Sound Propagation in 3D Environments

Chris Carollo
Ion Storm Austin
ccarollo@ionstorm.com

What Sound Propagation is
(at least in this talk)

- A method for modeling the movement of sound
- Relevant spaces:
  - Densely occluded indoor or faked outdoor
  - Multi-material
  - Basically, typical FP action game settings
- Sounds that carrying meaning

What we’ll go over

- Why sound is important
- What our requirements and goals are
- What academics are working on
- A practical implementation strategy
- What its performance cost is
- What its implementation cost is
- What you should watch out for
- Some neat tricks it enables

Why is sound important?

- We use it to gather our information about our surroundings
- One of two senses game can substantially appeal to
- More enveloping sense than vision, which is tightly focused
- Typically used to direct vision

Goals: Immersion

- Spatial cues that don’t disorient the player
  - Distance
  - Directionality
  - Updated as sound/listener move
  - Continuity is key!
- A world that feels solid
  - Player can’t hear through walls
  - If they can, sounds are filtered appropriately

Goals: Aesthetics

- Ability to impart a mood
  - Heavy filtering for cramped or warm spaces
  - Echo/reverb for open or cold spaces
  - Other filters based on environmental materials

Implementational Benefits

- Better suited to attach meaning to sound
  - Sound “packets” that fly through the world
  - Players and AIs can use sounds to make gameplay decisions
  - Similar to vision
- Single dispatcher for sound events
  - Same mechanism used to handle all sounds for all listeners
  - Easier to hook game events
  - Listeners (AIs, alarms, player, etc) hear by similar method to that of the player

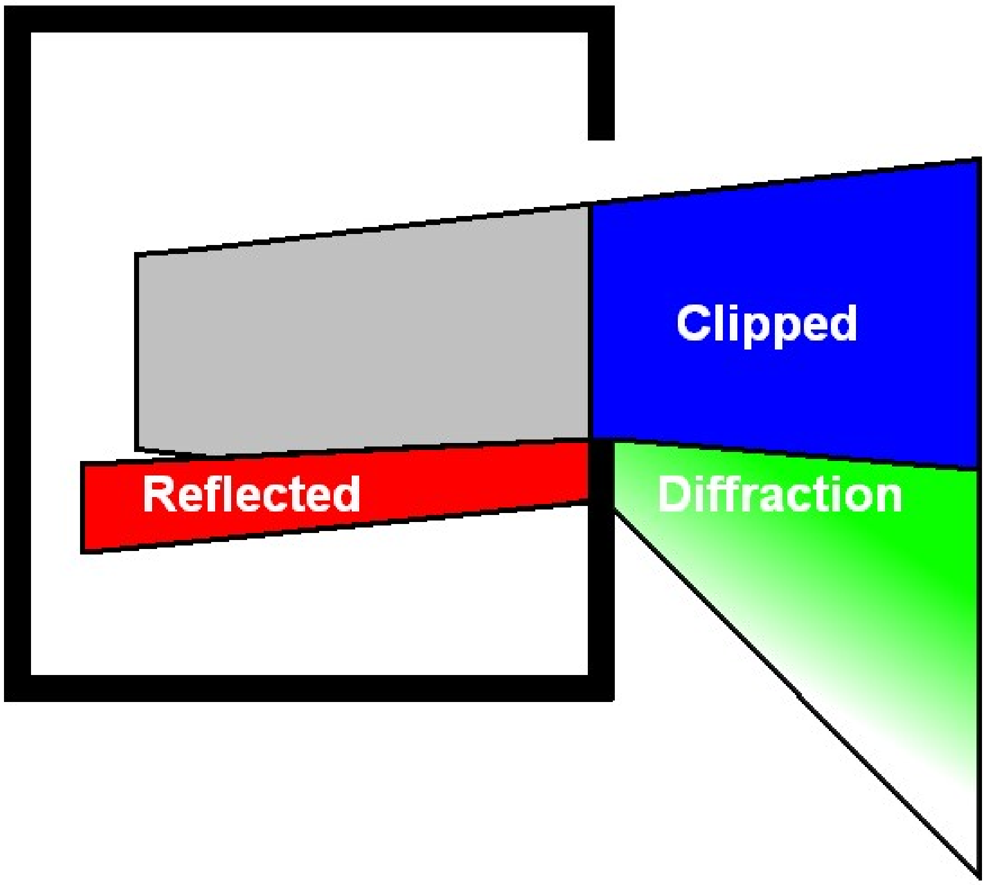
No Sound Propagation means…

- Generally poor sound cues that confuse players
  - Players will start ignoring sound cues after a while
- Raycasting for occlusion
  - Doesn’t give a real direction when occluded
  - Lack of continuity
  - Can actually be more expensive
- AI
  - Hearing through different mechanism can lead to broken player expectations
  - AI cascades because of hearing through walls

High-end solution: Beam Tracing

- Lots of good research has been done by Funkhouser et al
- Casts volumetric beams from sides of cube
- Hitting surfaces creates more beams:
  - Reflection
  - Clipped
  - Diffraction

High-end solution: Beam Tracing



High-end solution: Beam Tracing

- Once computed, beams are cached
- Moving listeners are cheap: just moving through beams
- Moving sounds are expensive: must recalculate entire beam tree for that sound
- **Lots** of beams: 2K poly world with 8 reflections is ~300K beams! (Funkhouser)
- At 100B per beam, that's 30MB per sound!
- Not really viable with today's hardware limitations

On to a practical implementation…

- We’ve got a bunch of goals
- We’ve got a bunch of spaces
- What do we need to make it happen?

Implementation Requirements

- Information about how world spaces are connected
- Similar to visibility for rendering, but for sound
  - Interested more in how areas are connected than whether they are potentially connected
  - Can be more coarse
- Basically, how does a sound generated in the library get to the kitchen?

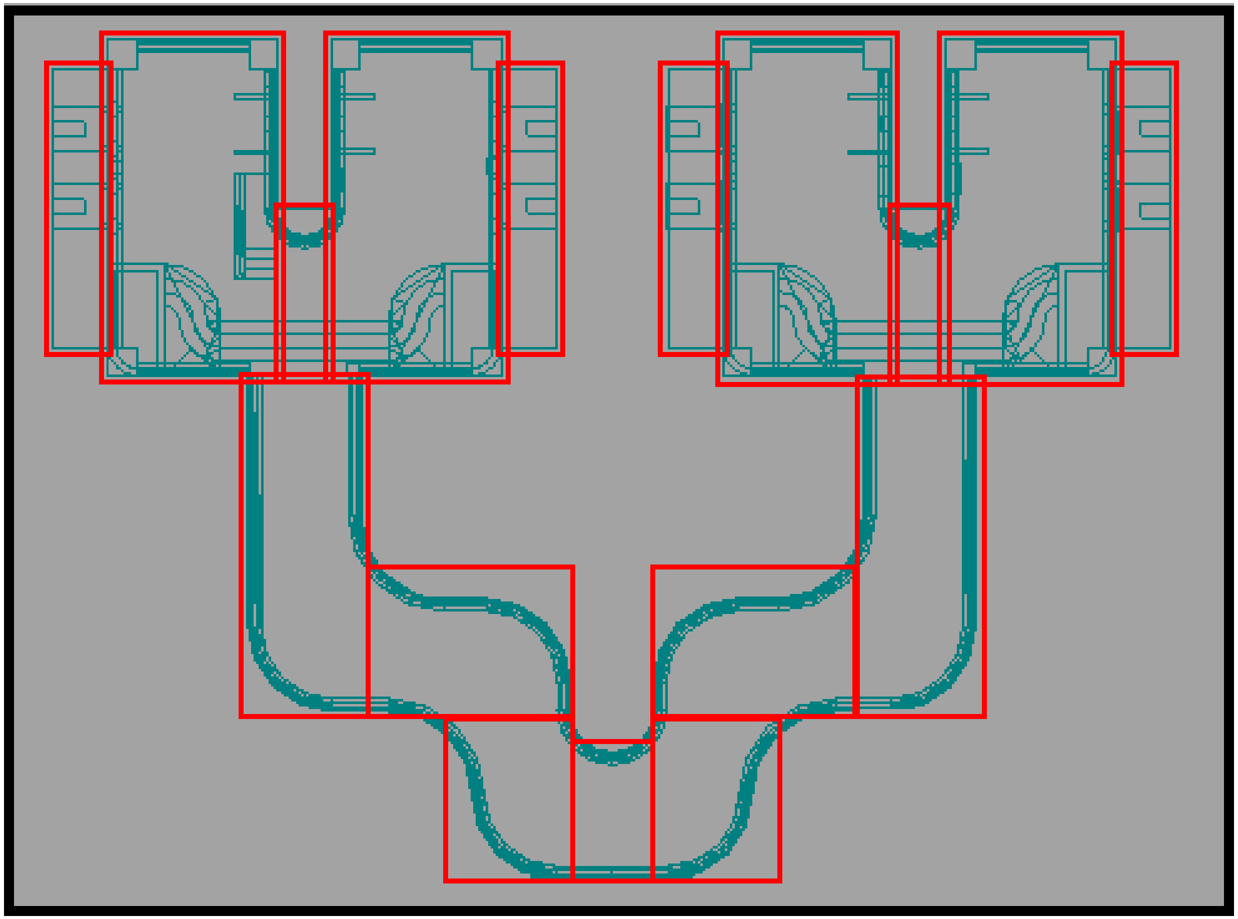
A Connectivity Database!

- World is divided into connected cells
- Cells are just convex polyhedra
- Connected through portals
- Portals work best if they're polygonal
- Needs to track listener objects in cells
- Need not be used solely for propagation
  - Top-level visibility culling for rendering
  - Pathfinding for AI
  - Object buckets for physics

Designer-Placed Volumes

- Lower-poly representation of connectivity
- Overlaid on real geometry
- Problems with manually placed volumes are because of human error and bad tools

Designer-Placed Volumes



Designer-Placed Volumes: Pros

- Very fast because of low poly count
- Can be used with any world geometry
  - CSG
  - Organic spaces
  - Polygon soup
- Used in shipping games
  - Dark Engine: Thief 1 & 2 and System Shock 2

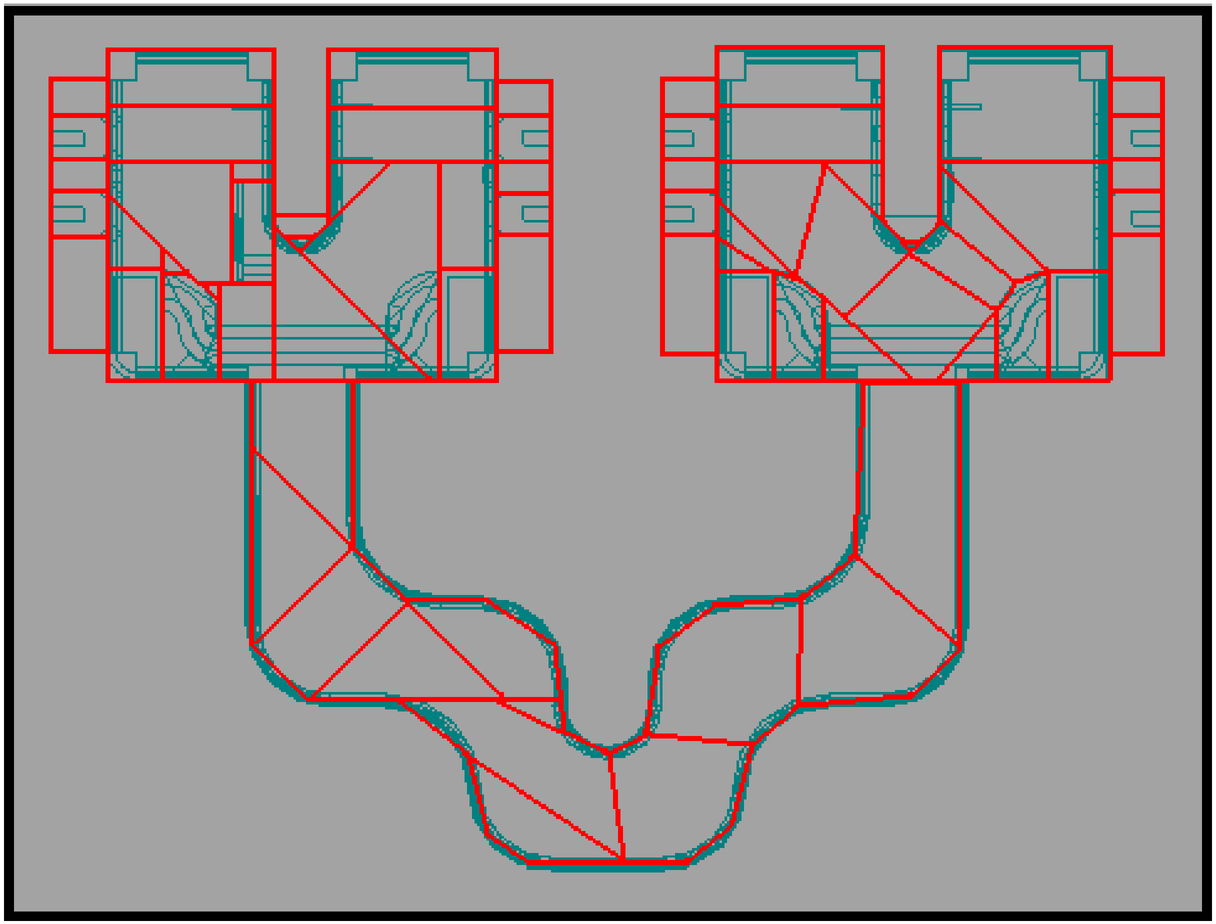
Designer-Placed Volumes: Cons

- Loose representation can cause inconsistencies
- Must be kept in sync with real geometry, which means **lots** of designer work
- Specific problems:
  - Leaks: world not connected but connectivity is
  - Blocks: world connected but connectivity not
- Need to spend the time to provide good tools to designers

Automatically Generated Volumes

- More accurate representation of connectivity
- Requires geometry amenable to automation
  - Generally solid spaces
  - Some sort of ordered connectivity
- Problems with automatically generated volumes are because of code bugs and numerical instability

Automatically Generated Volumes



Automatically Generated Volumes

- Pros:
  - Always in sync with real world geometry
  - Virtually no designer work
- Cons:
  - Potentially a lot of polygons
  - Poly reduction can be complex

Minimal Path Tracing Overview

- Far less computationally expensive
- Also less difficult to implement
- Consists of four steps:
  - Coarse floodfill
  - Path trace
  - Postprocess
  - Repropagation
- Example of path generation algorithm

Minimal Path Tracing: Implementation

1. Breadth-first floodfill of the cells, through the portals
   - Each cell knows what listeners are in it
   - Can precompute cost between portals
   - Intra-cell traversal costs determined by:
     - Distance
     - Size of portal
     - Relative facing of portals

Minimal Path Tracing: Implementation

1. Handle each listener object encountered
   - Generate minimal path back to sound source
   - Or not, depending on what kind of listener
   - Gather sonic properties of cells
   - Cache minimal path on object
   - Add object to “I heard something” list

Minimal Path Tracing: Implementation

1. End-of-frame processing step
   - Gather all paths from an object to a given sound
   - Deal with multiple paths
   - Generate filter/reverb based on paths and materials encountered
   - End up with distance, direction, and effect

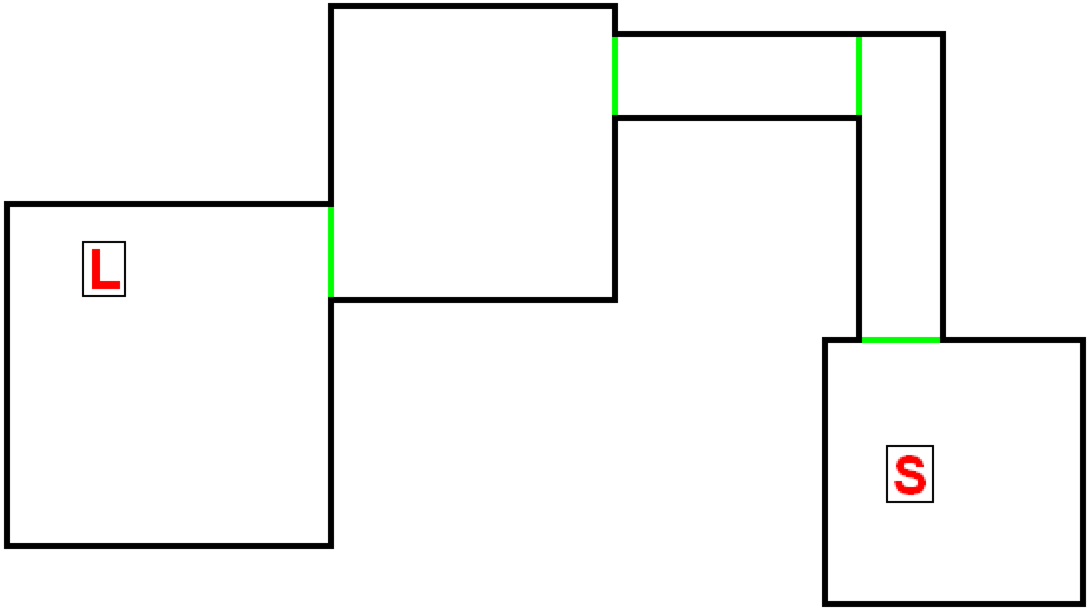
Minimal Path Tracing: Implementation

1. Note how long until repropagation
   - Rate of movement of the sound and listener
   - Dot product of velocity with direction vector
   - Assumes constant velocity between repropagations

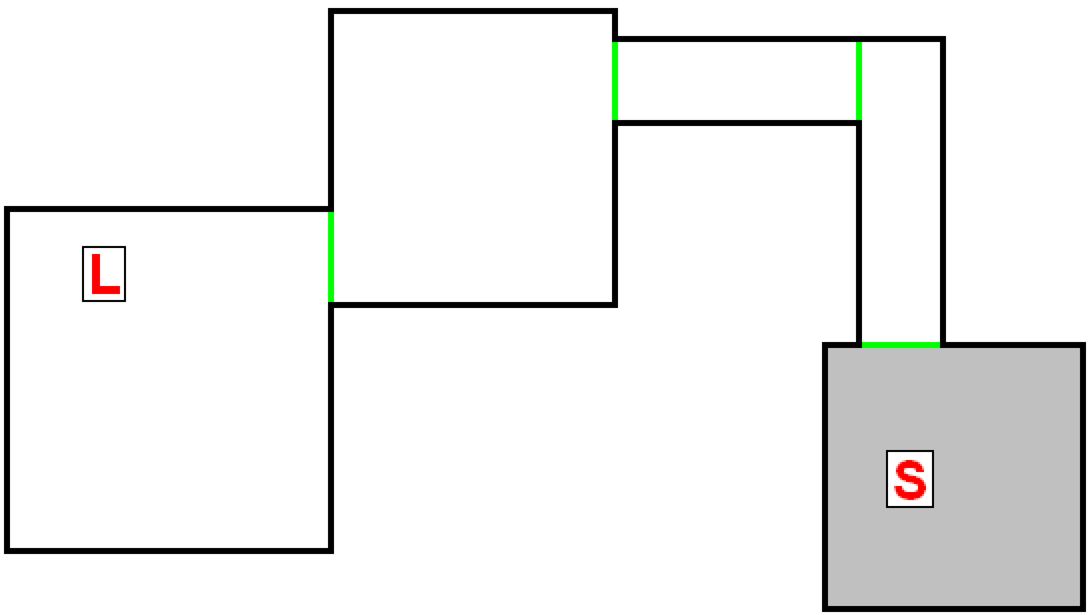
Minimal Path Tracing: Notes

- Minimal path generation
  - Analogous to stretching a string through portals
  - Only deals with portals
  - Solving from both ends, is analytically hard
  - Iterative solution to within a correctness epsilon
- Dealing with Head Relative Transfer Function (HRTF) hardware
  - Use “virtual speaker objects” for the hardware voice positions
  - Extrapolate positions from direction and distance

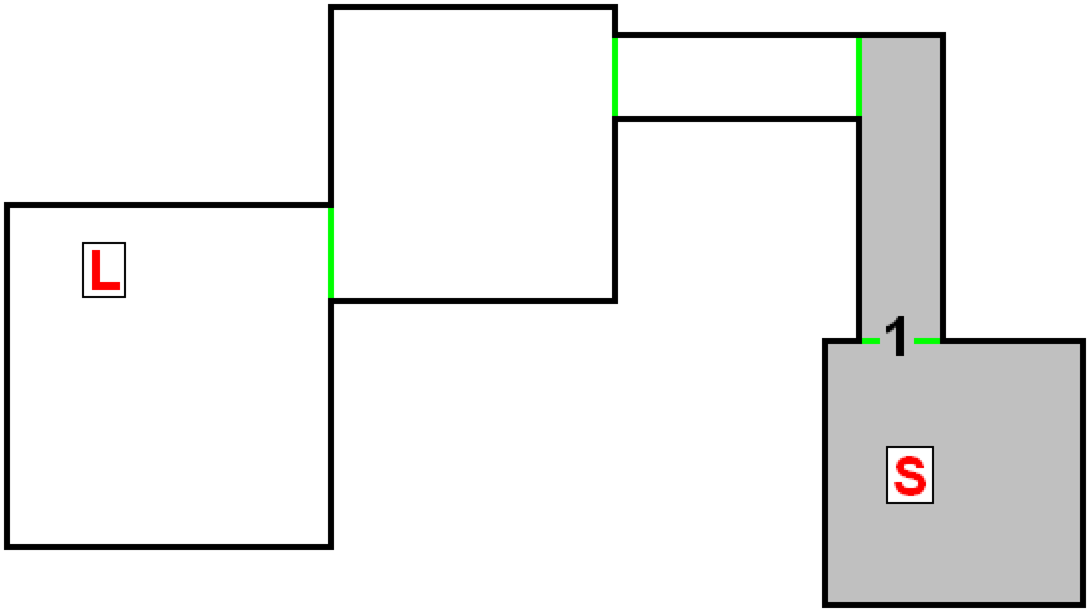
Minimal Path Tracing: Example



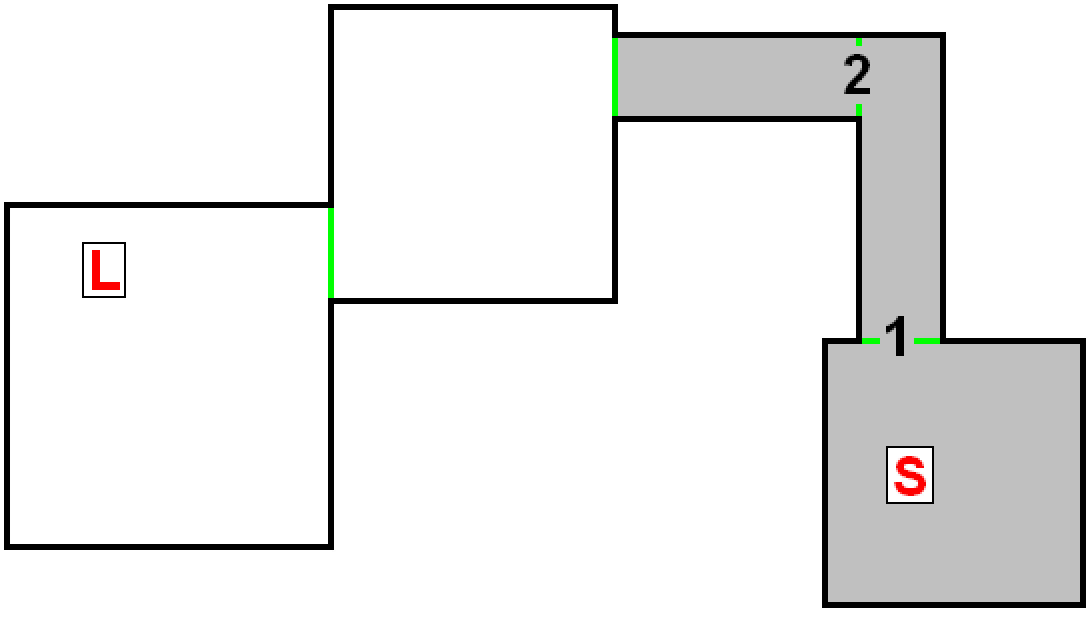
Minimal Path Tracing: Floodfill



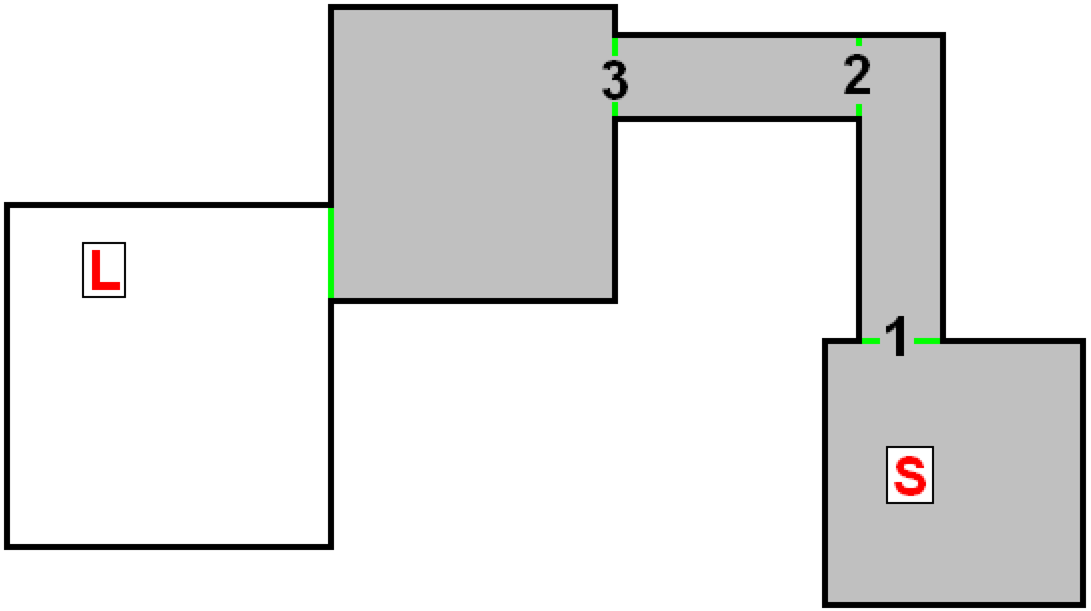
Minimal Path Tracing: Floodfill



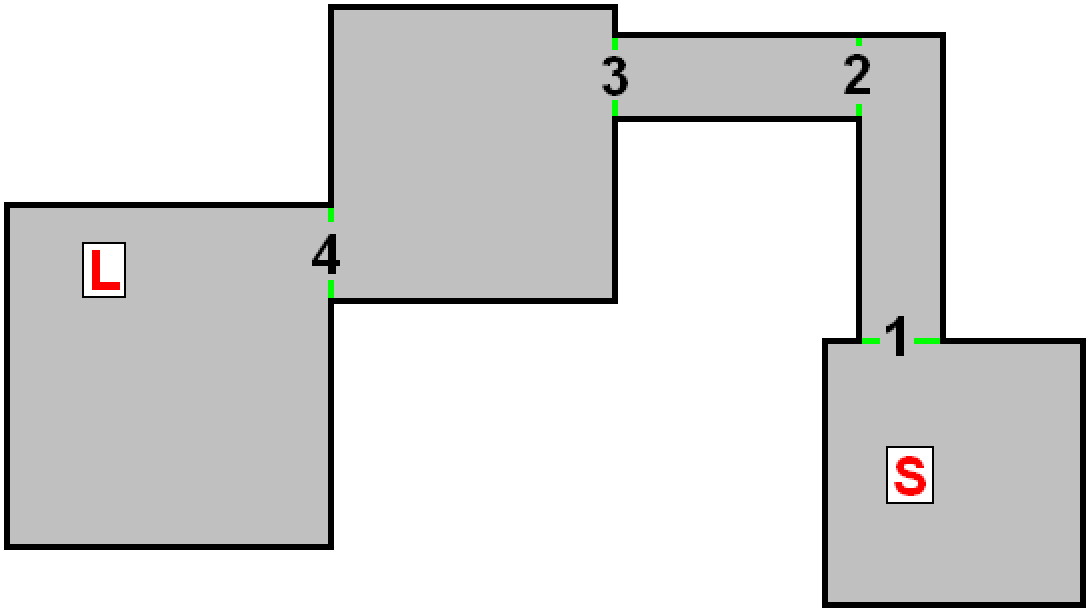
Minimal Path Tracing: Floodfill



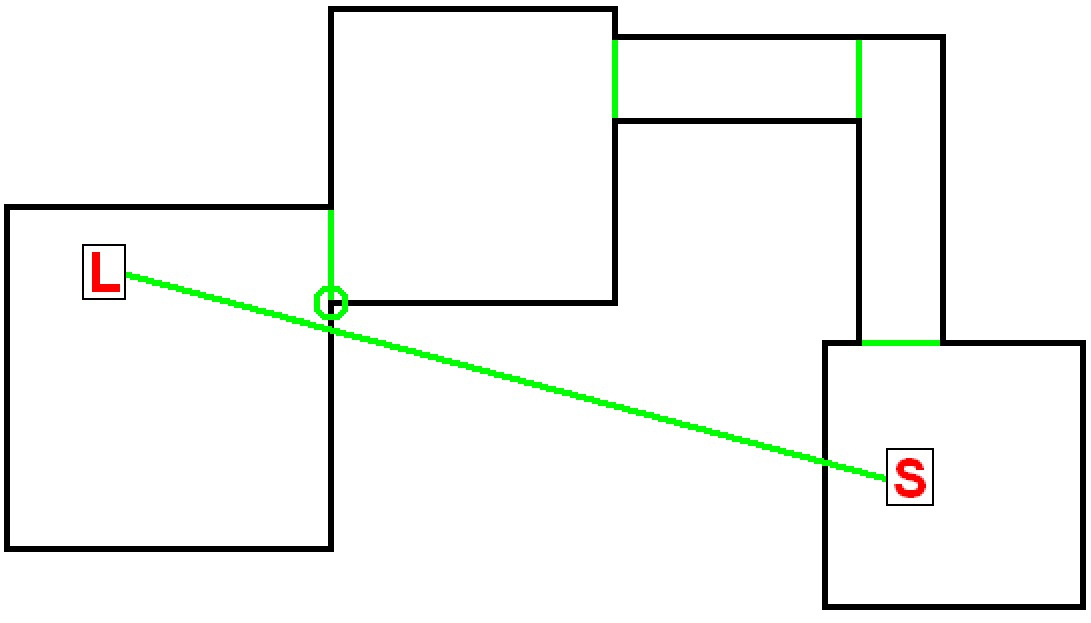
Minimal Path Tracing: Floodfill



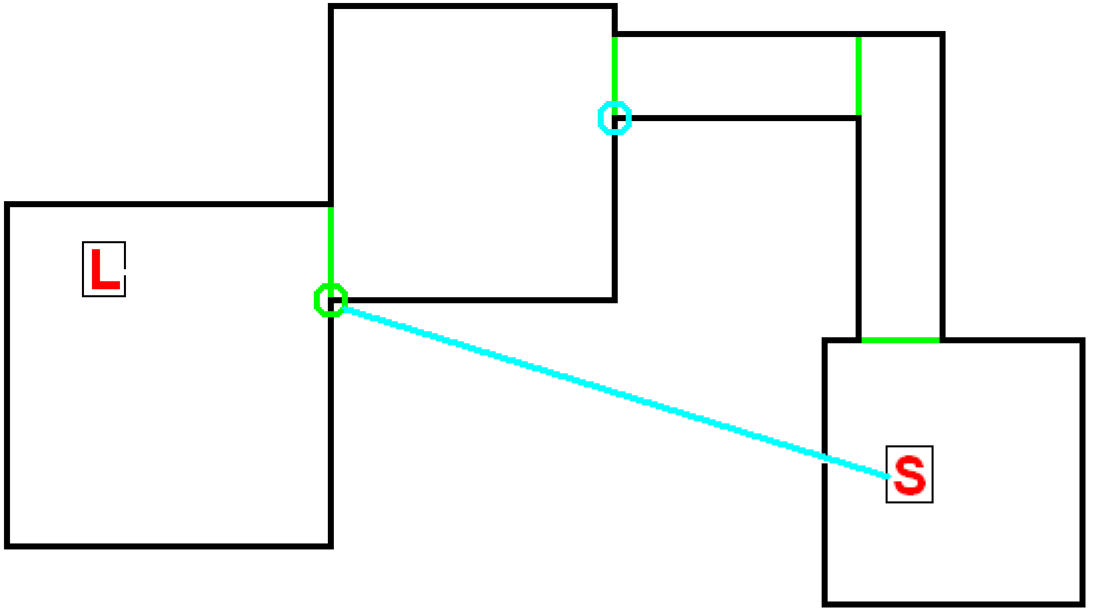
Minimal Path Tracing: Floodfill



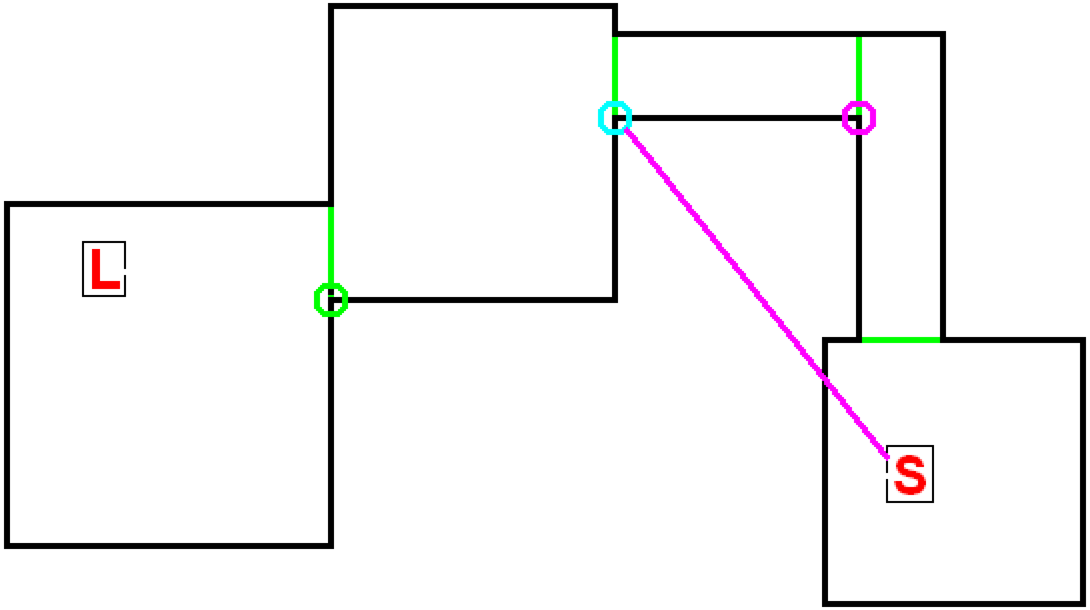
Minimal Path Tracing: Iterations



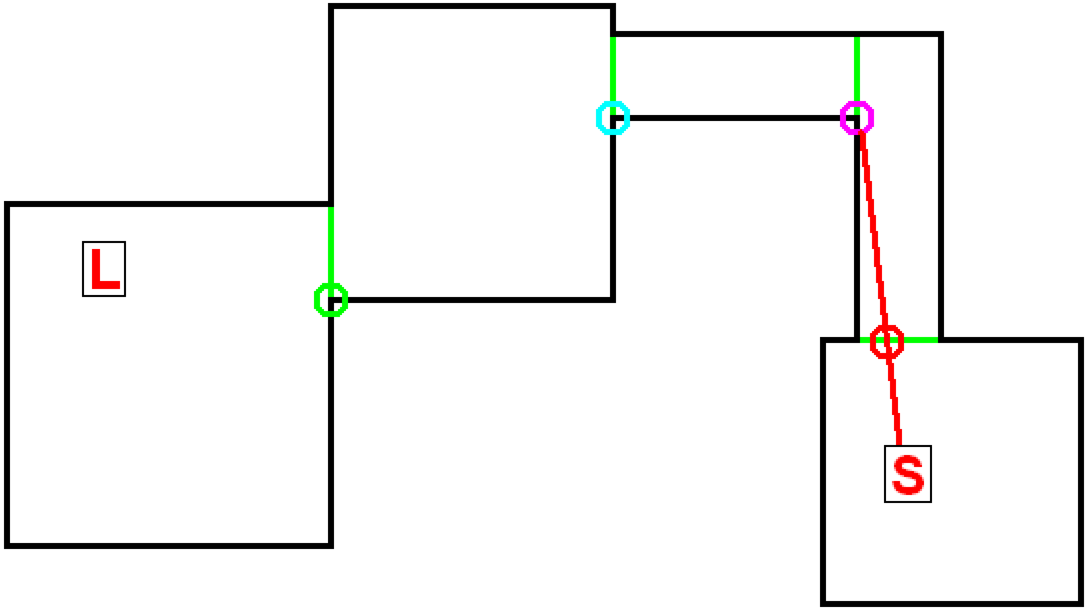
Minimal Path Tracing: Iterations



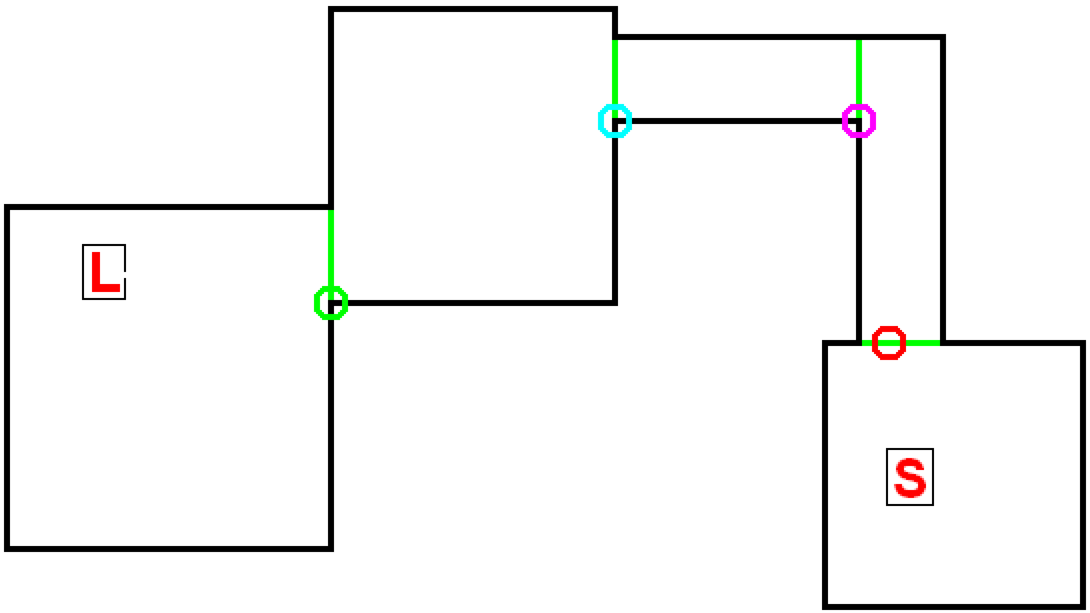
Minimal Path Tracing: Iterations



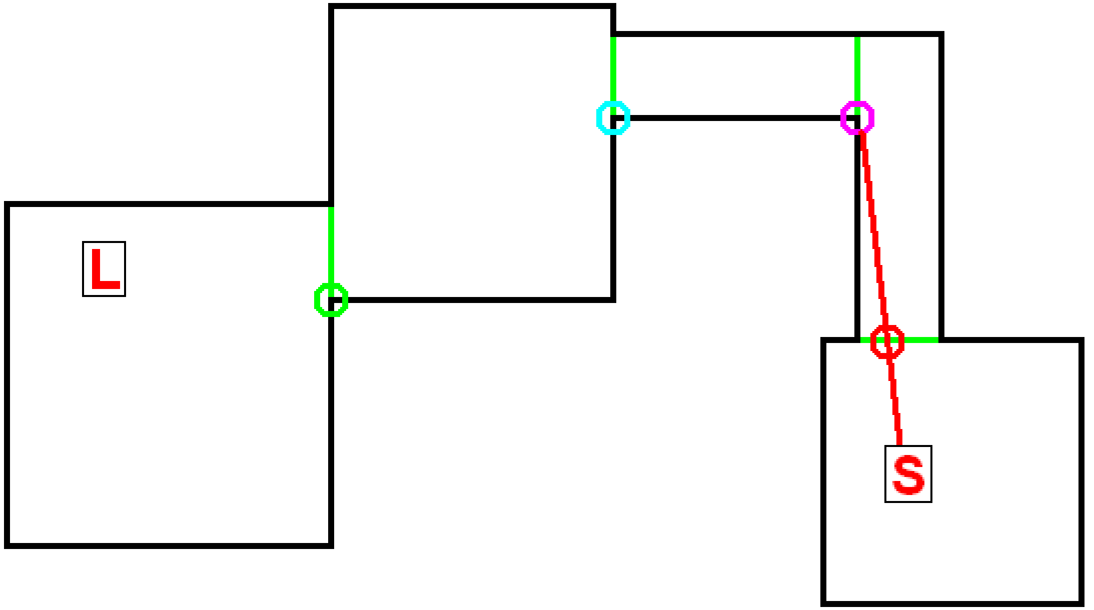
Minimal Path Tracing: Iterations



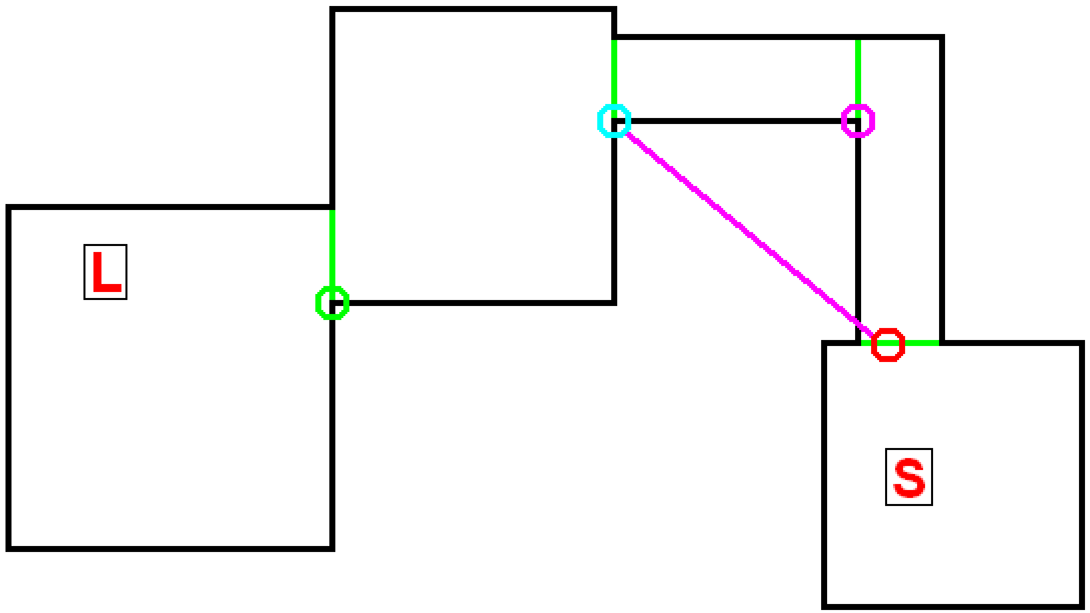
Minimal Path Tracing: Iterations



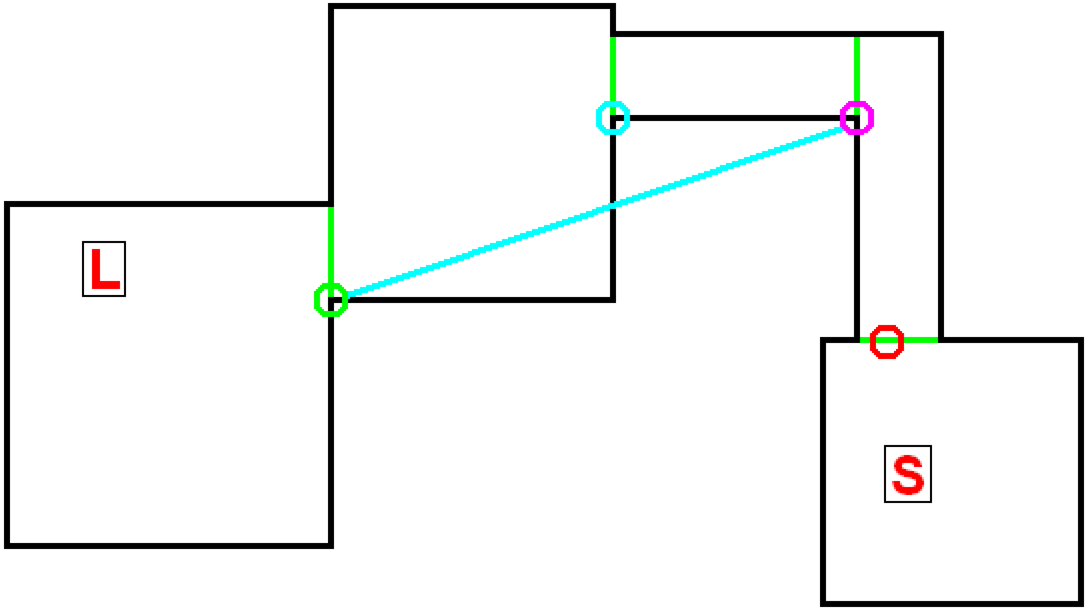
Minimal Path Tracing: Iterations



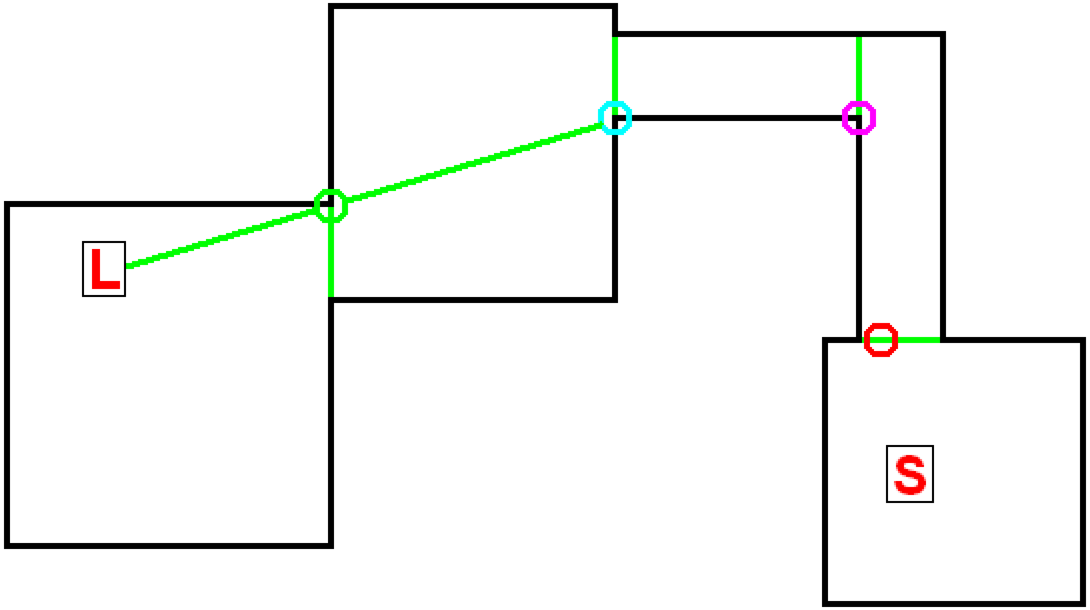
Minimal Path Tracing: Iterations



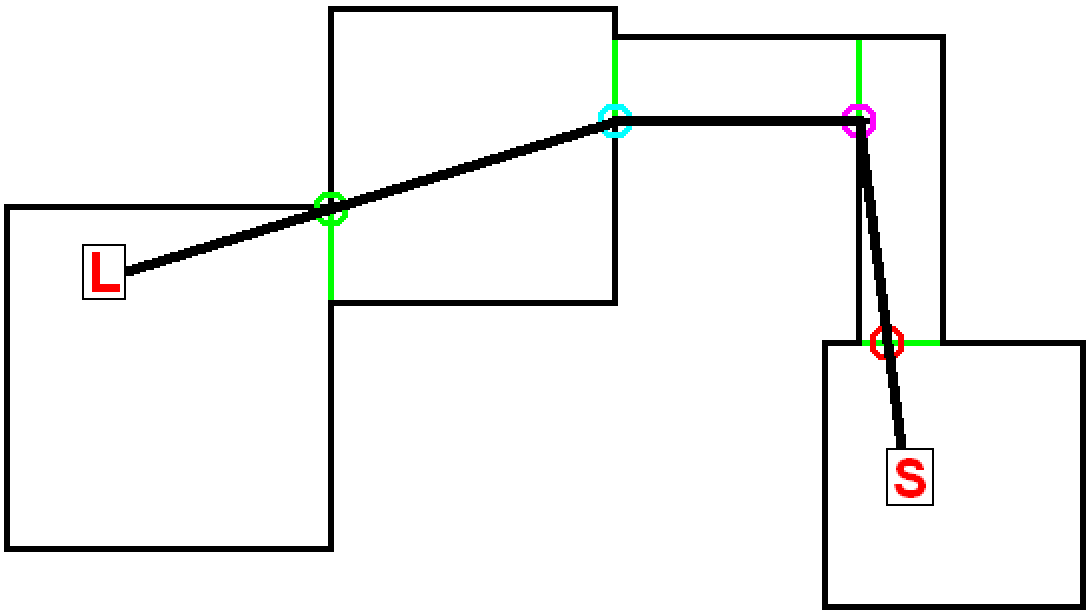
Minimal Path Tracing: Iterations



Minimal Path Tracing: Iterations



Minimal Path Tracing: Iterations



Minimal Path Tracing

- Real-time demo of minimal path example

Minimal Path Tracing - Multipath

- Allow re-entry during initial floodfill, to within a tolerance
- Can solve multiple higher-fidelity paths if the listener object warrants it

Minimal Path Tracing - Multipath

- Relatively few, or a variable number of channels?
  - Average direction and distance for virtual sound location
  - Don’t break continuity!
- Lots of channels?
  - Play all the sounds!
  - Make sure they’re synced – don’t want to echo!

Minimal Path Tracing - Occluders

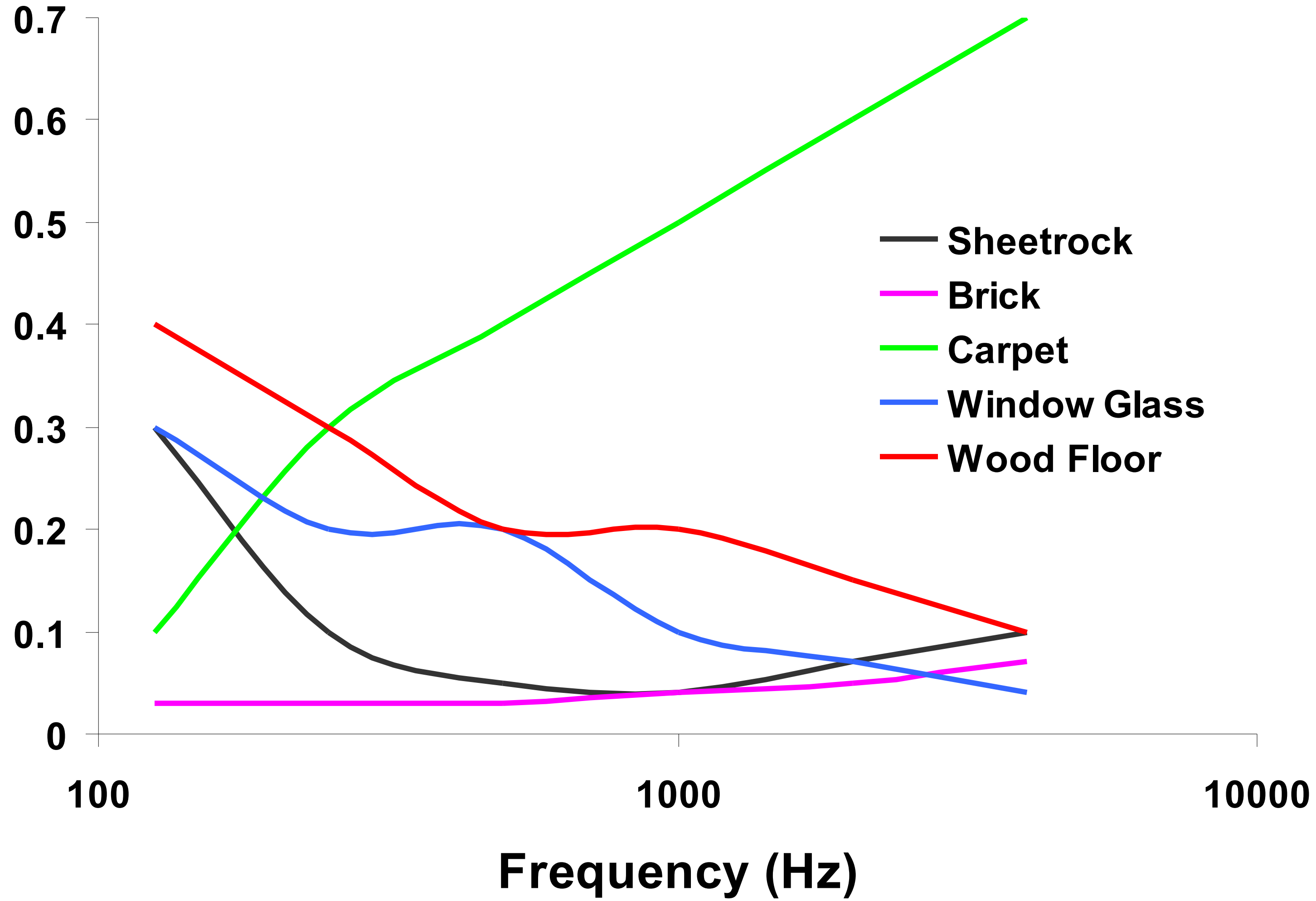
- Occlusion at portal boundaries
  - Doors, windows, etc
  - Partial occlusion just increases effective distance from source
  - Could also be dealt with by accumulating filters
- Mobile occluders
  - Probably only care about straight-line mobile occluders
  - Also might track accumulation of objects at portal boundaries

Filtering and Effects

- Filtering strength
  - Based on how materials absorb different frequencies of sound
  - Reflectance: Surface area of material
  - Transmission: Thickness of material
- Filtering and effects
  - Build as a DSP effect
  - Parametric equalization

Absorption by Material Type

(Hall, Muscal Acoustics 2nd Ed)



Caveats and Limitations

- Lots of data caching of portals and paths
  - Static worlds work best
  - Dynamic worlds can work, but require good cache invalidation and updating
- Assumes inaudibility distance
  - Sound stops propagating at some threshold
  - Must be large enough to handle best listener in world

Neat tricks

- Speed of sound delays
  - True distance is known so sound can be appropriately delayed
  - Seen as a bug?
  - Game design relevance?
  - Perhaps too pointy-headed

More neat tricks

- Better approximation of dynamic range
  - Speakers generally have poor dynamic range
  - Slow rate of attenuation makes sounds louder
  - Won’t be hearing loud sounds across whole level because distances are real!

Even more neat tricks

- Listening at doors
  - Eliminate the sound occlusion when close enough
- Laser microphone
  - Place virtual listener at partially occluding window and disable its occlusion effects
- Listening quietly
  - Boost volume of important effects to certain listeners (the player) when stationary

Yet more neat tricks

- Sound masking
  - Meaning bearing sound can be masked by louder sound
  - Player can use tools or environment to generate "sound cover"
- Can provide gameplay direction in open-ended environment
  - What areas are interesting?
  - Example: Horn of Quintus in Thief

Conclusions

- Minimal path sound propagation is fairly straightforward
- Connectivity database can be challenging, but can be useful in other areas as well
- Good sound propagation applies to many elements of gameplay
- Subtle parts of a game experience like audio contribute more than you might typically think

Questions?