2026 Real-Time Shading Ecosystem Survey Report

AI-assisted summary of survey results

July 2026

Survey conducted June 16 – July 10, 2026

404 Respondents

Contents

Executive Summary

Methodology

The purpose of this survey is to gauge the real-time shader developer community's current use of shading languages, tools, platforms, and APIs — and to identify where standardization efforts from The Khronos Group can have the most impact.

- The survey was open June 16 – July 10, 2026 and received 404 responses.
- Distribution channels included multiple developer Discord servers (Slang, Graphics Programming, Khronos, Vulkan), LinkedIn, Reddit, Mastodon, and BlueSky. Full channel list is provided in the Appendix.
- All quantitative results use the full response count for that question; some questions were skipped by a small number of respondents.
- Open-ended responses are quoted verbatim throughout this report. Responses have not been filtered for repetition — frequency of a theme is itself a signal.
- Editorial commentary from the Khronos team appears in light blue callout boxes.

The spreadsheet containing the full set of survey responses is available to Khronos members upon request.

Key Takeaways

The following themes emerge consistently across all 17 survey questions:

1. Shader debugging is the single biggest pain point, chosen by 53% of respondents as a top-3 issue — more than 15 percentage points ahead of the next item. A cross-vendor shader debugging standard received the highest Khronos activity rating (4.18 out of 5).
2. The debugging pain score (0–100 slider) averaged 52, with 40% of respondents rating it 61 or above. Nearly 1-in-10 gave a score of 90 or higher — indicating a population segment experiencing severe daily friction.
3. Cross-platform porting affects the majority of developers: 64% spend at least some effort adapting shaders across platforms, APIs, or tools. Nearly 10% call it a significant or largest engineering cost.

4. Linux usage of 74% is striking. While Windows leads at 88%, the gap is smaller than expected for an audience primarily from games. This signals a significant non-games developer base in scientific visualization, HPC, and AI — communities with very different tooling needs from the games majority.
5. SPIR-V direct usage (39%) reflects the significant number of developers working at the IR level or through tools.
6. Slang has achieved significant adoption, ranking #4 in usage. However, compiler instability, AI-generated commits, and documentation gaps are threatening trust with early adopters.
7. Pointers, function calls, and true addressability are the most-requested missing language features. Developers increasingly want GPU languages to feel like real systems programming languages, citing CUDA as the benchmark.
8. 73% of respondents are open to participating in future Khronos working groups or feedback sessions — the community is motivated to engage and contribute.
9. Apple/Metal issues were mentioned throughout free-text responses. The inability to target Apple platforms without a separate language/API stack is a recurring theme that structured questions do not fully capture.
10. VCC/Shady (Vulkan Clang Compiler) and Rust-GPU received notable mentions in open-text questions about what Khronos should incubate — both represent the "use a real programming language" school of thought.
11. A coherent minority argues that more standards are not the answer — they want fewer, better-implemented tools. Their implicit request: fix DXC, stabilize Slang, and stop adding languages before the existing ones are trustworthy.

Q1: Who Responded — Primary Role

Answered: 404 | Skipped: 0

Graphics Programmers and Game/Engine Developers together constitute 64% of respondents, which is precisely the audience most directly affected by shader tooling and language quality. The survey reached its intended population effectively.

Role	Respondents	Percentage
Graphics Programmer	140	34.65%
Game / Engine Developer	117	28.96%
Rendering Engineer	49	12.13%
Tools / Middleware Developer	18	4.46%
Researcher / Academic	18	4.46%
Other (please specify)	14	3.47%
Tech Artist / Technical Director	12	2.97%
Shader Developer	12	2.97%
Shading Language Designer / Compiler Developer	10	2.48%
Hardware Vendor	5	1.24%
Driver Developer	7	1.73%
Graphics API Designer	2	0.50%
DCC Application Developer	0	0.00%

Graphics Programmer		34.6% (n=140)
Game / Engine Developer		29.0% (n=117)
Rendering Engineer		12.1% (n=49)
Tools / Middleware Dev		4.5% (n=18)
Researcher / Academic		4.5% (n=18)
Other		3.5% (n=14)
Tech Artist / Tech Dir		3.0% (n=12)
Shader Developer		3.0% (n=12)
SL Designer / Compiler Dev		2.5% (n=10)

Q1 — Primary Role (n=404)

"Other" write-ins included: hobbyist (×3), AI Software Architect, ML Engineer, computer vision / 3D reconstruction, GPGPU/kernel developer, web developer, and VFX Artist.

> The overlap of hobbyists, academics, and compute developers in the "Other" category confirms that shader tooling is increasingly used outside traditional game/graphics contexts — especially for GPGPU, AI inference, and scientific simulation.

Q2: Which Domains Do You Work In?

Answered: 403 | Skipped: 1 | Select all that apply

Games clearly dominates, but the long tail of non-games domains is significant. The combined middleware/tools (26%), scientific visualization (16%), and hardware development (9%) segments represent a substantial developer community with different needs from the games majority.

Domain	Respondents	Percentage
Games	317	78.66%
Middleware / Tools Development	104	25.81%
Scientific Visualization	65	16.13%
Industrial Visualization	51	12.66%
Hardware / Driver Development	36	8.93%
Film / VFX	37	9.18%
Other (please specify)	36	8.93%
Real-time XR	33	8.19%
Automotive	17	4.22%

Games		78.7% (n=317)
Middleware / Tools Dev		25.8% (n=104)

Scientific Visualization		16.1% (n=65)
Industrial Visualization		12.7% (n=51)
Film / VFX		9.2% (n=37)
Hardware/Driver Dev		8.9% (n=36)
Real-time XR		8.2% (n=33)
Automotive		4.2% (n=17)

Q2 — Domains (n=403, select all)

"Other" write-ins included: HPC/AI/GPGPU (multiple), Simulation, Generative Art, Geospatial Visualization, Television Software, System UI, maps, video playback, GPGPU for audio, demos, retail fashion, and interactive immersive art.

> The HPC, AI, and GPGPU write-ins reinforce that SPIR-V and compute-focused shader tooling serve a population well beyond traditional real-time rendering. These users likely experience the "toy language" frustration more acutely, as their compute needs push against shader language limitations designed for rasterization.

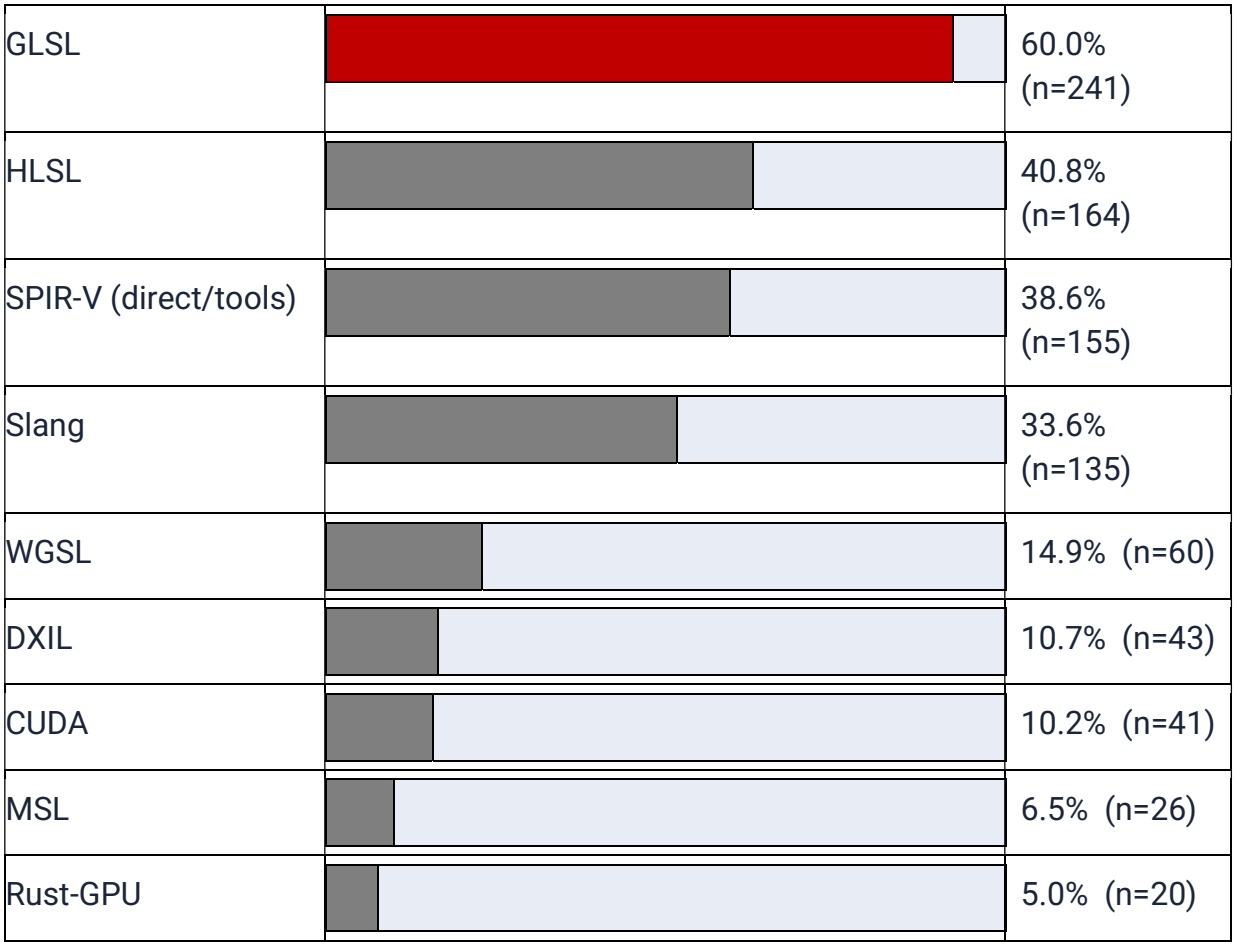
Q3: Shading Languages in Active Use

Answered: 402 | Skipped: 2 | Select all that apply

GLSL remains the most widely used shading language. Slang's adoption is growing — at 34%, it is approaching HLSL (41%). SPIR-V direct usage (39%) reflects the significant number of developers working at the IR level or through tools.

Shading Language	Respondents	Percentage
GLSL	241	59.95%
HLSL	164	40.80%
SPIR-V (directly or via tools)	155	38.56%
Slang	135	33.58%
WGSL	60	14.93%
DXIL	43	10.70%
CUDA	41	10.20%
MSL	26	6.47%
OpenCL C/C++	25	6.22%
Other (please specify)	25	6.22%
A proprietary language	24	5.97%
Rust-GPU	20	4.98%
HLSL 2018	36	8.96%
MaterialX	9	2.24%

Shading Language	Respondents	Percentage
OSL	6	1.49%
MDL	5	1.24%



Q3 — Shading languages in active use (n=402, select all)

"Other" write-ins included: NZSL (Nazara Shading Language), BSL (Blender Shading Language) (×3), Nabla C++/HLSL (×4), SYCL (×2), GDShader, TSL (Three.js Shader Language), Unreal Engine Material Blueprints, Julia (Metal/CUDA backends), PSSL, and custom/internal languages.

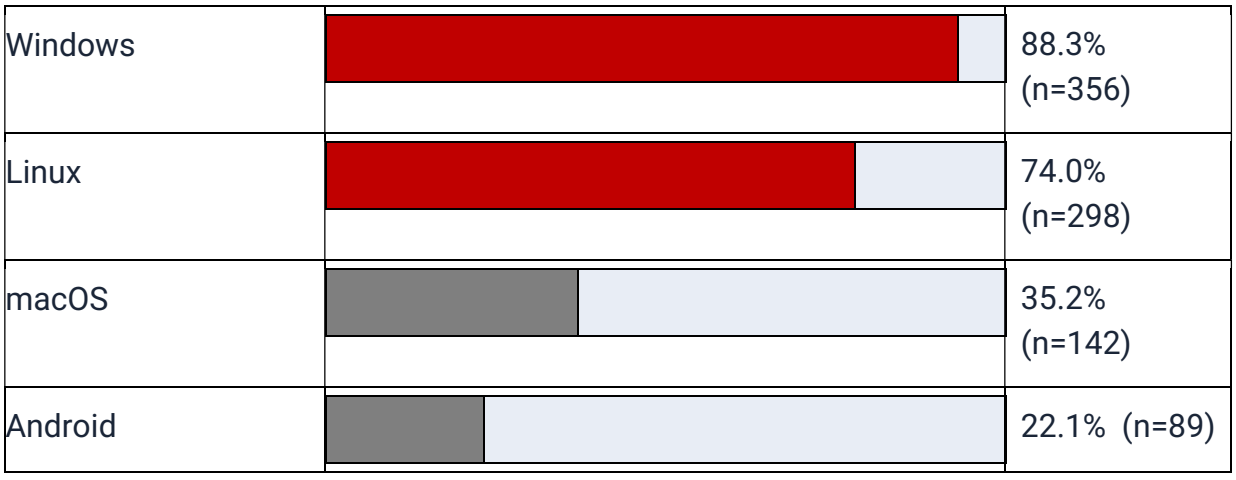
While Slang adoption continues to grow, open responses in the survey indicate that building greater confidence will be important. Some developers highlighted areas for improvement, particularly around compiler stability and the handling of AI-generated contributions.

Q4 & Q5: Target Platforms and Graphics APIs

Q4 Answered: 403 | Q5 Answered: 403 | Select all that apply

Target Platforms (Q4)

Platform	Respondents	Percentage
Windows	356	88.34%
Linux	298	73.95%
macOS	142	35.24%
Android	89	22.08%
Game Consoles	86	21.34%
Web	77	19.11%
iOS	53	13.15%
Embedded / Automotive	15	3.72%
Cloud Rendering	7	1.74%



Game Consoles		21.3% (n=86)
Web		19.1% (n=77)
iOS		13.2% (n=53)
Embedded/Auto		3.7% (n=15)

Q4 — Target platforms (n=403, select all)

Linux at 74% is the most striking finding in this question. While Windows leads at 88%, the gap is smaller than expected for an audience primarily from games. This reflects the significant compute/HPC/academic developer population. "Other" write-ins included FreeBSD, OpenBSD, NetBSD, Haiku, and "GPU Emulation."

Target Graphics/Compute APIs (Q5)

API	Respondents	Percentage
Vulkan	311	77.17%
DirectX 12	150	37.22%
OpenGL / OpenGL ES	136	33.75%
WebGPU	88	21.84%
Metal	69	17.12%
DirectX 11 or earlier	51	12.66%
Other (please specify)	38	9.43%

Vulkan		77.2% (n=311)

DirectX 12		37.2% (n=150)
OpenGL / GLES		33.8% (n=136)
WebGPU		21.8% (n=88)
Metal		17.1% (n=69)
DirectX 11–		12.7% (n=51)

Q5 — Graphics/compute APIs (n=403, select all)

"Other" write-ins were dominated by console-specific APIs (mentioned by over 15 respondents, most citing NDA restrictions). Additional write-ins included OpenCL (×3), CUDA (×3), AGC (×3), NVN, SDL3 GPU / SDL_GPU (×2), LevelZero, ISPC, wgpu, and proprietary Sony APIs.

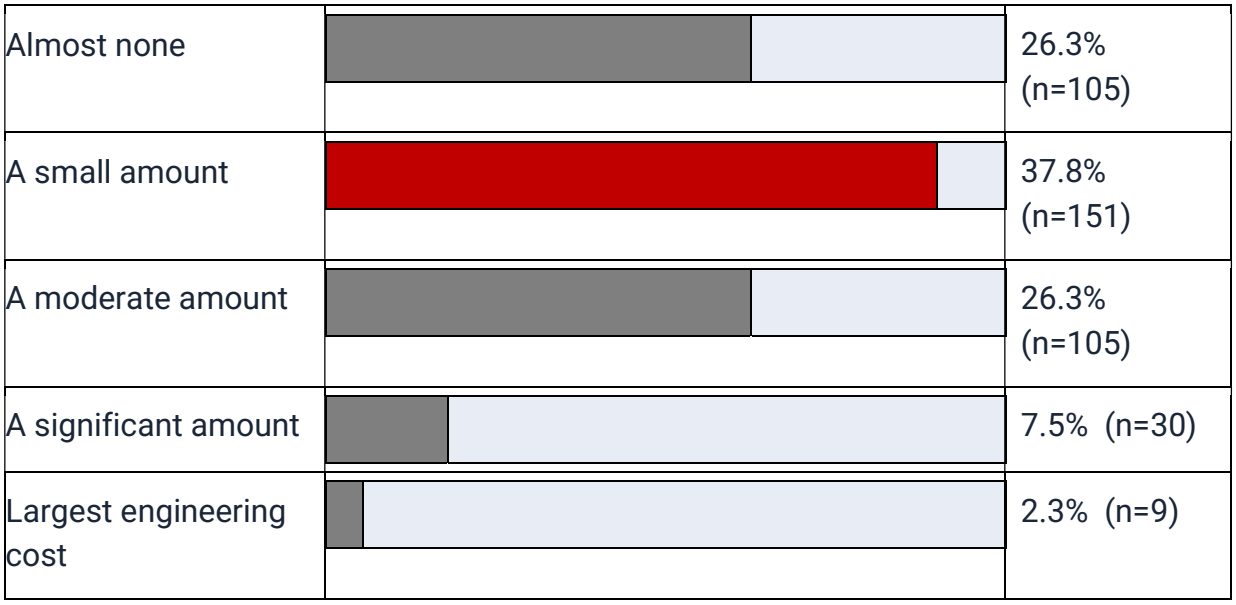
> One respondent noted with visible frustration: 'You forgot to list compute APIs here, shaking my head.' This encapsulates a broader theme: the survey, like much of Khronos' current tooling emphasis, is seen by some as graphics-centric while real-world usage has expanded significantly into compute.

Q6: Cross-Platform Shader Porting Effort

Answered: 400 | Skipped: 4

Question: How much effort do you spend adapting shaders between platforms, APIs, or tools?

Response	Respondents	Percentage
Almost none	105	26.25%
A small amount	151	37.75%
A moderate amount	105	26.25%
A significant amount	30	7.50%
It is one of our largest eng. costs	9	2.25%



Q6 — Cross-platform porting effort (n=400)

Spend at least some effort porting	64% (256 respondents)

Spend almost no effort	36% (144 respondents)
Call it significant or a major cost	9.75% (39 respondents)
Combined moderate + significant + largest	35.75% (143 respondents)

The 26% who report "almost none" likely skew toward single-platform developers — those targeting only one API (e.g., pure Vulkan on Linux) or using a high-level abstraction layer that handles cross-compilation. The 9.75% who rate it as a significant or largest cost represent a population experiencing real business impact from portability friction.

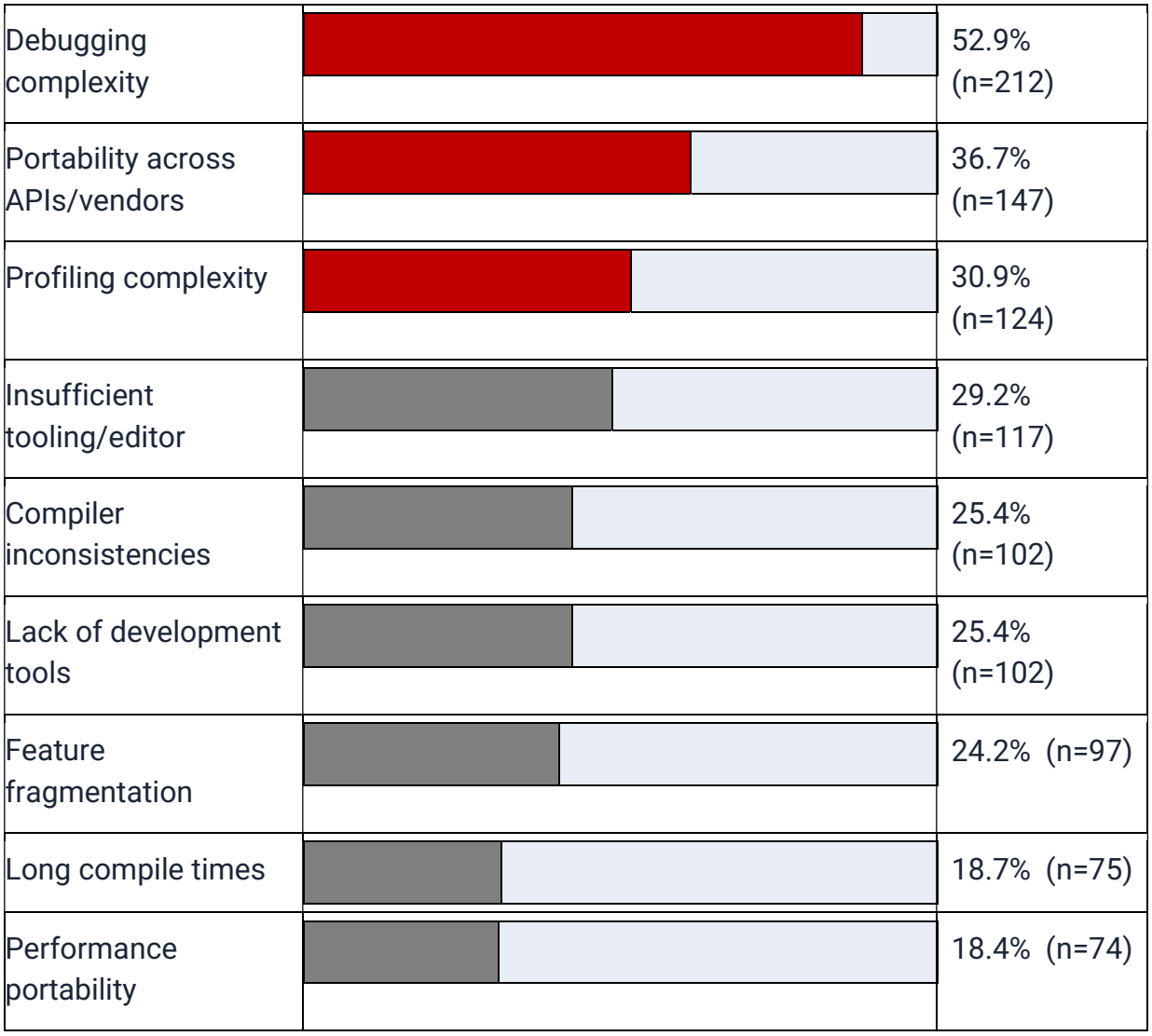
Q7: Biggest Pain Points When Working With Shaders

Answered: 401 | Skipped: 3 | Pick your top 3

Debugging complexity dominates by a wide margin — chosen by more than half of all respondents. The gap between #1 (debugging, 53%) and #2 (portability, 37%) is 16 percentage points, suggesting debugging is not merely a shared concern but an acute one.

Pain Point	Respondents	Percentage
Debugging complexity	212	52.87%
Portability across APIs/vendors	147	36.66%
Profiling complexity	124	30.92%
Insufficient tooling / editor support	117	29.18%
Compiler inconsistencies	102	25.44%
Lack of development tools	102	25.44%
Feature fragmentation	97	24.19%
Long compile times / iteration speed	75	18.70%
Performance portability	74	18.45%
Other (please specify)	52	12.97%
Toolchain integration	52	12.97%
Shader asset management	59	14.71%

Pain Point	Respondents	Percentage
Validation and conformance	54	13.47%
Hardware limitations	46	11.47%
Cross-compilation quality	66	16.46%
Material interchange	13	3.24%



Q7 — Top pain points (n=401, pick top 3)

Selected Open-Text Responses (Q7 "Other")

The following verbatim responses illuminate themes that the structured options did not fully capture:

"Languages just aren't good, I want to use a REAL programming language like C or Rust."

“Shaders can be so much more...”

"DXC buggy debug info generation"

"GPU crash debugging"

"Solving the Shader Permutation Problem."

"Code and data sharing with CPU side"

"Shader dialect restrictions (no pointers, indirect calls, etc)"

"Lack of a large library ecosystem"

"Bugs and lack of new Vulkan features in DXC"

"bugs and crashes in the slang compiler"

"Vibe coding causing Slang bugs"

> Although Slang adoption is growing, open responses across several questions pointed to ongoing challenges around compiler stability, crashes, and AI-generated commits. These issues were not reflected in the structured questions. The results indicate that further investment in reliability and quality processes could help strengthen developer confidence as adoption continues to expand.
>
> .

Q8: How Painful Is Shader Debugging and Profiling?

Answered: 379 | Skipped: 25

Question: On a scale of 0–100, how painful is shader debugging and profiling across your target platforms today?

Mean score	52.3 out of 100
Median score	50.0
Standard deviation	23.5
Respondents scoring 60+	156 (41.2%)
Respondents scoring 80+	59 (15.6%)
Respondents scoring 90+	23 (6.1%)
Respondents scoring 100	13 (3.4%)
Respondents scoring 0–20	46 (11.8%)

0–20 (low/no pain)		11.8% (n=46)
21–40		19.5% (n=76)
41–60		28.5% (n=111)
61–80 (high pain)		30.8% (n=120)

81–100 (extreme pain)		9.3% (n=36)

Q8 — Debugging pain score distribution (n=379)

The distribution is notably skewed toward the upper range. The 61–80 band is the single most populated bucket (31%), and combined with the 81–100 group, 40% of respondents experience shader debugging pain in the "high" range. The 13 respondents who gave a score of exactly 100 — the maximum — are expressing maximum frustration.

The 32% who score below 40 are likely single-platform developers, those using integrated debugging in their engine/editor, or those who rarely need to debug at the shader level. Their lower pain does not negate the severity expressed by the majority.

> The combination of Q7 (debugging as #1 pain point at 53%) and Q8 (average pain score of 52, median 50) makes the case for a cross-vendor shader debugging standard as unambiguously as any survey could. The fact that Q13 rates this the highest-value Khronos activity (4.18/5) closes the loop: the community has identified the problem, knows what the solution looks like, and is asking Khronos to deliver it.

Q9: Areas Most in Need of Stronger Standardization

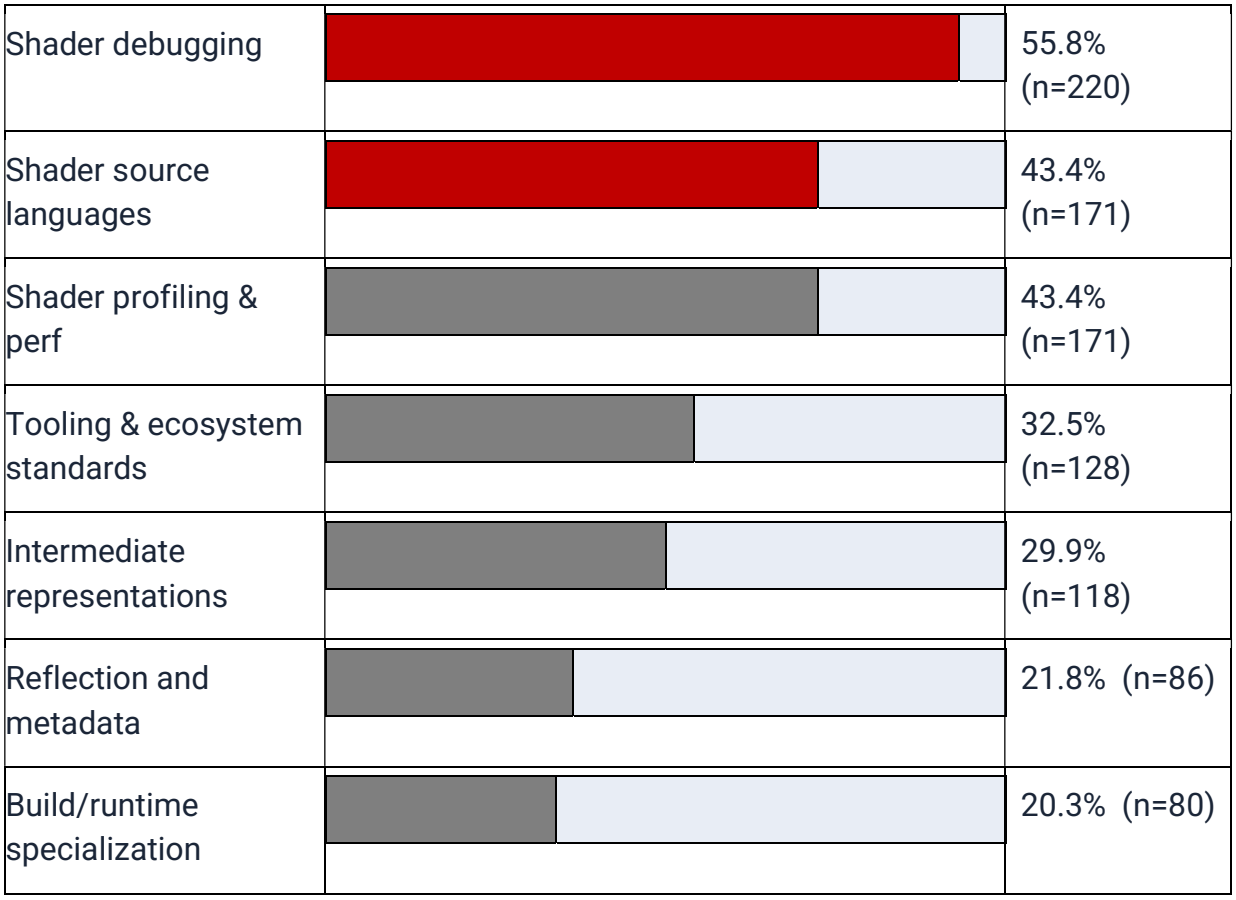
Answered: 394 | Skipped: 10 | Pick your top 3

Shader debugging leads standardization needs at 56%, consistent with its dominance in Q7

Shader source languages and profiling tools were both selected by 43% of respondents as top challenges, highlighting two important areas where further improvements could significantly enhance the shader development experience.

Area	Respondents	Percentage
Shader debugging	220	55.84%
Shader source languages	171	43.40%
Shader profiling and performance	171	43.40%
Tooling and ecosystem standards	128	32.49%
Intermediate representations	118	29.95%
Reflection and metadata	86	21.83%
Build/runtime shader specialization	80	20.30%
Shader package/distribution formats	36	9.14%
Conformance testing	37	9.39%
Neural / AI-assisted shading	14	3.55%
Shader security and validation	27	6.85%

Area	Respondents	Percentage
Material definitions	28	7.11%
Material interchange	30	7.61%
Other (please specify)	24	6.09%



Q9 — Standardization priorities (n=394, pick top 3)

Selected Open-Text Responses (Q9 "Other")

"In my opinion, HLSL is the only real contender and it needs to work everywhere despite not being a Khronos-sanctioned language."

"SPIR-V everywhere please. No DXIL, and definitely no WGSL. I know getting Apple on board is hard but maybe there's a path."

"One cross-platform API that is simple enough for 80% of the work"

"We need LESS standards, discontinuing SPIRV and WGSL could be a good start"

"PSSL is our most important target, all Khronos efforts are nice but pointless given our situation"

"Libraries / Modules beyond bare bones #include"

"a single shader IR target for all backends (webgpu/vulkan/etc)"

"we don't need shader languages. What we need are compilers that can emit SPIRV from existing languages"

> The "we need fewer standards" minority is coherent and worth taking seriously. Their argument: fragmentation is manageable if existing tools work reliably. A stable, bug-free DXC or glslang would solve more real problems than a new standard. This tension between breadth of standardization and depth of implementation quality runs throughout the survey.

Q10: Value of Specific Khronos Initiatives

Answered: 401 | Skipped: 3

Question: How valuable would the following be to you? (Rate each, 1 = not valuable at all, 5 = extremely valuable)

Initiative	Avg (of 5)	Rating Distribution (1→5)
Better cross-API shader translation tools	3.47	10% / 15% / 23% / 23% / 29%
Improved SPIR-V ecosystem and tooling	3.76	6% / 11% / 20% / 28% / 35%
Official Khronos shader validation/conformance	3.52	7% / 12% / 28% / 25% / 27%
Standardized shader libraries / materials	2.96	22% / 18% / 22% / 19% / 20%
Stronger common shading language (e.g. Slang)	3.32	21% / 12% / 15% / 19% / 34%

Improved SPIR-V ecosystem scores highest among the five options at 3.76, followed by the official conformance suite (3.52) and cross-API translation tools (3.47). The "standardized shader libraries/material definitions" option scores lowest (2.96) — not because material interchange is unimportant, but because it is seen as less urgent than debugging, language, and toolchain problems.

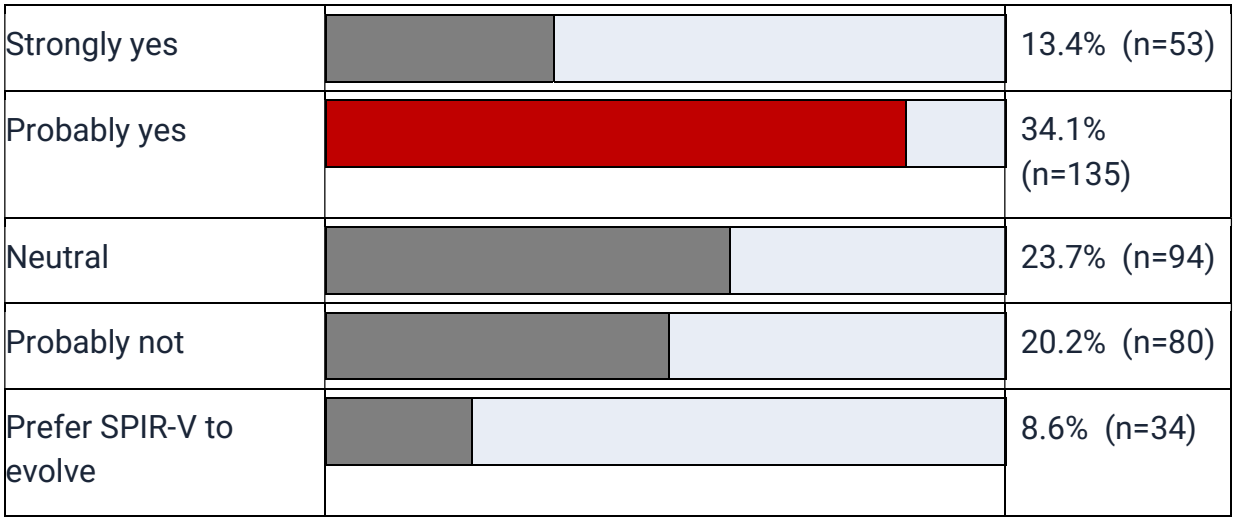
The bimodal distribution on the "stronger common shading language" option is notable: 21% rate it 1 (not valuable at all) while 34% rate it 5 (extremely valuable). This reflects the fundamental tension between developers who want language convergence and those who believe the existing landscape has too many languages already.

Q11: Would a Higher-Level Shader IR Benefit Your Project?

Answered: 396 | Skipped: 8

Question: Would your project benefit from a standardized, higher-level shader IR (above SPIR-V)?

Response	Respondents	Percentage
Strongly yes	53	13.38%
Probably yes	135	34.09%
Neutral	94	23.74%
Probably not	80	20.20%
I would rather see SPIR-V evolve	34	8.59%



Q11 — Support for a higher-level shader IR (n=396)

Lean toward a higher-level IR (strongly + probably)	47.5% (188 respondents)
Neutral / undecided	23.7% (94 respondents)
Prefer SPIR-V to evolve instead	8.6% (34 respondents)
Probably not	20.2% (80 respondents)
Combined lean against (probably not + prefer evolve)	28.8% (114 respondents)

Nearly half of respondents lean toward a higher-level IR, but with 24% neutral and 29% leaning against, this is not the clear mandate that the debugging and profiling standardization questions produce. The community is interested but not united on this direction.

Q12: Missing Capabilities in Today's Shader Languages or IRs

Answered: 151 | Skipped: 253 (open text, optional)

Although only 151 respondents answered this open-ended question, the responses are substantive and theme-dense. Several categories emerge repeatedly:

Theme 1: Pointers, Function Calls, and Addressability

This is by far the most common theme, appearing in roughly a third of substantive responses. Developers want:

- Generic pointers and pointer arithmetic
- Function pointers and indirect calls
- Recursion
- Generic address space (as in OpenCL / CUDA)
- SSBO as function arguments

"Pointers and references: I have spent literal MONTHS working around Vulkan/GLSL's complete lack of proper reference semantics."

"Physical addressing in most/all storage classes. Generic pointers. Unstructured/more permissive control flow."

"really GPU compute, function pointers, recursion, all that stuff that GPUs can actually do but which shader languages artificially hide."

"proper generic address space pointers in Vulkan like OpenCL / CUDA, there is nothing like it available across platforms"

Theme 2: Debugging and Introspection

Debugging-related missing features appear in roughly 20% of responses:

- Standardized assert / printf in shaders
- Explicit uniformity annotations (not just heuristic)
- Better debug info from compilers (especially DXC)

- Vendor-to-high-level-IR mappings for debugging
- Bindless debugging support

"Standardized debug features like assert/printf. Explicit uniformity (non-uniform annotation)."

"Having a standardized consistent mapping between low-level IHV assembly and high-level source"

"Some kind of debug markers for debugging shader ISA."

Theme 3: Modular Programming — Imports, Modules, Libraries

Module and library support is mentioned in roughly 15% of responses:

- Proper module/import system beyond #include
- Standard library (akin to CUDA's CUB)
- Shader linking across compilation units

"Shader languages seem to pride themselves on having bad tooling for modularity and libraries — this is a tragedy."

"Ease of reusing declarations and definitions across shaders. I.e. maintainability."

Theme 4: Language Modernity — Type System and Features

- Templates / generics (C++ style)
- Enums, structs with methods, operator overloading
- Type inference
- Sum types / match expressions
- Bounds checking options

"GLSL is not a great language in modern usage: doesn't have basic features a language should have in 2026."

"Modern programming language features, type system features such as type inference, enums, pattern matching, generics."

Theme 5: Performance Transparency

- Explicit control over register pressure
- Memory bound info / load-store transparency
- Low-level intrinsics for optimization

> The pointer and function call theme is so dominant that it warrants treating it as the primary language-level request from this survey. The underlying desire: stop artificially constraining what GPU hardware can actually do. CUDA is the reference developers cite — not because CUDA is perfect, but because it doesn't impose artificial limits on addressability and indirection.

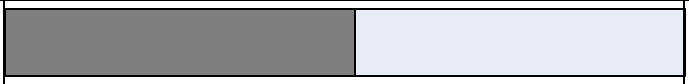
Q13: Value of Khronos Activities

Answered: 398 | Skipped: 6

Question: How valuable would you find the following Khronos activities? (1 = not valuable, 5 = extremely valuable)

Activity	Avg (of 5)	1→5 Distribution
Cross-vendor shader debugging standard	4.18	2% / 5% / 16% / 27% / 50%
Better docs & reference implementations	3.92	3% / 9% / 20% / 30% / 38%
Standardization of toolchain/tools	3.86	4% / 6% / 23% / 34% / 33%
SPIR-V evolution	3.80	5% / 8% / 23% / 31% / 33%
Education and best practices	3.75	5% / 10% / 24% / 26% / 34%
Conformance testing infrastructure	3.33	8% / 16% / 30% / 28% / 18%
Advancing Slang	3.18	19% / 15% / 19% / 21% / 26%
Material interchange standards	2.57	27% / 21% / 29% / 15% / 8%

Cross-vendor debug standard		83.6% (n=392)
Better docs & reference impl		78.4% (n=392)
Toolchain/tools standardize		77.2% (n=388)

SPIR-V evolution		76.0% (n=389)
Education & best practices		75.0% (n=394)
Conformance testing		66.6% (n=387)
Advancing Slang		63.6% (n=393)
Material interchange		51.4% (n=386)

Q13 — Khronos activity ratings (bars show weighted score as % of max; n=398)

The cross-vendor shader debugging standard scores 4.18/5 — with 50% of respondents rating it 5 (extremely valuable) and only 7% rating it 1 or 2. This is an exceptionally strong signal: not just plurality preference, but a genuine consensus mandate.

"Advancing Slang" scored 3.18 and shows the most polarized distribution — 19% rate it not valuable at all, while 26% rate it extremely valuable. This reflects the split in the community between those who see Slang as the way forward and those who are skeptical or committed to other languages.

Selected Open-Text Responses (Q13 "Other")

"Personally I would rather see more effort dedicated to hlsl-clang with first-class Vulkan SPIRV support."

"push for projects such as rustgpu, shady etc."

"linker, kernel and shader converging"

"Khronos' activities don't necessarily apply, SPIR-V is not a default target for us."

"slangc constantly segfaults or ICE if you try anything slightly tricky, feels like it was pushed to production too early"

"Advancing GLSL and putting effort into maintaining GLSLang"

"too much of slang's new commits are AI generated, this leads to very poor trust in the compiler"

Concerns about AI-generated commits in Slang appeared unprompted across multiple open-text responses (Q7, Q12, Q13, and Q16). This feedback suggests that even a small number of high-visibility issues can have a disproportionate impact on developer confidence, making it important to address perceptions of reliability alongside actual code quality.

Q14: Specifications Khronos Should Own or Incubate

Answered: 102 | Skipped: 302 (open text, optional)

Despite being optional, this question received 102 substantive responses. The theme distribution reveals clear camps within the community:

HLSL Standardization — The Pragmatist Camp

HLSL is mentioned in roughly 20 responses, making it the most-named specific language. The argument: HLSL has broad industry adoption across DX12 and Vulkan, is already de facto cross-platform, and Khronos legitimizing it would be more useful than building yet another new standard.

"HLSL support. Which I guess Slang kind of does, and spirv-cross kind of does. That's not enough though — there should be a proper HLSL spec."

"HLSL. Is perfect."

"Adopt HLSL."

"HLSL as a first class citizen"

"I would like to see a faster evolution of HLSL, rather than switch to Slang"

Slang Development — The Convergence Camp

Approximately 10 responses specifically mention Slang as what Khronos should own or focus on:

"Make Slang the compiler and Slang the language be the industry standard."

"Slang — don't lose focus — invest in nothing else"

"Enjoying slang. Any efforts to streamline and simplify will be appreciated."

VCC/Shady — The C++ for GPU Camp

VCC (Vulkan Clang Compiler / Shady IR) receives roughly 12 mentions, more than any project outside HLSL and Slang. This is notable given its relatively low public profile:

"Vulkan Clang Compiler (VCC). I just wanna use a real programming language for shaders, I don't want to learn new toy languages."

"vcc/shady give me c++ shaders you already have SYCL C++"

"Shady/VCC"

Rust-GPU — The Rust Ecosystem Camp

Rust-GPU appears in roughly 10 responses, often with a funding/staffing concern:

"I wish Khronos funded rust gpu more, it feels severely understaffed"

"Rust-GPU. IMO It's time for a reboot from the ground up"

SPIR-V Tooling and Infrastructure

"Not really. Get SPIR-V right, bootstrap the industry with tools to generate good SPIR-V, and everything else will follow."

"More love for GLSL"

“I can't think of an interesting capability that isn't in _some_ shader language or IR. The bigger problem IMHO is the fragmentation, especially across IRs. It would be great to have more vendors supporting SPIR-V, both for graphics and compute, to reduce fragmentation.”

"WebGPU to support SPIR-V"

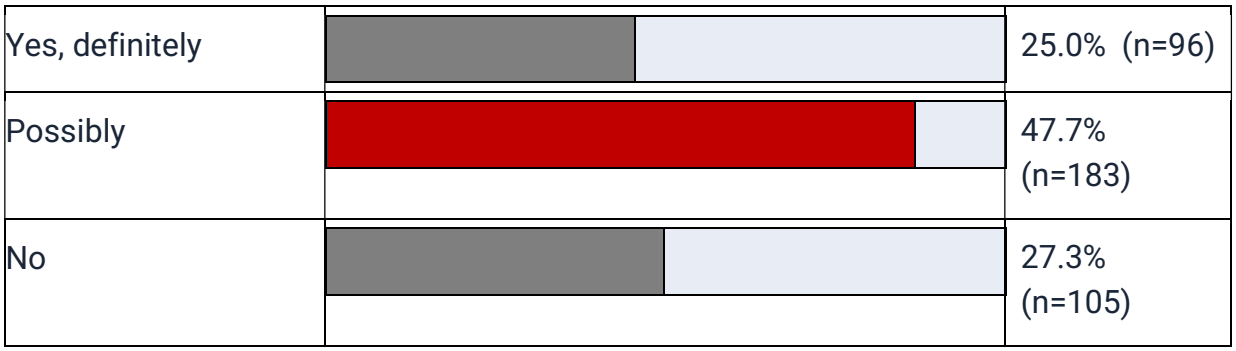
> The VCC/Shady mentions are a signal worth monitoring. A small but technically sophisticated population is converging on "C/C++ to SPIR-V" as the right abstraction, separate from both dedicated shading languages and high-level IRs. Hugo Devillers (identifiable from Q17) is the primary researcher behind this project, and the community is tracking his work closely.

Q15: Interest in Future Khronos Participation

Answered: 384 | Skipped: 20

Question: Would you be interested in participating in future Khronos working groups, review sessions, or early-access programs?

Response	Respondents	Percentage
Yes, definitely	96	25.00%
Possibly	183	47.66%
No	105	27.34%



Q15 — Interest in future Khronos participation (n=384)

Open to participation (yes + possibly)	73% (279 respondents)
Not interested	27% (105 respondents)
Definitely interested	25% (96 respondents)

73% expressing openness to participation represents a substantial pool of engaged developers. Of the 25% who said "definitely," many left contact information in Q17 (92 respondents), giving Khronos a direct outreach list of motivated community members.

The combination of high participation interest (73%) and the Q17 contact information responses creates an immediate opportunity: Khronos can stand up a real-time shading developer advisory group with willing, identified participants from this survey population. The infrastructure for community engagement exists — it needs to be activated.

Q16: Additional Comments and Suggestions

Answered: 83 | Skipped: 321 (open text, optional)

The final open-text question received 83 responses covering a range of perspectives. Major themes follow.

Apple and Metal Concerns

Apple and Metal come up repeatedly — more in this question than in any structured question, because no structured option captured it directly:

"Please force Apple to be interoperable by any means necessary. We don't need metal."

"Get Apple to abandon Metal"

"I like Slang, but is Khronos Group the best to develop it? I'd rather Khronos focus on getting Apple on board with SPIRV."

"Please continue integrating compute features from OpenCL into Vulkan and GLSL/Slang, and hopefully get Apple to adopt Vulkan at some point."

CPU-GPU Unification

Several responses articulate a vision of unified CPU/GPU programming:

"We need to seriously think about what we want the long-term GPU programming ecosystem to look like."

"Authoring shaders in a general-purposed programming language like cpp and rust."

"I think we need a new low level programming language that has built in reflection so we can use it on both CPU and GPU without needing to write a shader separately."

Slang Stability Concerns

"i appreciate slang as a language quite a lot but as of today it is way too unstable for production use"

"I stopped transition from GLSL to Slang. Slang needs to be a superset first."

"I wished WGSL moved faster, and/or Slang worked better for WebGPU and had a Rust API."

Positive Feedback

"I'm glad slang finally came along, I was hunting a decent C-style shading language for years."

"Good work everyone"

"Excited for the future"

"LLM advancements have made writing high-level languages and optimizing compilers much easier — we should see rapid improvement."

Survey Design Feedback

Two respondents provided meta-feedback on the survey itself:

"Whomever came up with this survey takes the AI slop badge. It definitely seems like it was generated with AI."

"In future surveys please add a 'I don't know / No opinion' option. Being indifferent shouldn't be registered the same as 'Neutral'."

> The survey design critiques are noted and will be addressed in future editions. The "No opinion" vs. "Neutral" distinction is a meaningful one in Likert-scale survey design and will be incorporated. The AI-generation accusation is taken as feedback on tone and question framing.

Appendix A: Survey Distribution Channels

The survey was distributed across the following channels between June 16 and July 10, 2026:

Khronos Newsletter

- June 16 Khronos Newsletter

Discord Servers

- Slang Discord Server (#general → moved to #announcements by moderators)
- Graphics Programming Discord Server — posted in #general, subsequently moved to #announcements by moderators;
- Khronos Discord Server (#lounge / community)
- Vulkan Discord Server

LinkedIn

- Original post by Khronos
- Original post by Randi Rost
- Repost by Takashi-san (amplified reach)
- Reminder post
- Repost by Randi Rost (secondary reminder)

Reddit

- r/GraphicsProgramming — accepted
- r/gamedev — accepted
- r/shaders — rejected (subreddit policy: no surveys)

Mastodon (Fosstodon — @thekhronosgroup)

- Initial post
- Reminder post

BlueSky (@khronos.org)

- Initial post
- Reminder post — tagged with #HLSL to specifically reach developers using HLSL who may not follow general graphics channels

Appendix B: Cross-Question Thematic Synthesis

The following cross-cuts emerge when the survey data is read holistically:

The Debugging Triangle

Three separate questions converge on shader debugging as the most urgent problem:

- Q7: Debugging complexity chosen by 53% as a top-3 pain point (#1 by a wide margin)
- Q8: Average debugging pain score of 52/100, with 40% scoring 61+
- Q9: Shader debugging cited by 56% as most needing standardization (#1)
- Q13: Cross-vendor debugging standard rated 4.18/5 (#1 Khronos activity value)

No other topic achieves top-ranked status across all four relevant questions. This is the clearest mandate in the survey.

Slang: Adoption vs. Trust

Slang appears across multiple questions with a consistent tension:

- Q3: 34% actively use Slang
- Q7 open text: Multiple mentions of compiler crashes, instability, AI-generated commits
- Q13: "Advancing Slang" scores 3.18/5 — lowest among structured Khronos activities, with the most polarized distribution
- Q14: Approximately 10 respondents explicitly want Khronos to prioritize Slang; approximately equal numbers express skepticism or opposition

Recommendation: Treat compiler reliability and trust-building as an immediate priority for Slang before further feature expansion. A public reliability dashboard or regular stability releases could help building confidence.

The Compute Developer Cohort

A coherent non-games compute population is present throughout:

- Q2: HPC, AI inference, simulation in "Other" write-ins
- Q4: Linux at 74% — far higher than a pure games sample would produce
- Q5: OpenCL, CUDA, LevelZero in "Other" write-ins
- Q12: CUDA cited repeatedly as the bar for addressability and function call support

This population experiences the "toy language" limitation most acutely. Tooling designed around rasterization pipelines frequently fails them. A dedicated compute developer track in future surveys and working group composition would improve representation.

The "Fewer Standards" Minority

A small but coherent camp explicitly argues against more standards:

"We need LESS standards, discontinuing SPIRV and WGSL could be a good start"

"None of what I can think of. SPIR-V is great. What sits on top is bad."

Their position: fragmentation is manageable; what's broken is implementation quality and tooling reliability. They are not opposed to Khronos activity — they want it redirected from language/IR proliferation toward making existing tools robust. This is a legitimate engineering position that deserves representation in planning discussions.

Apple as a Systemic Constraint

Apple's Metal ecosystem appears as a background frustration throughout the survey but is never captured by a structured question option. Future surveys should include explicit questions about:

- Cross-platform targeting including Apple platforms
- MoltenVK/KosmicKrisp vs. native Metal usage
- Impact of Apple's developer ecosystem on shader workflow choices