ROOM REVERB PLUGIN: REQUIREMENTS DOCUMENT

1. OVERVIEW & GOALS

1.1 PURPOSE

RoomReverbPlugin is a JUCE-based audio plugin and standalone tool that generates physically-inspired room impulse responses (IRs) using a two-pass, ray/path-tracing model of a rectangular room. The primary goals are to:

- Enable users to design simple room geometries and material properties and preview the acoustics in real time.
- Export high-quality, safe (non-clipping) IRs as 32-bit float WAV for use in convolution reverbs.
- Provide a clear visual first-person view to aid understanding of the acoustic model.
- Serve as a foundation for research and future features such as 3D/binaural rendering via an external engine or SAF.

The plugin targets VST3 and a JUCE standalone build for testing without a DAW. It prioritises deterministic results (seeded randomness), responsiveness (non-blocking processing), and transparent UX.

1.2 STAKEHOLDERS

The following stakeholders influence the scope, priorities, and acceptance of the project:

Role	**Primary Interests**	**Example Responsibilities / Decisions**
Project Owner / Lead	Overall vision; roadmap; licensing strategy (GPL/commercial).	Define milestones; approve scope; triage issues; maintain dual-licence terms.
DSP Engineer(s)	Acoustic fidelity; IR stability and quality; performance.	Implement ray/path tracing; amplitude/windowing; validation tools.
Graphics / UI Engineer(s)	Usability; scene clarity; performance of first-person view.	Build 3D view, controls, diagnostics overlay, and UX polish.
Build/Release Engineer	Cross-platform reliability; reproducible builds.	Set up exporters/CI; produce installers/artifacts; DAW validation.
QA / Audio Testers	Functional correctness; audible artefact detection; regression coverage.	Create test plans; perform null tests; verify success criteria per release.

Open-Source Contributors	Clear contribution path; understandable code; labelled issues.	Submit PRs; review code; extend presets; report performance findings.
End Users (Producers, Sound Designers, Researchers)	Useful IRs; easy workflow; stability; predictable latency.	Evaluate usability; provide feedback; request features/presets.
Legal / Compliance (as needed)	Licence clarity; third-party code compliance.	Review notices; ensure JUCE and any HRTF/SAF assets are compliant.

1.3 SUCCESS CRITERIA

Measure success via the following objective, testable criteria:

Functional

- IR export: 32-bit float WAV writes successfully at host sample rate (44.1–192 kHz) with correct channel count (mono/stereo/binaural).
- Determinism: identical seed + parameters produce identical IR files (bit-exact or within floating-point tolerance).
- Preview: in-plugin convolver preview works with bypass and wet/dry mix; latency reported to host (when supported).

Quality & Performance

- No clipping: exported IR peak ≤ −6 dBFS by default (configurable), 0 samples over 0 dBFS.
- Responsiveness: long IR generation executes off the audio thread; UI remains interactive; Cancel stops within 2 s.
- 3D View: ≥ 30 FPS in standalone at 1080p on reference hardware with Medium quality.
- Throughput: ≥ 2× speedup when GPU mode is enabled on supported devices (if GPU option is compiled in).
- Memory: high-budget runs (e.g., ≥ 500k rays, order ≥ 6) complete without OOM or leaks on reference hardware.

Usability

- Room size, source, and listener can be set from the UI (not code-only); values persist across project reloads.
- IR export workflow achievable in ≤ 3 clicks from default state; filename template applied automatically.
- Diagnostics overlay surfaces ray count, paths found, est. RT60, and CPU/GPU usage; can be toggled.

Distribution & Compliance

- Plugin validates and loads in at least two major VST3 hosts (e.g., REAPER, Cubase) and runs as JUCE standalone.
- Builds produced for Windows, macOS, and Linux with documented toolchains; README build steps verified by clean-machine test.

- License notices included for JUCE and any third-party assets; dual-licensing statement present in README.

2. SCOPE & NON-GOALS

2.1 IN-SCOPE (V1: C-ONLY)

- Rectangular room acoustic model with per-surface absorption presets and numeric overrides.
- Two-pass ray/path tracing with seeded randomness; user controls for ray budget and reflection order.
- IR generation and export (32-bit float WAV), with headroom management and optional windowing/gating.
- First-person 3D room visualiser with basic navigation and selectable quality levels.
- VST3 plugin and JUCE standalone builds; parameter automation and state save/restore.
- Non-blocking IR processing with progress and Cancel; diagnostics overlay; error logging.

2.2 DEFERRED / NEXT RELEASES

- In-plugin convolver preview (shipping when stable latency reporting is verified).
- 3D/binaural IR export via external 3D engine integration or SAF build option.
- GPU acceleration path and advanced multi-thread controls.
- Import arbitrary shaped rooms via .ply file instead of using standard rectangular room.
- Preset library expansion (≥ 8 curated presets) and material editor improvements.
- Advanced validation tools (crest-factor, internal sweep test, null-test helpers).

2.3 NON-GOALS

- Use complex diffraction models.
- Frequency-dependent scattering/diffusion at a physically detailed level (beyond simple absorption presets).
- Head-tracked real-time binaural rendering for interactive VR/AR use cases.
- AAX/Audio Units releases (may be considered later, separate licensing/validation required).
- Automatic room correction for loudspeaker calibration (this tool focuses on IR generation, not live correction).
- Large asset libraries (e.g., commercial HRTF datasets) bundled with the plugin.

3. FUNCTIONAL REQUIREMENTS

ASSUMPTIONS

- Plugin formats target VST3 and JUCE Standalone as primary build outputs.
- IR generation uses a two-pass ray/path tracing approach with randomized rays and a reproducible seed.
- Default room is rectangular; user can set room size and place source/listener within bounds.

- IR export uses 32-bit float WAV with optional anti-clipping, windowing, and channel formats (mono/stereo/binaural).
- Integration with a 3D sound engine and Spatial Audio Framework (SAF) is planned.
- Long computations run off the audio thread with progress and cancel.

ID	Title	Description	Input	Output	Constraints
FR-001	Room Geometry (Rectangular)	Model a 3D rectangular room defined by width, length, and height.	Room W/L/H (meters) via UI/params	Updated room model & 3D view reflecting size	Ranges 0.5–50 m; values must keep source/listener inside
FR-002	Surface Properties	Per-surface absorption with presets and numeric override.	Absorption per wall/ceiling/floor (0.0–1.0) or preset	Surface coefficients applied to path/energy calculations	Preset library (e.g., plaster, concrete, carpet); values clamped 0–1
FR-003	Speed of Sound / Environment	Compute speed of sound from temperature; update path times.	Temperature °C (optional humidity)	Recomputed path arrival times and IR timing	Temp range −10–40 °C; deterministic updates
FR-004	Source & Listener Placement	Place sound source and listener (mic) in 3D space and render them in the view.	XYZ positions (meters)	Positions stored and used for tracing and export	Must lie within the room; automatable parameters
FR-005	Reflection Order & Ray Budget	Expose maximum reflection order and ray count to control fidelity/compute.	Max order (1–12); ray count (1k–500k)	Tracer configuration affecting discovered paths and IR density	Higher values increase compute time; validated ranges
FR-006	Two-Pass Tracing	First pass discovers viable paths; second pass refines around them.	Process/Generate IR command	Set of refined path hits and statistics	Two distinct phases with progress metrics
FR-007	Randomized Path Generation (Seeded)	Random ray directions with reproducible results via seed.	Seed (integer)	Deterministic ray field for identical seeds	32-bit seed; new seed yields different micro-structure

FR-008	Memory-Safe Reflection Processing	Bound per-ray state and free buffers to prevent OOM at high budgets.	Ray budget / order	Stable memory usage during long runs	Runs large traces without allocation failures on target machines
FR-009	IR Synthesis	Accumulate arrivals into an IR buffer at host sample rate and selectable length.	Host sample rate; IR length (0.5–10 s)	In-memory IR buffer	Default to host SR; length validated
FR-010	Amplitude Management (Anti- Clipping)	Peak-limit/normalise the IR with headroom to avoid downstream distortion.	Headroom setting (default −6 dBFS)	IR scaled to safe level	No sample > 0 dBFS; peak ≤ configured headroom
FR-011	Windowing & Denoising	Optional late-tail fade (e.g., Hann/Tukey) and noise-floor gating.	Window type & amount; gate threshold	Windowed/ gated IR	User-toggleable; avoids clicks/artifacts
FR-012	Channel Formats	Export mono or stereo/binaural IRs; binaural uses HRTF/3D engine when enabled.	Channel mode selection	IR with 1 or 2 channels	Binaural requires 3D engine/HRTF data
FR-013	File Export UX	Export IR to 32-bit float WAV using a filename template and chosen folder.	Target folder; filename template	WAV file written to disk	Template: RoomReverb_{date}_{sr}_{len}s_{fmt}.wav
FR-014	Plugin Formats	Build as VST3 (and configured JUCE formats).	Build configuration	Plugin binary(ies) recognised by hosts	Meets JUCE/host validator requirements
FR-015	Offline “Process IR” (Non- blocking)	Generate IR on a worker thread with progress and cancel.	User action: Process / Cancel	Background task with status; final IR	Never blocks audio thread; responsive UI
FR-016	In-Plugin Convolution Preview	Optionally convolve incoming audio with latest IR using JUCE Convolution.	Preview toggle; wet/dry mix	Convolved output for monitoring	Reports latency; bypass is glitch-free

FR-017	Latency Reporting	Report processing/convolution latency to the host for compensation.	Computed latency	Host delay compensation active	Accuracy within ±1 sample where supported
FR-018	Parameter Automation & State	Expose controls as automatable parameters and persist via ValueTree/state.	Parameter changes; save/load	Stable automation and restored state	Names/IDs stable; versioned state
FR-019	3D First-Person Display	Render room, source, and listener; basic navigation.	Camera controls (WASD/ mouse)	Interactive 3D view	Target ≥30 FPS in standalone at 1080p
FR-020	Visual Quality Levels	Add lighting/depth cues and simple texturing with selectable quality.	Quality selector	Improved visual feedback	Low/Medium/High profiles
FR-021	UX: Room Size Controls in UI	Expose room sizing in the UI (was code-only).	Sliders/inputs for W/L/H	Immediate update to model & view	Units in meters; min/max enforced
FR-022	Diagnostics Overlay	Show ray count, paths found, est. RT60, CPU/GPU, memory usage.	Overlay toggle	On-screen diagnostics during runs	Minimal overhead; hideable
FR-023	3D Sound Engine Integration	Route early/late components through the 3D engine for binaural IRs.	Enable 3D engine; routing options	3D-processed IR (stereo/ binaural)	Engine dependency must be present
FR-024	Spatial Audio Framework (SAF) Option	Optional build target to compare renderers (engine vs. SAF).	Build flag / runtime switch	Alternate renderer output	A/B consistent sample rate/length
FR-025	GPU Acceleration (Optional)	Offload intersection batches to GPU when available.	GPU enable flag	Faster tracing throughput	Graceful fallback to CPU; device detection

FR-026	Multi-Threaded CPU Path	Parallelise rays across cores with configurable threads.	Threads setting (auto/manual)	Reduced wall-time for large traces	Good scaling (target ≥80% of ideal on 4–8 cores)
FR-027	Real-Time Preview Mode (Experimental)	Debounced re-generation of IR on parameter changes for interactive use.	Interactive mode toggle; debounce time	Updated convolver within budget	Target ≤500 ms update at 128-sample buffer; no dropouts
FR-028	Robust Error Reporting	Surface file I/O, build, and runtime errors in a non-modal log panel.	Errors/exceptions	Readable log entries	Never crashes the UI; actionable messages
FR-029	Validation Tools	Built-in IR peak meter, crest-factor readout, and quick test-convolve (sweep).	User runs validation	Metrics and warnings for unsafe IRs	Flag risky IRs before export
FR-030	Factory Presets	Ship with presets (e.g., Booth/Studio/Live Room/Hall) that recall all params.	Preset selection	Parameters set to preset values	≥8 presets; versioned
FR-031	Host Sample-Rate Conformance	Default to generating/exporting at host sample rate unless overridden.	Host SR; optional override	IR at correct sample rate	Override clearly indicated
FR-032	Cross-Platform Builds	Build on Windows, macOS, and Linux using JUCE exporters.	Build scripts/project files	Artifacts for each platform	Documented toolchains; CI optional
FR-033	Standalone App	Run as a JUCE standalone for testing without a DAW.	Audio device selection; UI controls	Realtime audio path and IR export	Stable on default devices; no host required

NOTES

- Items marked Optional/Experimental may be deferred to later milestones. All long-running work must execute off the audio thread with visible progress and cancellation.

4. NON-FUNCTIONAL REQUIREMENTS

ID	Category	Requirement	Metric / Acceptance
NFR-001	Performance (Audio)	Audio processing must not glitch at 128-sample buffer with preview disabled.	No xruns/dropouts over 10-min session at 48 kHz, 128-sample buffer.
NFR-002	Performance (Computation)	Long IR generation runs on a worker thread; UI stays responsive.	UI input latency ≤ 100 ms during tracing; Cancel stops within 2 s.
NFR-003	Performance (Throughput)	High ray budgets complete in reasonable time on reference hardware.	≥ 500k rays, order 6 completes ≤ 2 min on reference PC; progress shown.
NFR-004	Latency	If preview is enabled, total reported latency is accurate to the host.	Host delay compensation within ±1 sample where supported.
NFR-005	Memory	Bound memory to avoid OOM for large traces; no leaks.	Peak RSS documented; leak-free under sanitizers and long-run tests.
NFR-006	Determinism	Same seed + params → identical IR.	Bit-exact or within 1e-6 RMS float tolerance.
NFR-007	Portability	Build and run on Windows, macOS, Linux using JUCE exporters.	CI/local scripts produce working artifacts on all 3 OSes.
NFR-008	Usability	Core workflow achievable quickly with sensible defaults.	Generate & export IR in ≤ 3 clicks from default state.
NFR-009	Reliability	Robust error handling/logging; no crashes on invalid inputs.	All file/compute errors reported in non-modal panel; fuzzed inputs handled.
NFR-010	Documentation	Users can build and use plugin with provided README.	Fresh machine build passes using documented steps.
NFR-011	Compliance/Licensing	Dual licensing (GPL v3 + commercial) and third-party notices present.	LICENSE and LICENSE-COMMERCIAL included; JUCE & assets credited.

NFR-012	Interoperability	Exported IRs load in common convolution reverbs.	Tested with at least 2 popular plugins/hosts without errors.
NFR-013	Observability	Diagnostics overlay exposes key runtime metrics.	Ray count, paths, est. RT60, CPU/GPU visible; toggle works.
NFR-014	Security/Privacy	No network I/O or telemetry by default.	Binary contains no outbound net calls; opt-in only for any future features.

5. HIGH-LEVEL ARCHITECTURE

The plugin is structured around JUCE's standard Processor/Editor split, a ray/path tracing engine for acoustics, an IR synthesis/export pipeline, and a real-time preview convolver (optional). The first-person view is rendered via JUCE graphics with OpenGL helpers for camera/lighting where available.

- Audio Processor Layer — Owns audio I/O, parameters, background jobs, and (optional) real-time convolution preview.
- Acoustic Model — Rectangular room geometry, per-surface absorption, environment (speed of sound), source/listener placement.
- Tracing Engine — Two-pass path discovery/refinement with seeded random rays; bounded reflection order and ray budget.
- IR Synthesis — Time-of-arrival & amplitude accumulation at host SR; headroom management; optional windowing/gating.
- Export Subsystem — WAV writer (32-bit float) with filename templating and user-chosen directory.
- UI Layer — Parameter controls; 3D first-person room view; diagnostics overlay; progress/cancel for long tasks.
- Integration Points — (Deferred) JUCE Convolution module for preview, 3D engine / SAF for binaural IRs, GPU acceleration hooks.

Proposed component map:

Component	**Purpose / Responsibilities**	**Notes**
PluginProcessor	Audio entry point; parameters; background job orchestration; preview convolver (optional).	Standard JUCE pattern; reports latency to host when preview is active.
PluginEditor	UI controls; 3D view container; diagnostics & logs.	JUCE Components; keyboard/mouse navigation.

RoomModel	Geometry & material properties; validates positions.	Rectangular room; per-surface absorption.
TraceEngine	Two-pass tracing; collects path hits for IR synthesis.	Seeded randomness; thread-pool or GPU hooks.
IRSynth	Bins arrivals into an IR buffer; applies headroom/window/gate.	Generates 1x (mono) or 2x (stereo/binaural).
IRExport	Writes 32-bit float WAV with templated filename.	Template: RoomReverb_{date}_{sr}_{len}s_{fmt}.wav
Diagnostics/Logger	Non-modal log; overlay metrics.	Ray count, paths found, est. RT60, CPU/GPU.
Integrations	Adapters for JUCE Convolution, 3D Engine, SAF.	Compile-time switches; runtime toggles.

6. TESTING & VALIDATION

- **Unit Tests:**
    - Geometry & bounds: room sizing clamps; source/listener inside room; reflection order limits.
    - Determinism: fixed seed reproduces path sets and IR (bit-exact or numeric tolerance).
    - Amplitude pipeline: headroom limiter; windowing/gating produces no clicks; WAV headers valid.
    - Parameter/state: automation IDs stable; ValueTree save/restore round-trips.
- **Golden-Master / Regression:**
    - Reference presets produce IRs equal to stored goldens per SR (44.1/48/96).
    - Numerical tolerances documented to allow benign floating-point drift.
    - Binary compatibility tests for project save/load across versions.
- **Audio Validation:**
    - Null tests: dry vs convolved/bypass correctness; latency-compensated checks when preview is on.
    - Crest-factor/peak checks: IR peak ≤ configured headroom; 0 samples > 0 dBFS.
    - Convolution interop: load exported IR into at least two third-party convolution reverbs; verify identical output within tolerance.
- **Performance & Soak:**
    - Throughput: timed runs at multiple ray budgets; publish table; GPU mode (if built) ≥ 2× speedup on reference GPU.

- Memory: long-run tracing (≥ 30 min) without leaks using sanitizers; peak RSS logged.
- UI soak: 60-min interaction with camera + parameter changes; no lost input; no crashes.

- **Build, Packaging & Host Validation:**
  - Build scripts produce VST3 and standalone on Windows/macOS/Linux from clean machines.
  - Validate in at least two DAWs (e.g., REAPER, Cubase/Bitwig) and run JUCE/host validators.
  - Check licensing files and attributions (GPL v3, commercial option, JUCE, OpenGL resources).

- **Acceptance Criteria Mapping:**
  - Each FR/NFR is mapped to one or more tests above; a release is shippable when all MVP FRs and NFR-001..NFR-010 pass, and no Priority-1 defects remain.

7. ROADMAP & MILESTONES

Milestone	Focus	Key Deliverables	Dependencies	Exit Criteria
M1 — MVP Core	Core room model & IR export	Rectangular room; two-pass tracing; UI room sizing; IR export (32-bit float); diagnostics/log panel	JUCE project builds on 3 OSes	All FR-001..FR-013 pass; NFR-001..NFR-010 green
M2 — UX & Safety	User experience & IR quality	Headroom limiter; windowing/gating; improved first-person view (lighting/textures); presets	M1	No clipping IRs; 3D view ≥30 FPS; presets shipped
M3 — Preview & Latency	In-plugin convolver & host integration	JUCE Convolution preview; wet/dry; bypass; accurate latency reporting	M2	Preview stable in 2 DAWs; null/latency tests pass
M4 — Spatial/Binaural	3D engine / SAF integration	Binaural/stereo IR export path; integration toggle; A/B renderer switch	M3	Interop tests pass; binaural examples included

M5 — Performance	Multithreading & optional GPU	Thread-pool scaling; (optional) GPU hooks; performance benchmarks doc	M3	≥2× speedup on reference GPU; scaling targets met
M6 — Polish & Release	Docs, packaging, contributor path	Expanded README; build scripts; CLA; issue templates; sample projects	M1–M5	Clean-machine build; DAW validation; all blockers resolved