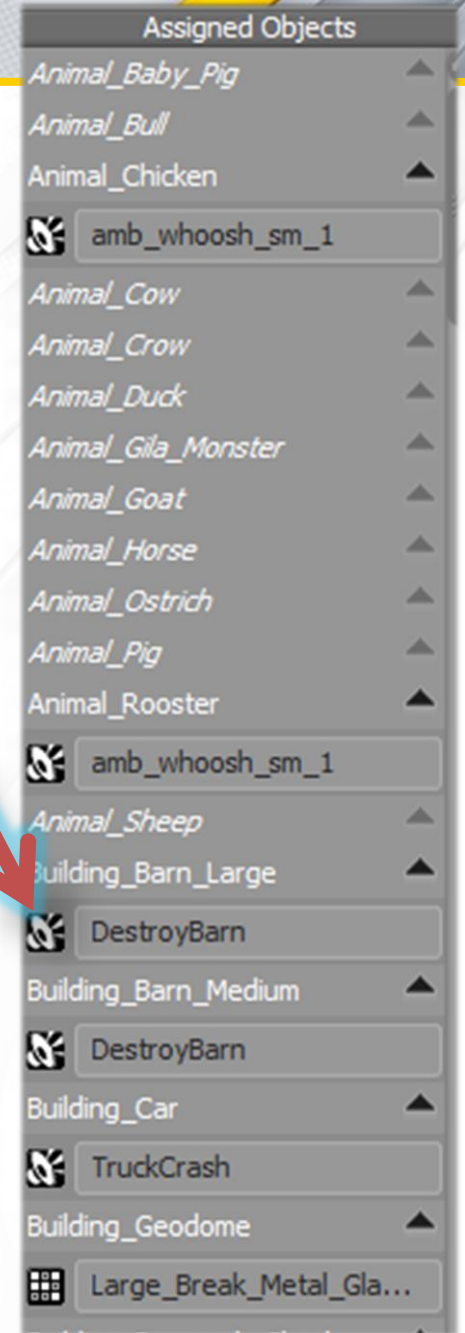
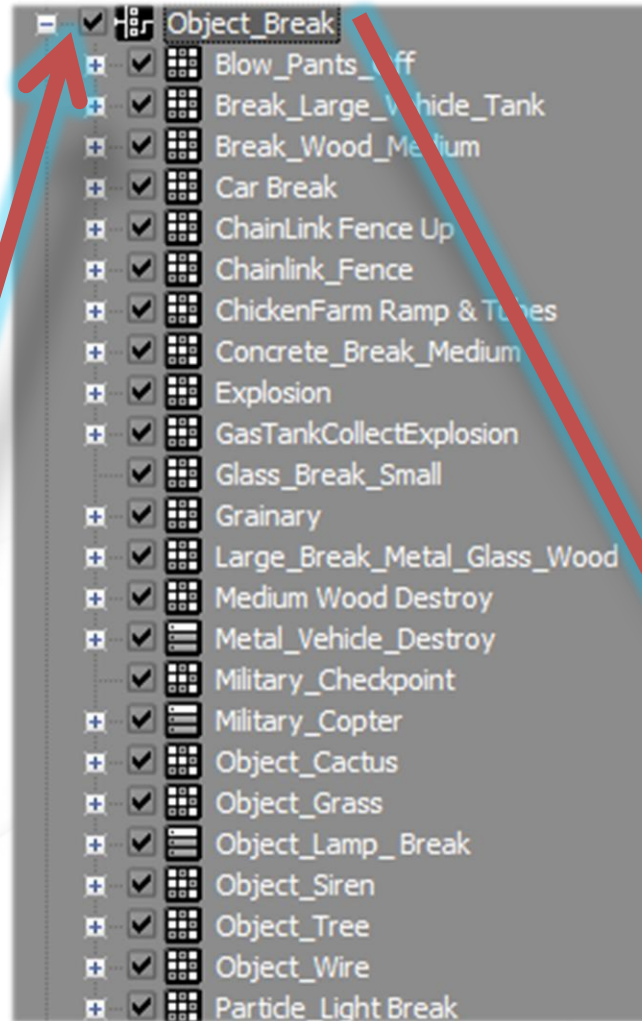
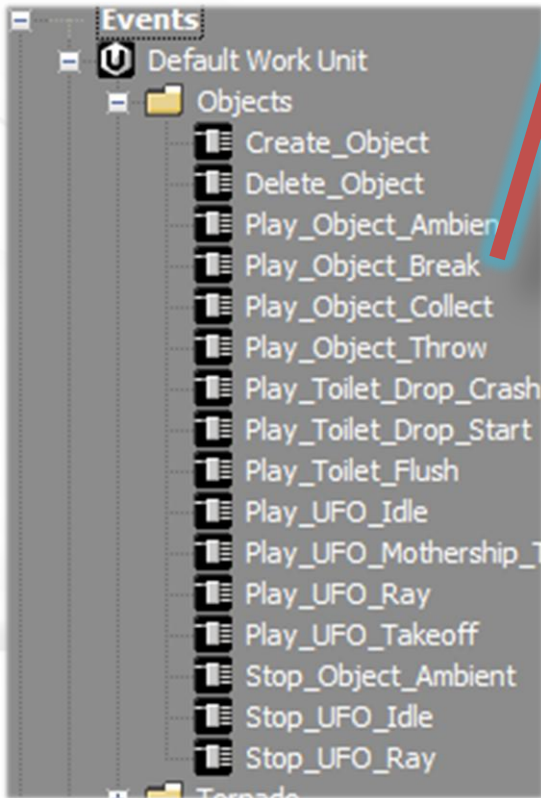
OMG!!



Implementation

- New property: “audio object type”
 - Per game object template or instance
 - Configured in Maya and other tools
 - Inheritance, overrides, the usual...
- Audio-typed game object lifetime
 - Registered with Wwise on load until destruction
 - Audio object type sets “the big switch” on load
 - Runtime state for idle/panic/alert/shaking/etc.
- General events that were routed on through
 - Play_Break, Play_Collect, Loop_Idle...
- This is probably what most games do, right?

Switch of Switches of Switches



Single Giant Tree: Bad

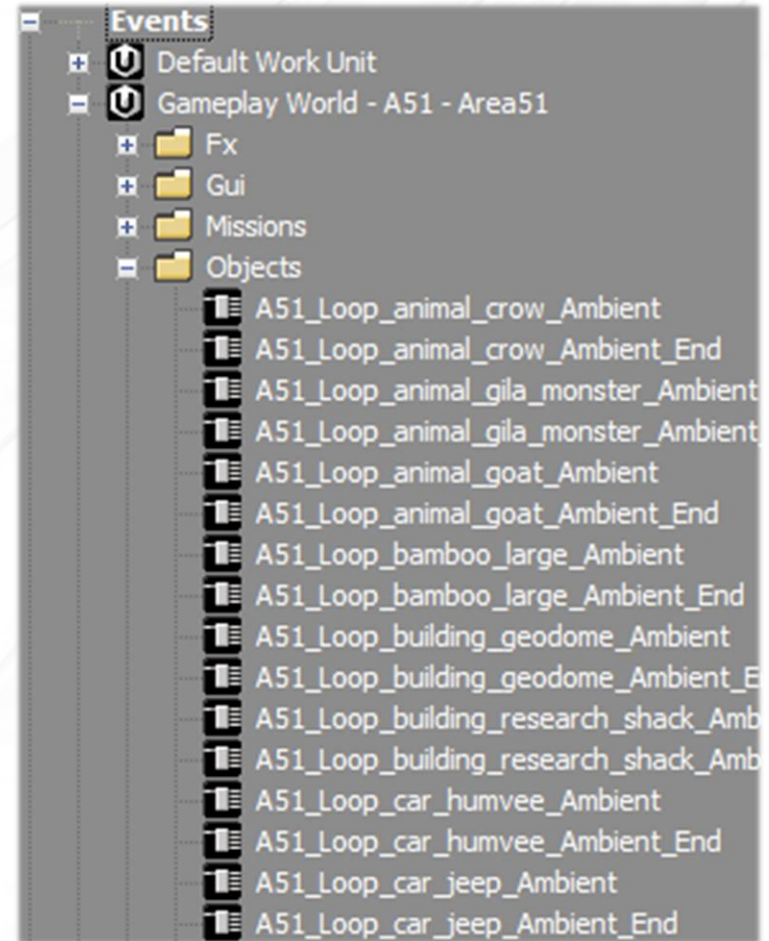
- Difficult to scale up
 - Audio object type switch rapidly exploded
 - UI issues from Wwise 2007/8 made it painful
 - Hard to add people working simultaneously
 - Ultimately always hit that “one tree” problem
 - Can slow prototyping significantly
- More work involved in managing assignments
 - Wiring up a new type needed careful tree nav
 - Hard to separate concerns of sound construction from game object behavioral structure
 - Event actions had to be least common denominator
 - Harder to share across a tree (no instancing)

Switch of Switches of Switches: Bad

- Diagnostics more difficult
 - Object setup caused a lot of noise
 - More work to find what the state of something is
- The nail in the coffin was media-bank assignment
 - Each of the 10 or so events splayed out to nearly all ways in the entire project
 - Made it impossible to split bank files per-world
 - “Uh oh, next milestone must run on green Wii”
- Note: 2009 features may mitigate some of these
 - UI improvements (especially windowing overhaul)
 - “Prepared” media concepts

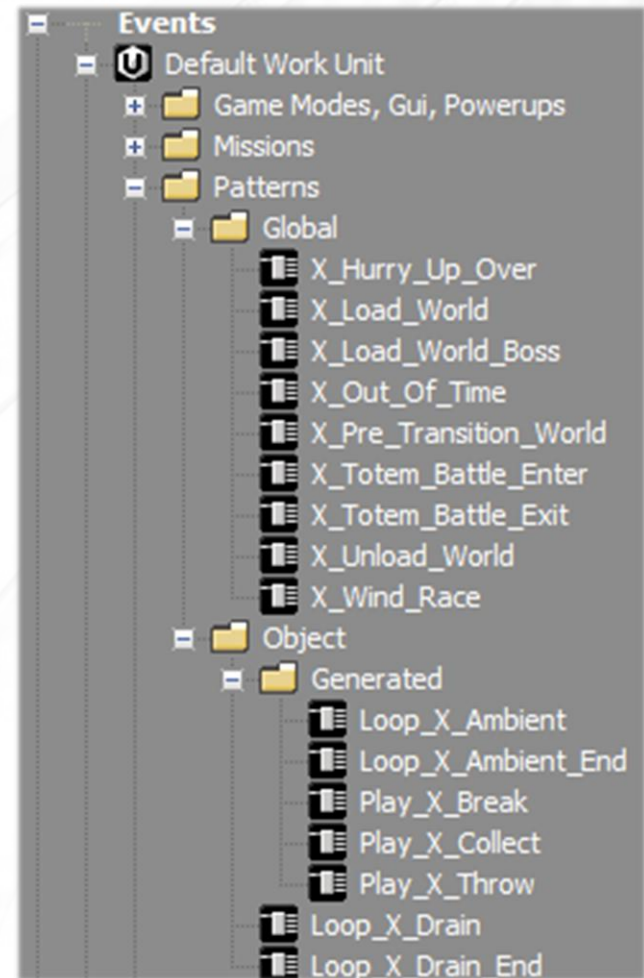
Second Attempt: A Different Approach

- Implement per-world banks
 - Split off universal sounds into common.bnk
 - Three letter “world prefix” for anything with a name
- Turn that tree inside-out!
 - Only ~10 object events, so put per-world events in flat list for all type/event combos
- Some assembly required
 - Generic events
 - Optional events
 - Tool-generated events



Generic Events

- Translation layer for patterns
 - `g_audiomgr.PostEvent(AK::EVENTS::X_LOAD_WORLD)`
 - “X” converted to world prefix (CHK for the chicken farm)
 - Routed to Wwise as string
 - Example: `CHK_Load_World`
- Per-world banks now possible!
 - World-specific events can be posted by world-agnostic code
 - One event tree in each bank
- Maintains type safety
 - Generic events are in `wwise_ids.h!`
 - (Intellisense loves this)

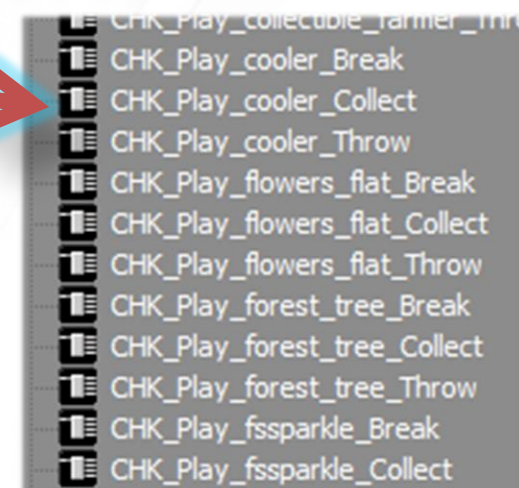
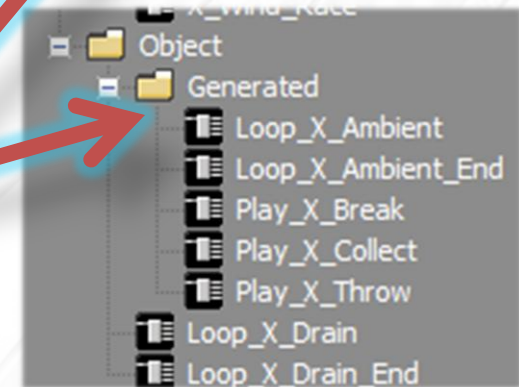
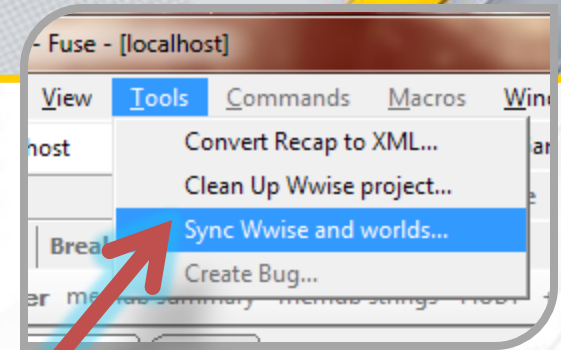


Optional Events

- Many events will not be wired up
 - On purpose or not (does not matter)
 - Will change over time as audio tree fills out
- Generics will fire a lot of false positive errors
 - This makes team not trust error system
 - Masks true problems with required audio sounds
- Solution: hack in `AK_EventIsOptional` for `PostEvent`
 - Small modification to add a new flag to Wwise source
 - Tells system to not report an error on failed lookup
 - `g_audiomgr.PostOptionalEvent(AK::EVENTS::PLAY_X_BREAK)`
 - Wwise is very easy to modify

Generated Events

- Problem: that's a lot of events!
 - Lots of typing and error potential
 - Solution: generate them
- New command in our "Fuse" tool
 1. Use patterns found in .wwu under Object/Generated
 2. Use a spec file of audio object types detected from level load
 3. Write empty events directly into .wwu (it's just XML...)
- Filling in the events
 - Audio designer fills in with easy drag and drop
 - Instance from the audio tree



It Worked!

- Solved most of our issues
 - Event and actor-mixer trees independent
 - Per-world banks no problem
 - Wiring new types simple, consistent
 - Full event actions available, instancing possible
 - Diagnostics straightforward and obvious
- Drawbacks
 - Duplication of media across worlds
 - Doing a build causes lots of warnings we ignore
 - Takes up more DVD space but not a problem for us
 - Combinatorial explosion possible with more generics
 - Wasn't a problem on this game

Our Event Conventions

- Events *never* named after sounds
 - Bad: “play_clock_tick” or “play_cat_screech”
 - Good: “loop_clock_ambient” or “play_cat_kicked”
 - Sounds and rigging inevitably change
 - Events usually only get added and removed
 - Also reminds gameplay engineers not to direct audio
- No reuse of events, always make a new event
 - Super important when expanding audio content
 - Audio designer can do it all without code changes
 - Hard to resist when prototyping, so provide proto events
- World prefix
 - CHK_Loop_animal_cow_Ambient, CHK_Wind_Race...
 - Helps to prevent use of foreign world event

Final implementation

- Optimizations
 - 9k objects registered at once on Wii bad (memory)
 - Tornado damage radius for registration bad (CPU)
 - Solution
 - Game object only registered on its first posted event
 - Also helped with attaching to Wii locking it up
 - Also: special tag for objects containing ambients
- Final tally
 - 750 object types across 15 worlds (410 unique)
 - 4600-line `wwise_ids.h`
 - Though 99% not used directly by code due to generic events!

Final Thoughts

- Gotchas
 - Compiling PS3 SPU jobs yourself? Careful!
 - Match build options precisely to AK's
 - Always use release versions of jobs, even in debug
 - Make sure to schedule time to tune memory pools
- Email Support!
 - Get real developers, fast answers, bug fixes..
 - Also can get good advice, best practices
- Bottom line: we love Wwise!

Thank You!

- Slides will be posted to:

<http://scottbilas.com>

- Contact me at:

Scott Bilas

scott@bilas.us

