

RTX rendering in your Quest 3S

Feasibility and implementation brief for Centroid's RTX 5060 Ti 16 GB
17 October 2026

A worthwhile renderer prototype. The current NVIDIA fork is not a ready-made path-traced VR upgrade.

Your hardware can run RTX features, and your project already works in Linux PCVR. The missing link is a correct stereo implementation in the selected Godot fork, followed by proof that a representative occupied room meets your frame-time budget. DLSS 5 is real and released, but it is a separate neural appearance feature, with no verified complete integration in this Godot/Linux/OpenXR combination. [N1-N4, G1-G4]

Goal	Assessment
Try desktop path tracing + DLSS	Feasible pilot, most directly with the fork's existing Windows/Vulkan integration. Project assets still need compatibility checks.
Use current fork unchanged for native path-traced VR	Source audit finds single-view tracing, shared output and shared reconstruction context. Engine changes are required.
Add stereo DLSS Super Resolution	A credible first renderer feature: NVIDIA documents one instance per eye. Correct history, inputs and synchronization must be implemented.
Add full path tracing throughout Expedition	Substantial renderer and material integration, plus uncertain 5060 Ti performance. Start with one representative ship room.
Add Frame Generation or DLSS 5	Separate later integrations. FG now has documented per-eye SDK instances; neither feature is proven in this project's headset pipeline.

Recommended order

Resolve current runtime configuration and frame-time attribution while building a small stereo RTX proof. Pursue stereo Super Resolution before full scene path tracing. Expand to Ray Reconstruction and a representative room only after both eyes and their temporal histories are correct. Evaluate generated frames and neural rendering against measured latency and visual consistency.

Scope: all 15 archive parts were verified and reassembled; relevant source, configuration and recorded evidence were inspected. No engine was compiled, no new GPU/headset benchmark was run, and no live session was changed. This report separates source findings, historical observations and recommendations.

What your project already establishes

The recorded production route is Godot 4.7.2 / Forward+ / Vulkan on Centroid's RTX 5060 Ti, through WiVRn to Quest 3S. Physical-session logs identify a focused XR session and working headset use. The recorded headset stream uses H.265/NVENC on the Ti; the separate RTX 5060 handles an Xbox mirror. TTS is routed to Boiler. Those roles do not provide pooled rendering memory or automatic two-GPU scene rendering. [E1-E4]

Observation	What it establishes
Older forest: 33.60 FPS; ~29.83 ms viewport GPU	Controlled synthetic stereo, 1832 x 1920 per eye, 0.7 scale, MSAA 2x. Full-rate synthetic VRS, older foliage code and a different terrain setting. Evidence of that scene's vegetation cost, not current ship performance.
Later focused session: roughly 90 FPS	A project-loop counter at changing viewpoints after asset placement. Useful working-session evidence, without a sustained headset delivery or latency distribution.
October 15: live 11-18 FPS; ~63 ms sampled intervals elsewhere	A separate, later slowdown. A six-second sample found the main thread at ~91% CPU and most sampled cycles in Godot/GDScript/allocation. The remaining hot caller was not established.
Isolated two-person gallery: 15.09 vs 12.96 ms	Flexible vs stiff hair on the RTX 5060, mono 1280 x 720. It did not reproduce the live slowdown and is not a Ti stereo benchmark.

Do not subtract the old monitor numbers

Some reports compared 60-80 ms process monitoring and ~25 ms physics monitoring with 4-9 ms viewport GPU timing. The archive explicitly corrects their interpretation: process/physics monitors are sampled per-second maxima, not matching average frame components. Process includes rendering synchronization; physics represents a peak tick. Subtracting these fields to calculate exclusive CPU cost is invalid. [E5-E6]

Use the corrected source that is already prepared

The attached schema-2 timing code records the metric basis and physics ticks per rendered interval. Prepared recorder and frame-registry changes reduce conversion and redundant copying; their isolated gains do not establish whole-session improvement. The latest pending handover still records them as awaiting runtime assembly. Verify the actual loaded revision before measuring. [E5-E7]

Implication: CPU work and scheduling are concrete investigation leads. A GPU upscaler cannot shorten GDScript callbacks, serialization or physics catch-up. The older forest result also shows why separate CPU and GPU scene measurements are necessary.

The engine work is identifiable

Audited official branch: **nvidia-pt-dlss**, commit **135dff3887**, Godot **4.8.dev**, bundled Streamline headers **2.10.0**. The inspected development branch has the same source tree. The 4.6.3 backport retains the same path-tracing stereo deficiencies. [G1-G4]

Observed source	Required outcome
Path tracing dispatches width x height x 1; ray generation and guide outputs use image2D.	Implement independent eye projections/origins and outputs, either through proper layered tracing or explicit per-eye dispatch. Increasing the dispatch depth alone is insufficient.
copy_output_texture loops over eye destinations but reads the same RT source texture.	Each eye must receive its own image, depth and motion. A headset showing the same traced view twice is not correct stereo.
DLSS loop uses a single context and camera history; RR guides are not selected per eye.	Provide independent per-eye reconstruction contexts, matching inputs, asymmetric projections, motion, jitter and history reset/lifetime handling.
SConstruct disables Streamline outside Windows.	For the existing Linux stack, implement a native NGX/Vulkan path or integrate a separately verified port. DLSS Linux SDK support does not make this fork's integration cross-platform.

A temporal ordering issue deserves an early check

Open PR #14 reports DLSS output being declared as sampled rather than written in the render graph, allowing the displayed result to lag by one frame in its author's test. The audited default source retains that declaration. Review the resource usage/layout correction and verify current-pose output before drawing conclusions about reconstruction blur. The author's measurements are not our reproduction. [G5]

Linux has a real technical route

NVIDIA supplies native Linux NGX libraries for Super Resolution and Ray Reconstruction, and now also distributes an FG library. SR's guide documents Linux and per-eye instances. The FG guide documents Linux integration but still has a Windows-oriented introductory requirements section: query actual feature support on the target driver. The existing fork still needs an integration. [N3-N5]

For a fast desktop visual proof, use the existing Windows/Vulkan path with matching SDK/runtime files. For your established Linux VR experience, native NGX is the relevant engineering route. A Windows boot alone does not repair stereo; Wine adds another compatibility variable and is not the first proof to pursue.

Your materials need deliberate integration

The fork does support custom ShaderMaterial vertex/fragment compilation, ordinary mesh assets, skeleton/blend-shape deformation and MultiMesh work. The issue is the meaning of those shaders in a ray-tracing pipeline. Its custom vertex stage runs after a ray has already hit the underlying mesh, so it cannot establish a different intersection silhouette. [G6-G8]

Project feature	Why it needs attention
TressFX strand hair addons/tressfx/shaders/ tressfx_strand.gdshader	The internal MeshInstance3D is eligible geometry, but the shader builds visible ribbons from VERTEX_ID and a simulated position texture, facing each eye. The underlying mesh is not that final geometry. Its custom light() model is also not the RT lighting stage. A raster overlay may stay visible without becoming correct ray-visible hair.
Terrain, vegetation and water terrain/shaders/; vegetation/shaders/; water/system/	Height placement, wind and FFT waves move vertices in shaders. The acceleration structure must match the visible surface for correct intersections, shadows and reflections. Use a supported deformation/BLAS update or retain a deliberately composed raster path.
Refraction, wet glass, visor and screen effects	The current RT BRDF has no dielectric transmission/refraction lobe. Traced glazing needs that implementation; alpha glass can remain an explicitly identified raster composite. RT screen, depth and normal inputs also alias a dummy slot. [G8-G9]
Sky and atmosphere sky/atmosphere/sky_atmosphere.gd	Compositor callbacks remain in the PT pipeline. LUT generation can remain useful, but a screen-composited effect is not automatically consistent along secondary rays. Check sky illumination, aerial perspective and exterior transitions.
Skin, cloth and changing bodies	Skeletal/blend-shape deformation has an RT update path. ArrayMesh vertex-region writes can invalidate and rebuild BLAS; direct GPU buffer writes need explicit invalidation. Audit deformation cost and temporal motion before scaling the cast.

Appearance is not guaranteed by tracing more rays

The inspected material result carries basic PBR fields; custom light(), subsurface scattering, anisotropy and clearcoat are not automatically reproduced by the current RT BRDF path. Hair, skin and glass need matched visual comparisons. Preserve character identity, authored color and cloth detail when testing later neural appearance enhancement. [G6-G8]

The archive's TressFX extension manifest names Linux x86-64 libraries only; the native binaries and several runtime assets are external to this source export. A Windows project trial must rebuild/package the locally patched extension and relocate data paths. It must not silently compare a scene missing the adopted hair. [E8]

A concrete implementation assignment

1. Make runtime quality controls authoritative

Consolidate the assignments in **ground_environment.gd**, **site_a_xr.gd** and **quest_experience.gd**. Today they force bilinear scaling, disable TAA, set MSAA and repeatedly force scale 0.7. Apply a single resolved configuration after scene initialization. Log actual upscaler, scale, AA, VRS, eye size, refresh, physics rate, device and engine revision. The documented `--render-scale` argument currently has no effective production consumer; implement it before using it in a comparison. [E9]

2. Establish the current physical-headset baseline

Use the current intended assembly, including prepared timing/recorder/frame-copy fixes where approved by the integration owner. Measure a warmed-up occupied ship room and a representative exterior. Keep recording, voice, cast, movement and simulation settings comparable. Existing `--timing-report` is useful; extend it with frame-time percentiles and runtime presentation metrics. A brief `--motion-record-off` comparison can attribute cost, but routine recording must remain part of acceptance. [E5-E7]

3. Build stereo Super Resolution as its own renderer result

Start with raster rendering. Create one reconstruction context per eye. Feed correct eye-layer color/depth/motion, jitter, exposure, current/previous view/projection, feature state and output. Handle new sessions, recentering, teleports, camera cuts and resizing with appropriate temporal invalidation. Fix render-graph write dependencies. Compare equal-output-resolution bilinear/MSAA and DLSS results in motion; report genuine stereo application throughput. [G3, G5, N3]

4. Add correct stereo path tracing and Ray Reconstruction

Use a small fixture with a diffuse surface, glossy object, occluder, emissive light and moving/skinned object. Share scene acceleration data where valid; keep eye-dependent rays and temporal data separate. Verify eye-specific visibility, parallax, depth, motion and reconstructed history during translation and rotation. Then load one real occupied ship room, including actual skin, clothes, adopted hair, glazing and readable displays. Make every missing or raster-composited feature explicit. [G1-G4, E8]

5. Tune the 5060 Ti against a selected experience target

Begin the traced experiment at one sample per pixel and a modest bounce count, with reconstruction and measured render scale. Restrict active scene content through normal visibility, instancing and level of detail. The existing raster VRS setting does not by itself reduce ray-dispatch sample count. Do not generalize an empty-room result to the dome, forest, water or full cast. [G1, G4]

6. Evaluate Frame Generation and DLSS 5 separately

Only after the base stereo path is stable, add FG with per-eye instances, synchronized pairs, application-owned pacing and valid OpenXR pose/depth metadata. Confirm generated images actually reach the headset, rather than only its desktop mirror. For DLSS 5, first obtain a complete usable integration package and demonstrate two-eye consistency. Report base cadence, displayed cadence, added GPU cost and latency separately. [N1-N5]

What to target on the 5060 Ti

The 16 GB card has RT and Tensor hardware and a hardware encoder. Capacity is useful for textures, acceleration structures and temporal buffers, but it does not guarantee a fast enough stereo frame. The Quest display and the PC runtime's render target are different quantities; always record the runtime eye size. [N6-N8]

Refresh rate	Whole stereo update interval	Use in this investigation
72 Hz	13.89 ms	Practical first native-cadence target for the quality experiment.
80 Hz	12.50 ms	Optional intermediate rate if exposed by the active runtime.
90 Hz	11.11 ms	Next target once a representative room is stable.
120 Hz	8.33 ms	Later option if measured headroom permits.

Intervals above are 1000 / refresh rate; they are not separate per-eye budgets or guarantees of comfort. Leave operating headroom and measure missed deadlines. GPU execution and CPU work can overlap, while encoding, networking and decoding add pipeline latency; do not simply sum unrelated last-sample counters.

Rendering scale has a squared pixel effect

At 1832 x 1920 per eye, the pair contains 7.03 million output pixels. A 0.7 linear scale shades roughly 49% as many pixels before upscaling; 0.67 shades roughly 45%; 0.5 shades 25%. Geometry, acceleration updates and simulation do not scale down by the same ratios. DLSS also has its own cost, so these are pixel counts, not speedup predictions. [N7, N9]

Spend quality where you will notice it

For the occupied room, inspect a face at conversation distance, hair against a bright window, moving hands, cloth, a mirror or glossy surface, glass and world-space text. Evaluate temporal swimming, fine-strand loss, disocclusion smears and left/right mismatch. For the exterior, include foliage, water and a moving viewpoint. A stable clear view may be preferable to more bounces or a stronger reconstruction preset.

Use foveation deliberately

The project requests XR VRS; the old synthetic benchmark supplied a full-rate VRS texture and therefore did not demonstrate savings from foveation. Verify what the active runtime supplies. Quest 3S has no eye tracking, so the immediate option is fixed foveation. Foveated encoding and raster shading-rate reduction affect different costs, and neither automatically implements foveated path tracing. [E3, E9, N7-N8]

Engineering judgment: limited path-traced interiors are worth testing on your hardware after integration. There is no evidence here to promise native 72/90 Hz full-scene path tracing with dense dynamic people, vegetation and water. The useful answer comes from the representative room, not a desktop marketing multiplier.

Cadence, streaming and acceptance

Keep physics behavior comparable

The current XR startup sets physics ticks from headset refresh. Choosing 72 rather than 90 Hz therefore also changes physics rate unless the configuration is separated. Several body/cloth studies use 90 Hz. Record and deliberately control this during comparisons rather than accidentally changing simulation and rendering together. [E9]

Hair has another clock dependency: the retained native implementation advances with render updates, caps large dt and applies constraints per render step. A separate opt-in fixed-clock adapter reproduced guide endpoints at 30/60 render caps, but 90 Hz solving required more work per rendered frame. It was not adopted in the recorded runtime. Preserve the selected pose/collision clock and measure its real cost when the renderer changes. [E4]

Retain your working transport baseline

Use the existing Ethernet to 5 GHz router to Quest 3S route while measuring the renderer. Start with its working H.265/NVENC setup. AV1 is a possible measured comparison where the installed WiVRn and headset negotiate it; it is not a remedy for an overloaded application. The archive does not include the effective private WiVRn bitrate/foveation configuration. WiVRn's current documentation also cautions about NVIDIA latency above default rendering resolution. [E2, N8]

Deliverable	Acceptance evidence
Correct images	Both eyes show appropriate parallax and different visibility; no copied-eye image, shared history contamination, missing required geometry or stale camera frame.
Representative motion	Inspect head rotation and translation, disocclusion, recentering, hands, moving people and thin geometry in the actual headset.
Honest performance	Record engine/source/driver, GPU UUID, eye dimensions, scale, scene/cast, CPU/GPU timing distribution, physics ticks per frame, base stereo FPS and missed deadlines.
Complete delivery pipeline	Where available, record encoder, network, decoder and compositor/reprojection data. Separate actual base frames, generated frames and display refresh.
Useful comparison	Same scene, viewpoint, output resolution, simulation clock, recording and service conditions. Isolate startup/compilation. Report image improvement alongside incremental cost.

A 30 FPS renderer would leave roughly 33 ms between new scene images, even if later stages present more often. That can be useful for a deliberate preview or a tested reprojection mode, but it is not evidence of native 90 Hz scene rendering. NVIDIA's current FG guide permits per-eye instances while assigning synchronization and presentation to the application. [N4]

Evidence register: your archive

Archive integrity: 15 parts; 1,673,689,119 assembled bytes; 5,466 ZIP members. Every part matched its supplied checksum. The complete SHA-256 matched the supplied value. Regular members were extracted with CRC checks; 27 repository-relative symlinks were restored with targets constrained to the extracted repository.

4641e15ac96140f13a76a731970bf68eeda2beb7876cb470fc813c54a9c6a0a4

Paths below are relative to expedition-main/. The ZIP is a source snapshot, not proof of the currently deployed revision. The last recorded active assembly is 28df6b7ed5837a1e65d0553d7e137d0b96a88045; later prepared files are listed in the pending-session handover.

ID	Source and relevant location
E1	AGENTS.md; godot/project.godot. Device/service responsibilities, Forward+ features, Jolt, physics 90 Hz and OpenXR.
E2	docs/graphics/QUEST-CENTROID-LIVE-SESSION.md (5-48); QUEST-CENTROID-SESSION-HISTORY.md (75-107); scripts/quest_wivrn_start_server.sh; scripts/quest_start_player_on_connect.sh; scripts/quest_cast_server.py.
E3	docs/graphics/QUEST-RUNTIME-INVESTIGATIONS.md (41-57); godot/quest/xr/steady_render_comparison.json; steady_render_probe.gd; README.md. Historical controlled stereo scope and timing method.
E4	docs/graphics/hair/HABITAT-DYNAMICS-HANDOFF.md (73-109). Native hair cadence, fixed-clock study, physical-session CPU sample and mono gallery comparison.
E5	docs/QUEST-3S-EXPERIENCE-PLAN.md (655-683). Slowdown, direct callbacks, timing correction and queued optimizations.
E6	godot/quest/telemetry/quest_runtime_timing.gd (38-54); quest_motion_recorder.gd (111-127,181-205); godot/quest/spatial/quest_frame_registry.gd. Corrected timing basis and source optimizations.
E7	docs/handovers/QUEST-PENDING-SESSION-ASSEMBLY.md (3-6,83-88). Prepared runtime changes and explicit deployment distinction.
E8	godot/addons/tressfx/tressfx.gdextension; shaders/tressfx_strand.gdshader (65-163); godot/quest/clinic_gallery/strand_hair_binding.gd. Linux packaging, texture-driven ribbons and custom lighting.
E9	godot/ground/ground_environment.gd (66-74); site_a_xr.gd (76-87); godot/quest/quest_experience.gd (38-44,104-105). Actual quality assignments and refresh/physics coupling.
E10	godot/terrain/shaders/terrain_ground.gdshader; vegetation/shaders/veg_plant.gdshader; water/system/water_surface.gdshader; sky/atmosphere/sky_atmosphere.gd. Deformation, screen inputs and compositor integration.

08 / REFERENCES

Public sources and exact engine pin

NVIDIA fork pin: 135dff3887a3d2ccf45cafb9bdffa7b718e1e76

G1 [Path tracing dispatch and compositor ordering](#)

lines 47-60, 309-470

G2 [Ray output copying and acceleration updates](#)

lines 635-813, 2492-2511, 3339-3353

G3 [DLSS per-view loop and shared context](#)

lines 1843-1900

G4 [Platform gate; supported path-tracing controls](#)

lines 608-612; also doc/classes/Environment.xml, 203-221

G5 [Open PR 14: output resource usage and reported one-frame lag](#)

author report; open at review time

G6 [RT shader compiler](#)

custom stages and built-in mappings

G7 [RT custom vertex/fragment execution](#)

post-intersection vertex call; basic material result

G8 [Dummy screen/depth/normal inputs in RT](#)

lines 34-38

G9 [RT BRDF: no dielectric transmission lobe](#)

lines 106-107, 1100-1160; also open NVIDIA-RTX/godot issue 11

N1 [NVIDIA: DLSS 5 release, separate neural appearance feature](#)

1 September 2026

N2 [NVIDIA: October DLSS 5 model announcement](#)

13 October 2026; announced availability 16 October

N3 [NVIDIA DLSS SR programming guide](#)

31 March 2026; section 3.17, printed page 31

N4 [NVIDIA DLSS FG programming guide](#)

June 2026, SDK 310.7.0; printed pages 3,7,23,40-43

N5 [NVIDIA public SDK and Streamline records](#)

also NVIDIA/DLSS lib/Linux_x86_64/rel and NVIDIA-RTX/Streamline changelog.txt

N6 [NVIDIA RTX 5060 family specifications](#)

GPU capabilities are not a project benchmark

N7 [Meta Quest device comparison and rendering rates](#)

also vr/documentation/unreal/unreal-advanced-rendering/

N8 [WiVRn official project and configuration documentation](#)

Quest 3S, Linux OpenXR, encoder options and latency caveat

N9 [Godot resolution scaling](#)

scale is per axis; CPU limits do not disappear with lower resolution

DLSS 5 access note: Streamline changelog 2.14.0 names `sl.dlss_nr`, while the inspected public `include/docs` inventory did not expose a complete NR integration guide/header. No claim is made that all SDK access is unavailable; a usable package for this exact custom engine, platform and stereo route remains unverified.