What game developers actually want from procedural level generation tools

Bojan Endrovski
Delft University of Technology
Delft, Netherlands
Breda University of Applied Sciences
Breda, Netherlands
b.endrovski@tudelft.nl

Joris Dormans
Ludomotion
Amsterdam, Netherlands
joris@ludomotion.com

Rafael Bidarra
Delft University of Technology
Delft, Netherlands
R.Bidarra@tudelft.nl

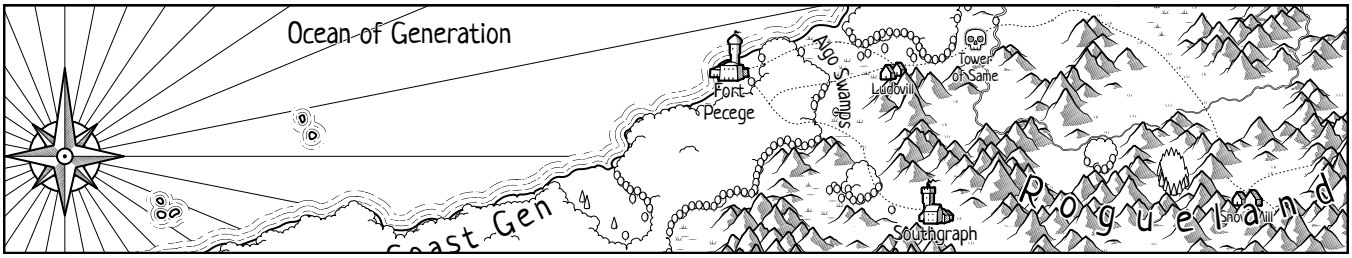


Figure 1: Procedural Map created with Perilous Shores [37]

Abstract

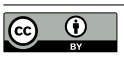
Academic research has produced numerous innovative PCG techniques, yet procedural level design remains unevenly adopted across the game development community. To understand why, we conducted a comprehensive survey of 120 game development professionals examining what developers actually experience when working with procedural tools. The survey covered current tool usage, adoption barriers, technical preferences, and future needs across level designers, game designers, technical artists, environment artists, programmers, and researchers. We present and analyze the survey responses using various techniques and visualizations. In addition, we developed an interactive online tool for anyone to explore patterns across demographic data segments and draw their own interpretations.

Two significant patterns emerge from the data. First, a statistically significant adoption gap exists between artists and designers. Artists use procedural generation far more frequently than designers. Second, multiple survey questions examining developer preferences for generative AI methods reveal consistent prioritization of creative control and process transparency over automation.

These findings point to an opportunity to expand PCG adoption by reducing technical barriers for designers while aligning tool design with observed practitioner priorities.

CCS Concepts

• **Applied computing** → **Computer games**; • **Human-centered computing** → *User centered design.*



FDG '26, Copenhagen, Denmark

ACM ISBN 979-8-4007-2495-4/26/08
https://doi.org/10.1145/3815598.3815682

Keywords

Games, Level Design, Procedural Generation, Survey, Analysis

ACM Reference Format:
Bojan Endrovski, Joris Dormans, and Rafael Bidarra. 2026. What game developers actually want from procedural level generation tools. In *Foundations of Digital Games (FDG '26), August 10–13, 2026, Copenhagen, Denmark.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3815598.3815682

1 Introduction

Tools shape their outputs. The first sharpened flint enabled toolmaking, precision-engineered silicon powers our digital age, and the tools available to game developers shape the games they create.

Academic research has produced a multitude of innovative procedural content generation (PCG) techniques. Yet, despite this progress, procedural level design remains unevenly adopted across the game development community. Procedural level generation has been a core feature of the roguelike genre since Rogue (1980) [10], but none of the top 20 best-selling roguelikes on Steam (February 2026) employ full procedural level geometry, and nearly half use no level generation at all (Appendix Table 4). This pattern is visible at the Everything Procedural Conference [5], the field's largest dedicated industry event: world building proposals far outnumber those addressing procedural level generation (Appendix Table 5).

This uneven adoption can be examined from multiple angles: market trends show what players buy, while tool design determines what developers can practically build. We focus on the tools, investigating how they align with the workflows of the people who would use them. What barriers prevent wider adoption? What capabilities do developers actually want? For this, we designed a survey that maps what game developers actually experience when working with procedural tools, with a particular focus on procedural level generation.

The outcome gives a vivid picture of what many game developers experience and think in relation to PCG usage. The goal is to inform the development of PCG tools that complement algorithmic innovation with observed developer needs. By understanding how procedural generation fits (or fails to fit) into real development workflows, we can identify opportunities to expand its practical application. We hope this research will help better understand the gap between PCG's proven academic potential [24] and its practical adoption in game development.

2 Related Work

Procedurally-generated content in games has become increasingly prominent in recent years, transforming how games are created. Recent titles like Returnal [21] and Hades [17] have brought procedural level generation into the spotlight, offering players truly unique experiences. However, such games remain exceptions.

Contemporary PCG workflows for world building provide numerous established options. Houdini [20] and Blender [3], with the addition of Geometry Nodes, offer robust modeling and simulation capabilities. The Unreal Engine [14] now integrates dedicated PCG tools. These tools primarily focus on offline generation workflows.

Some level generation and co-creation methods targeting specific types of content seem particularly attractive for level designers. For example, mixed-initiative tools allow designers to define key pieces of a level, with the system completing the remaining elements [27]. Identifying and understanding which qualities designers value in such tools is a very important question for both researchers and tool developers, but so far it has not received much attention [25].

PCG techniques have enabled many applications across game development [33]. They can create entirely unique experiences for every playthrough [30], represent vast worlds like entire galaxies [16], or adapt dynamically to players' behavior [28], enabling more accessible and personalized gaming experiences. Van der Linden et al. [35], for example, provide a comprehensive overview of PCG techniques for procedural dungeon generation. Among the most influential techniques of the last decade are Model Synthesis [29] and its derivative, Wave Function Collapse (WFC) [18], recognized for its ease of use and expressive range. Many variants of WFC have been explored, including extensions to support mixed-initiative, iterative design [23], and hierarchical approaches featuring semantically meaningful tile sets [1].

Graph grammars have been used to support higher-level gameplay concepts, including procedurally generated mission graphs [11], and even the harmonized generation of multiple game facets in a mixed-initiative setting [22], so-called game orchestration methods [26]. Dormans [12] formalized the level generation process in terms of model-based transformations, which materialized in the introduction of Ludoscope, a tool enabling visual creation of level generators. Ludoscope employs graph grammars and rule-based model transformations to generate mission graphs, which can then be translated into levels facilitating desired missions.

The PCG Workshop [38], which has been organized annually for the last 16 years, is a valuable resource offering many PCG proposals originating from academic research. However, despite their power and versatility, practical adoption of most of these techniques remains rather limited.

One of the challenges posed by many prototype procedural tools is that they require from their users a minimal level of "an algorithmic mindset", to understand their inner workings and be able to steer their output. Conversely, while generative AI technologies can lower technical barriers to content generation, they introduce concerns including intellectual property issues and reduced transparency. AI-driven approaches often function as black boxes [2, 39], potentially reducing designer control and creative agency. Balancing designer agency with automation capabilities remains a fundamental challenge in procedural level generation tool design.

Between academic research and commercial tools we find what we could call 'practitioner-driven PCG innovation', shared through open-source repositories and interactive demonstrations rather than through formal publication. Oskar Stålberg's Townscaper [34], and Oleg Dolya's generator suite [37] exemplify this approach. A good part of Kate Compton's work, including Tracery [8] and various Casual Creators [9], also falls into this category. These practitioners possess a rare combination: the technical skill to implement complex algorithms and the design sensibility to make them effectively understandable and controllable.

3 Research Methodology

We conducted a comprehensive industry survey to capture perspectives across different roles, experience levels, and development contexts. We collected 120 responses in our dataset.

The survey comprised 21 questions spanning demographics, current tool usage, technical preferences, and future needs. Questions employed multiple formats: single-choice, multiple-choice, matrix scales, ranking, and open-text. This multi-format approach captured both quantitative patterns and qualitative insights.

All questions were optional, allowing participants to skip items outside their expertise or not applicable to their work, prioritizing data quality over quantity. The complete survey, including question motivations, and response counts per question, is reported in Appendix Tables 11 and 12. All closed-ended questions included an 'other' option with free-text entry. We distributed the survey through targeted channels likely to reach developers familiar with PCG: the PCG mailing list, Discord communities of relevant conferences, and LinkedIn outreach via our institutional networks[1]. There was no prescreening; participation was fully anonymous and voluntary, with no personally identifiable information collected. We did not include attention checks, and while the average time was approximately 45 minutes, the median was 10 minutes, indicating most participants moved through the survey at a focused pace.

Following data collection, we developed an analysis tool enabling rapid visual querying and filtering of responses. This tool enabled us to identify patterns across demographic segments and correlations between preferences, usage patterns, and professional contexts. The complete dataset and analysis tools are publicly available [13].

To analyze open-ended responses, we performed inductive thematic coding following Braun and Clarke's approach [4]. An initial codebook of 11 candidate themes was generated using a large language model based on the survey's question structure and common PCG terminology, then manually refined through iterative review to

[1]We did not collect studio size data; our focus was on individual tool usage and workflow patterns.

arrive at a final codebook of 9 themes (Appendix Table 6). Responses could receive multiple codes when addressing distinct themes.

4 Results and Analysis

This section presents our survey findings through systematic analysis of each question, accompanied by appropriate visualizations. We structure our analysis chronologically, examining first the current state of PCG adoption (demographics and present usage), then exploring preferences and needs for future tools.

Throughout our analysis, we provide role-based breakdowns where meaningful differences emerge between professional categories. The colors and role abbreviations defined in Table 1 appear consistently in figure legends for clarity.

To maintain readability in charts, certain response options have been shortened or abbreviated. For brevity, some questions present condensed analysis with full visualizations provided in Appendix B. Complete question text and full multiple-choice responses appear in full in Appendix D.

Table 1: Professional Role Abbreviations and Color Coding

Professional Role	Abbreviation	Color
Level Designer	LD	
Game Designer	GD	
Technical Artist	TA	
Environment Artist	EA	
Programmer/Technical Designer	PTD	
Academic/Researcher	AR	
Other	O	

4.1 Demographics

The first three questions establish the demographic composition of our respondent pool. These demographics enable us to analyze how PCG tool usage, preferences, and concerns vary across different professional roles, experience levels, and technical environments. Understanding these patterns is essential for designing tools that serve diverse segments of the development community.

4.1.1 Professional Role Distribution. Professional roles vary significantly across the game industry, with procedural generation often bridging multiple disciplines [15]. This allowed analysis of how procedural generation needs vary by creators' specific responsibilities.

The survey reached a diverse audience with sufficient representation across all major game development roles (Figure 2). Where patterns emerge more clearly at a higher level, we group professional roles into broader categories: Artists (Technical Artists and Environment Artists) and Designers (Level Designers and Game Designers).

4.1.2 Experience Level Distribution. The data shows an even distribution across all experience levels, from newcomers to industry veterans (Figure 3). This balanced representation enables detection of experience-based trends that could influence tool development approaches. However, we observed minimal variation in responses

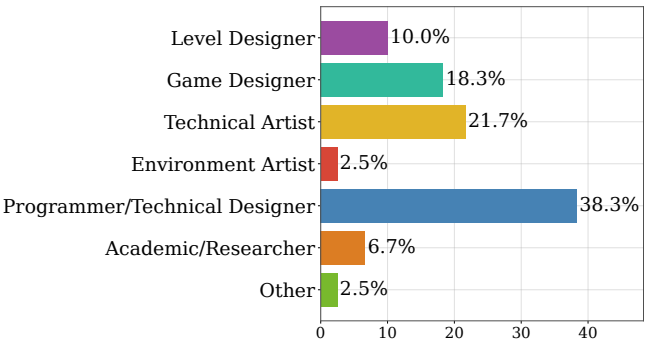


Figure 2: How would you primarily describe your professional role?

across the experience scale and will therefore sparsely use it as a filtering variable for the remainder of the analysis.

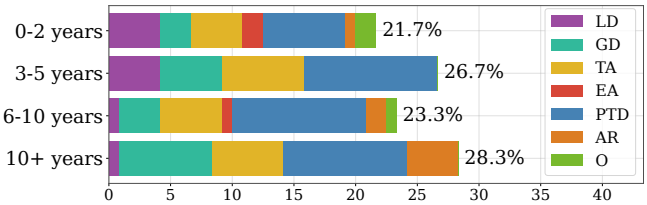


Figure 3: How many years of experience do you have in game development?

4.1.3 Game Engine Usage. Unreal Engine and Unity dominate our sample, with notable Godot representation (Appendix Figure 23).

4.2 The Present

We examine current PCG tool adoption across the industry: which tools developers use, their proficiency levels, what they generate with them, how frequently they incorporate procedural generation, and their primary concerns with existing solutions.

4.2.1 Procedural Tools Experience. Understanding which tools developers currently use and their proficiency levels shows adoption patterns, and identifies gaps and opportunities in the PCG ecosystem (Figure 4).

In addition, the experience landscape varies considerably by role. Among Game Designers and Level Designers, custom code-based solutions dominate, with 75% having some experience and 35% extensive experience. Technical and Environment Artists present a contrasting picture, with 55% having extensive Houdini experience, aligning with their focus on world building rather than level generation, a distinction we explore in the following question. Despite being introduced only recently, Unreal Engine's PCG framework has achieved notable adoption.

4.2.2 Current PCG Applications. Understanding where PCG is well-utilized and underutilized reveals research opportunities and gaps in current tooling. World-building and level layout emerge as the

dominant PCG applications (Figure 5). The prevalence of world-building is unsurprising given both its widespread necessity and the maturity of available tools.

More intriguingly, tasks do not neatly match traditional roles. Programmers/Technical Designers and Technical Artists are involved in many generation aspects traditionally associated with design roles. This pattern is explored in detail in Mind the Gap (Subsection 5.1).

4.2.3 Procedural Level Generation Frequency. This core survey question directly measures the adoption gap our research addresses. Understanding current usage patterns is essential for designing pathways toward more widespread PCG adoption in level design workflows.

A stark contrast emerges between artist and designer roles in their PCG adoption patterns (Figure 6). Artists (Technical Artists and Environment Artists) demonstrate substantially higher PCG adoption than designers (Level Designers and Game Designers), with artists frequently incorporating procedural generation while most designers use it rarely or never. Notably, no designer considered PCG essential to their process.

This validates the adoption gap and pinpoints a significant opportunity to bridge the divide between design intent and procedural implementation, examined in detail in Mind the Gap (Subsection 5.1).

4.2.4 Primary Concerns with Procedural Level Generation. These concerns guide future research directions and highlight challenges that tools and methodologies should address. Concerns are relatively evenly distributed (Figure 7), with lack of precise artistic control emerging as the primary barrier, followed by doubts about whether the benefits justify the setup time. Game designers in particular noted debugging unexpected outputs and unpredictable results as stronger concerns. Notably, most "other" responses pointed toward the "procedural oatmeal" problem [7], a lack of meaningful variety in generated content. In retrospect, this should have been included as a standard survey option.

4.2.5 Views on Current PCG Tools. To understand how current PCG tools meet developer needs, we asked respondents about their perceptions of existing solutions. The responses reveal considerable room for improvement (Figure 8). The most frequently cited concern is insufficient control over final output, closely followed by tools being too complex to integrate into workflows and tools

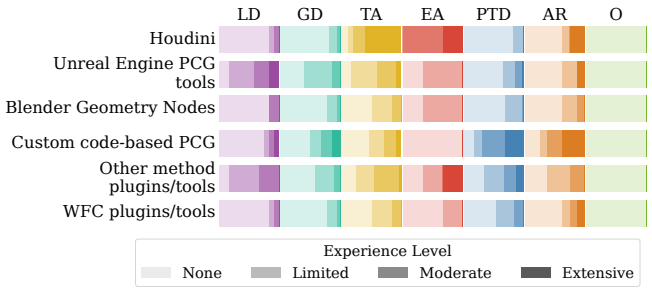


Figure 4: How would you rate your current experience with the following procedural tools?

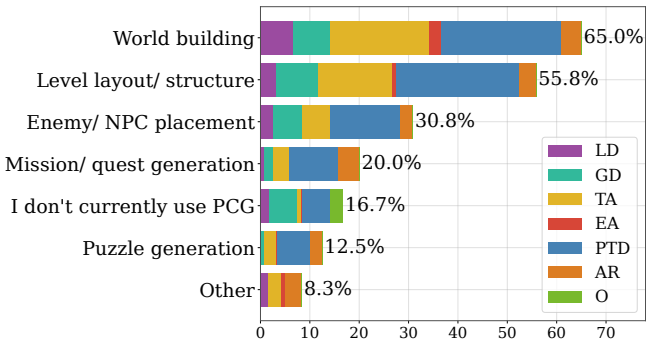


Figure 5: If you use procedural generation, what aspects do you currently use it for? (Select all that apply)

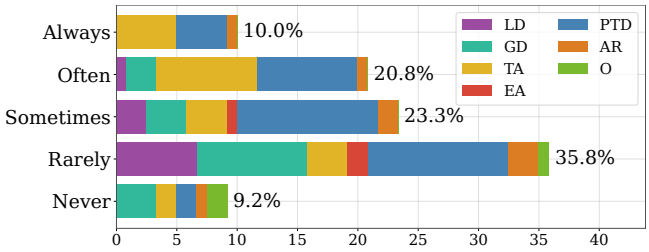


Figure 6: How frequently do you incorporate procedural level generation (not just world building) in your design workflow?

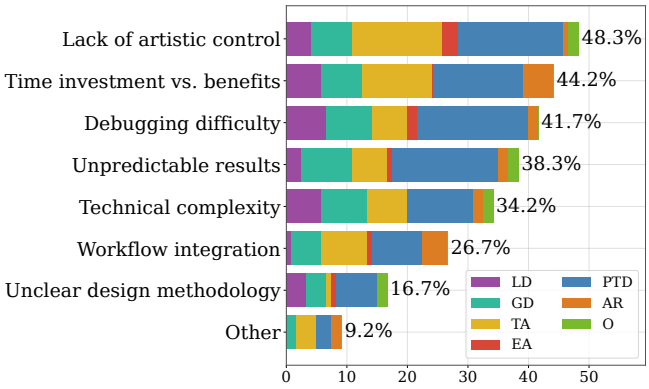


Figure 7: What are your primary concerns when considering procedural level generation? (Select up to 3)

being built for programmers rather than designers. Only a small minority expressed satisfaction with available tools. The distribution suggests PCG tools face multiple simultaneous barriers spanning control, complexity, and accessibility, rather than one critical flaw.

Notably, the "other" category captured more responses (Appendix Table 13) than any single predefined option, indicating our predefined options did not fully capture the specific challenges developers experience with current tooling.

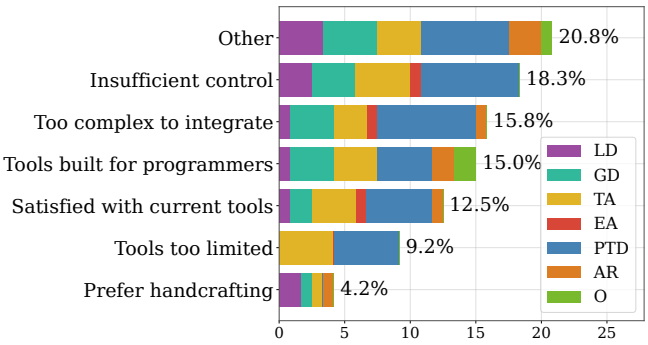


Figure 8: Which statement best describes your view on procedural level generation tools?

4.3 The Future

We explore developer preferences for future tools: what evaluation criteria matter most, which features they prioritize, how they want tools integrated into their workflows, what role they envision for AI, and which problems they most want solved.

4.3.1 Critical Factors for Tool Evaluation. Understanding what convinces creatives to adopt PCG and invest in new tools reveals the threshold requirements for successful tool design.

Flexibility emerges as the top priority, followed by documentation and learning resources (Figure 9). Simplicity and reliability also rank highly. Notably, while integration with existing workflows was preferred by most participants in Subsection 4.3.5, only a third considered it crucial.

Experience level reveals interesting patterns: seasoned developers prioritize reliability over documentation and simplicity, while junior participants value learning support more heavily. Surprisingly, familiarity ranks low across all groups, suggesting developers are willing to learn new paradigms when benefits justify the investment.

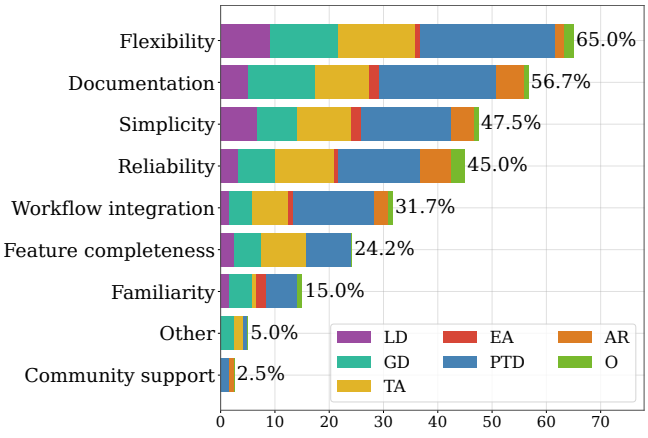


Figure 9: What do you consider the most critical factor when evaluating a new design tool? (Select up to 3)

4.3.2 Node-Based Tool Feature Priorities. Given the dominance of node-based interfaces in DCC and game engines, we expected this to be the preferred approach for PCG tools.

Control over generation constraints and rules emerges as the most crucial feature, echoing concerns raised about current PCG tools (Figure 10). The ability to mix handcrafted and procedural content follows closely, though this proves slightly less critical for game designers and level designers. Visual previews and debugging capabilities rank highly across all roles, pointing to the importance of transparency in the generation process.

The consistent emphasis on control and debugging validates that successful PCG tools must prioritize designer agency and process transparency over automation convenience.

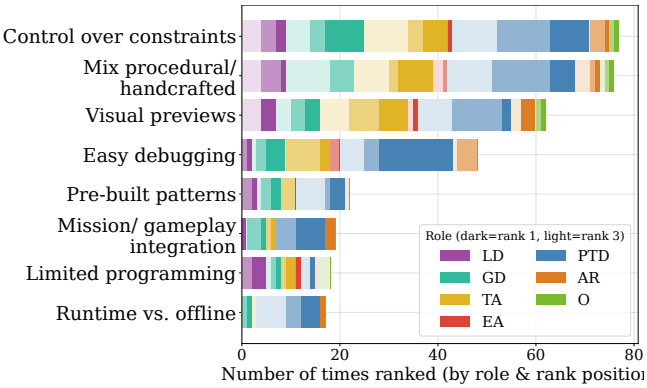


Figure 10: If using a node-based tool for creating a procedural level generator, which features would be most important to you? (Rank top 3)

4.3.3 Real-Time Feedback Importance. Two-thirds of participants rate real-time feedback as essential or very important for PCG tools (Appendix Figure 24), suggesting successful tools must move beyond "generate and test" workflows toward interactive, responsive design environments.

4.3.4 Preferred Generator Creation Approach. Different philosophies exist for creating procedural generation tools, each balancing flexibility against complexity. Understanding developer preferences helps determine the optimal approach for tool design.

The overwhelming majority prefer building generators through assembling pre-built configurable components or using programming primitives, favoring flexibility over simplicity (Figure 11).

This finding appears paradoxical given that steep learning curves and time investment ranked as major concerns in Subsection 4.2.4, while Subsection 4.2.5 validated that tools are built for programmers rather than designers. The apparent contradiction suggests that while developers want powerful, flexible tools, current implementations fail to make complexity manageable. This opens an intriguing research direction: how can we provide programming-level flexibility while maintaining designer accessibility?

4.3.5 Integration Preference for PCG Tools. Developers strongly prefer deep engine integration over standalone tools (Appendix Figure 25). Interestingly, while Subsection 4.3.1 showed integration

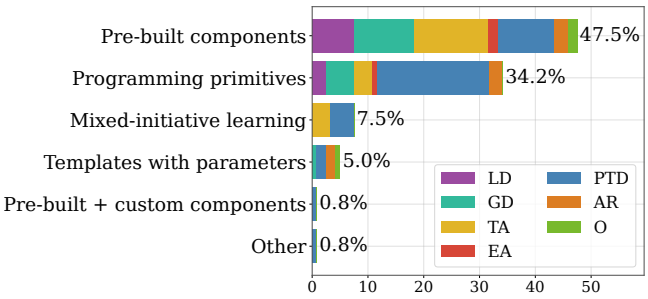


Figure 11: Which approach to creating procedural generators would you prefer?

ranked lower in general tool evaluation, developers strongly prefer it specifically for PCG implementation, suggesting procedural generation's complexity and iterative nature demands tighter workflow coupling than other tools.

4.3.6 Genre Interest for Procedural Level Generation. Roguelikes show highest interest as expected, while strong interest in action/adventure and platformers reveals untapped potential for genre-specific PCG approaches (Appendix Figure 26).

4.3.7 Level Representation and Storage. This technical question provides crucial guidance for determining which generation primitives our tools should prioritize. Understanding dominant representation formats directly impacts technical architecture decisions and algorithm selection.

Rectangular grids dominate level representation (Figure 12), likely due to their simplicity, intuitive nature, and processing efficiency. Cross-referencing with Subsection 4.2.2 reveals representation patterns by application: free-form geometry proves most prevalent for world building and remains the preferred choice among artists.

Node-based graphs, capable of representing abstract level design concepts, appear significantly underutilized despite their expressive power. This suggests an opportunity for tools that better leverage graph-based approaches for higher-level design concepts (keys, locks, challenges, etc.)

4.3.8 Level Generation Approach Preferences. Two fundamental design philosophies exist: generating spatial layouts and fitting gameplay to them, versus designing key gameplay moments first and generating space to facilitate those experiences. Understanding preferences helps determine whether tools should favor one approach or remain agnostic.

Rather than favoring either extreme, most participants prefer hybrid approaches balancing spatial and mission-driven generation (Figure 13). Context-dependent and balanced approaches dominate, suggesting developers want flexibility based on project needs. Successful PCG tools should therefore support both philosophies seamlessly or remain agnostic, allowing designers to blend spatial and gameplay-driven methods as their projects require.

4.3.9 AI Role Preferences in PCG Workflows. This is the first of three questions exploring how level and game designers envision AI's role in an increasingly AI-prominent future. Understanding

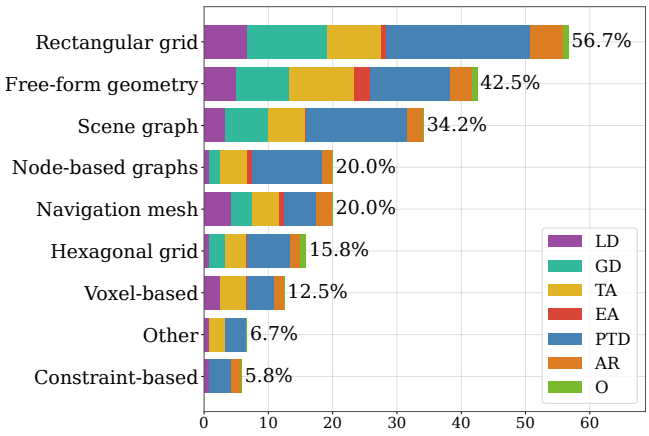


Figure 12: How are levels typically represented and stored in your projects, especially when procedurally generated? (Select all that apply)

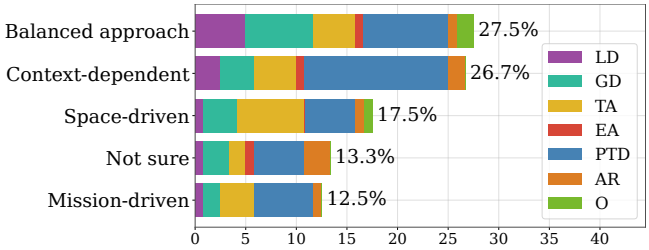


Figure 13: Which of these approaches to level generation would be most useful to your workflow? (Select one)

current attitudes helps predict adoption patterns for AI-assisted PCG tools.

Developers prefer AI as an assistant that either helps implement design intentions or enhances specific workflow components (Figure 14). A substantial portion still favors traditional rule-based generation without AI involvement, primarily technical designers and programmers.

Suggestion-based approaches significantly outperform full automation, reflecting designers' desire to maintain creative control. Notably, learning-based and analysis-based tools both scored poorly, suggesting skepticism about AI's ability to understand design intent from existing content.

4.3.10 AI-Assisted PCG Priority Factors. This question targets specific focus points for AI-powered tool development, helping identify which capabilities should be prioritized when designing AI-assisted PCG systems. Maintaining creative control over the final output emerges as the paramount concern, followed by consistency with existing game assets and style (Figure 15). Understanding AI decision-making processes ranks as important to approximately a quarter of participants.

These findings reinforce previous themes: designers demand granular control and transparency in generation processes, even with AI assistance. The emphasis on asset consistency suggests AI tools must integrate with existing pipelines.

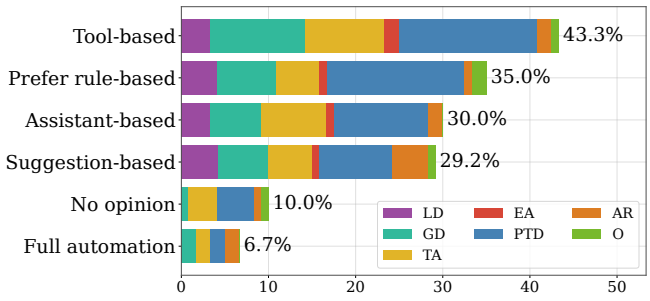


Figure 14: What role would you prefer AI to play in your procedural level generation workflow? (Select up to 2)

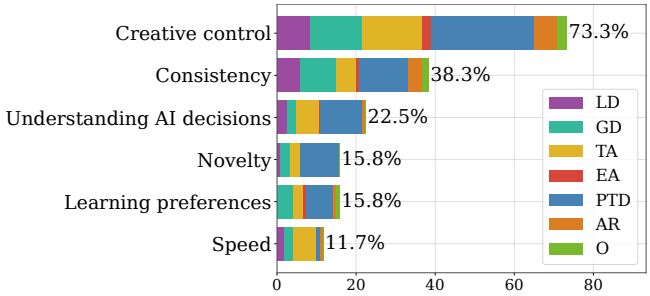


Figure 15: When considering AI-assisted procedural level design, which is most important to you? (Select up to 2)

4.3.11 AI Integration Concerns. Multiple factors could prevent designers from adopting AI as part of their procedural generation toolkit. Understanding these perceived obstacles helps address barriers to AI-assisted PCG adoption. The results reinforce findings from previous AI questions: unpredictable or inconsistent results, loss of designer agency, and lack of transparency emerge as primary concerns (Figure 16). Additionally, intellectual property and ownership issues surface as significant ethical considerations.

These concerns about predictability, control, transparency, and IP ownership suggest that successful AI-assisted PCG tools must prioritize deterministic behavior, designer oversight mechanisms, explainable processes, and clear intellectual property frameworks. The consistency across AI-related questions validates these as fundamental requirements for professional adoption.

4.3.12 Desired Solutions. Identifies the most pressing problems that PCG tools should address.

Creating more content variations with consistent quality emerges as the top priority, followed by reduced technical barriers and better workflow integration (Figure 17). Time savings and improved iteration speed also rank highly, while community marketplaces and learning resources show moderate interest.

4.3.13 Most Important Problem. This open-ended question provides a more free-form complement to the previous closed-ended question, allowing participants to express their most critical concerns in their own words. As described in Section 3 we used thematic coding to analyze the answers of this question.

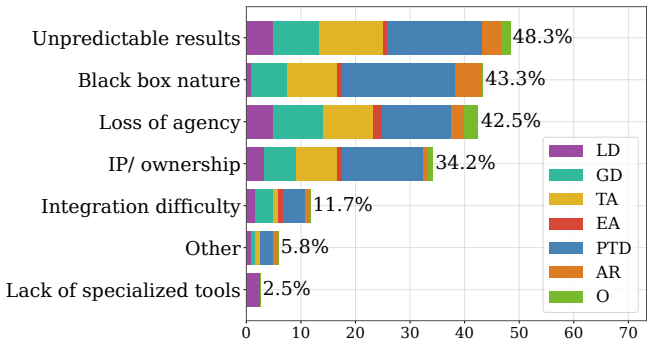


Figure 16: What concerns you most about using AI in procedural level generation? (Select up to 2)

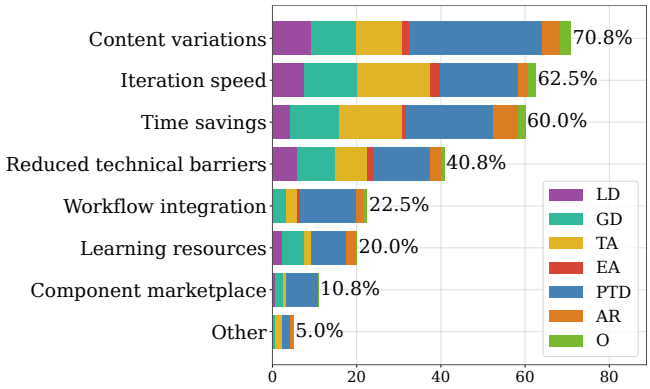


Figure 17: Which problems do you wish a procedural level generation tool could solve for you? (Select all that apply)

Time & Efficiency emerged as the dominant theme (n=28, 51.9%), followed by *Quality & Consistency* (n=14, 25.9%), and *Control & Flexibility* (n=10, 18.5%). For visualization, we chose a co-occurrence network (Figure 18), where the size of the nodes represents the occurrence of the theme and the thickness and saturation of the edges correspond to the co-occurrence. The network shows these concerns are deeply interconnected, with *Quality & Consistency* and *Time & Efficiency* being connected to 7 other themes each, while *Control & Flexibility* co-occurred in 6 responses, suggesting participants view these as interrelated challenges.

This pattern indicates the primary barrier to PCG adoption is not a single isolated problem but rather the tension between speed, reliability and flexibility. The prominence of *Control & Flexibility* further emphasizes that efficiency gains cannot come at the cost of designer agency.

5 Discussion

Our analysis tool enabled rapid examination of the response data, revealing patterns in how different professional roles approach procedural generation. The quantitative data establishes usage patterns, preferences, and desired capabilities. Thematic coding of

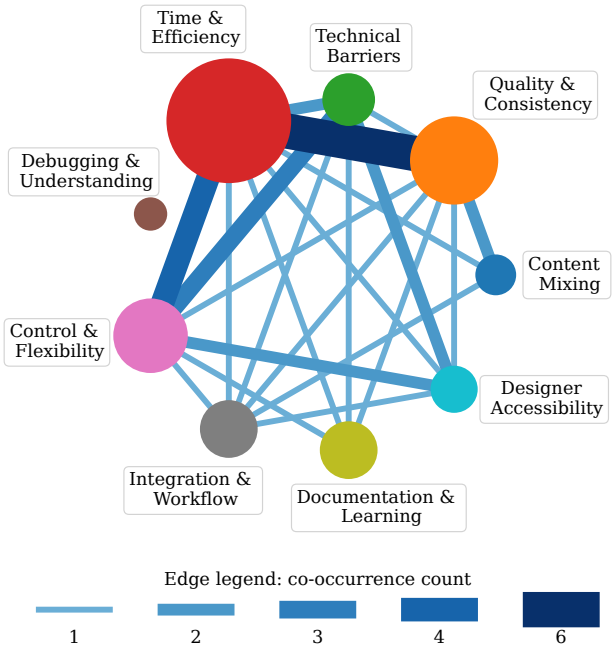


Figure 18: What is the single most important problem you would want a procedural level generation tool to solve in your workflow?

open-ended responses reinforces these findings, showing how concerns cluster around interconnected challenges rather than isolated issues.

Several patterns emerge that point toward opportunities for better alignment between current tools and developer needs. These patterns reveal structural factors in how procedural generation integrates into development workflows, and where integration remains difficult. We organize these findings into three key observations: an adoption gap across professional roles, a consistent preference for creative control over automation, and opportunity for closer alignment between academic research and industry concerns.

5.1 Mind the Gap

Our data reveals that level designers and game designers adopt PCG significantly less than other production roles in the game industry. This pattern holds whether we compare adoption rates directly against artists, or look at who actually performs design-oriented PCG tasks in practice.

To quantify this gap, we focused on the answers in Subsection 4.2.3. Without loss of generality, we treated the five response categories as numerically equally spaced, a common approach in survey analysis that allows for meaningful statistical comparison [19]. Using a scale from 0 (Never) to 1 (Always), we assigned equally-spaced numerical values to each response category. Professional roles were clustered into broader categories: Artists (Environment Artists and Technical Artists) and Designers (Level Designers and Game Designers).

Using this numerical weighting and role categorization, we calculate mean adoption scores by summing the weighted responses

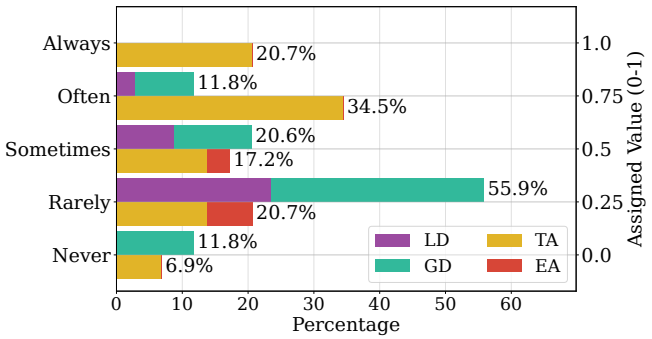


Figure 19: Procedural level generation frequency by professional role category

Table 2: PCG Adoption Analysis by Professional Category

		Artists (n=30)		Designers (n=34)	
Response	**Scale**	**Count**	**Value**	**Count**	**Value**
Always	1.00	6	6.00	0	0.00
Often	0.75	10	7.50	4	3.00
Sometimes	0.50	5	2.50	7	3.50
Rarely	0.25	6	1.50	19	4.75
Never	0.00	3	0.00	4	0.00
Total		**17.50**		**11.25**	
Mean Score		**0.58**		**0.33**	

Statistical Analysis
Mean difference: 0.25 **(76% higher)**
Mann-Whitney U test: p = **0.001**

for each group and dividing by group size. This produces a score between 0 and 1 for each category, with the difference quantifying the adoption gap. A Mann-Whitney U test confirms this difference is statistically significant (p = 0.001) [36], indicating that the observed gap between artists and designers is highly unlikely to have occurred by chance. The comparison reveals strong differences in PCG integration patterns (Table 2).

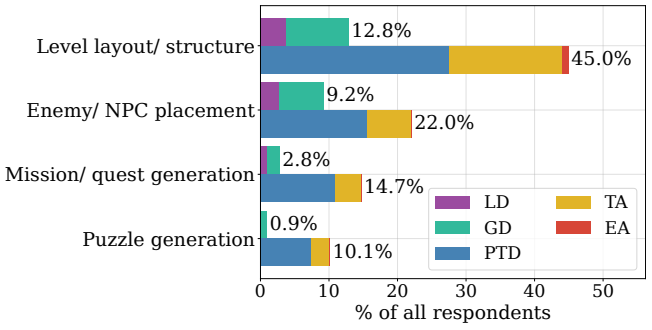


Figure 20: Role composition across design-focused procedural generation tasks.

We can also examine this pattern from a different perspective. We extracted the design-oriented tasks from Subsection 4.2.2, specifically Level layout/structure generation, Enemy/NPC placement, Mission/quest generation, and Puzzle generation. For each task, we investigated who performed it by using two role categories: Design Roles ● (Level Designers and Game Designers) and Technical/Art Roles ● (Environment Artists, Technical Artists, Programmers/Technical Designers).

The results reinforce a fundamental mismatch in the current PCG landscape. Level designers and game designers, the roles these tasks traditionally belong to, are underusing procedural generation. Meanwhile, technically proficient artists and programmers are compensating by expanding into design territory. The need for procedural generation in production is there; it is just being fulfilled by non-designers.

5.2 The Creator's Seat

When it comes to creative work, developers are willing to let AI ride shotgun, but they insist on staying in the driver's seat. This pattern emerges consistently across questions about automation, AI assistance, and tool design, pointing to a clear preference for creative ownership.

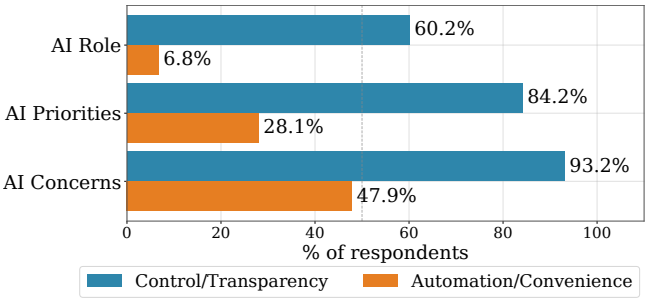


Figure 21: Control vs Automation in AI tools

Table 3: Categorization of survey responses

Control/Transparency	Automation/Convenience
AI Role Preferences in PCG Workflows	
Assistant-based	Full automation
Tool-based	Learning-based
AI-Assisted PCG Priority Factors	
Creative control	Speed
Decisions Transparency	Novelty
AI Integration Concerns	
Unpredictable results	Performance requirements
Loss of agency	Integration difficulty
Black box nature	Lack of specialized tools
	Copyright/IP issues

Developers favor assistant-based and tool-enhancement roles (Subsection 4.3.9) over autonomous generation, with learning-based approaches, where AI operates most independently, scoring much lower. When prioritizing AI capabilities (Subsection 4.3.10), maintaining creative control dominates, followed by asset consistency and understanding AI decision-making. The concern aspect (Subsection 4.3.11) shows similar patterns: unpredictable results, loss of agency, and lack of transparency emerge as primary barriers, alongside IP ownership concerns unique to AI-generated content. These consistent choices show more than user preference, they expose a mismatch between how AI tools are often positioned (as fully autonomous creative assistants) and what creators actually want (controllable tools that amplify their capabilities). Developers chose creative work to own their creative decisions.

The implications for PCG tool development are clear. Successful tools should be built to amplify designer agency rather than black-box automation [2]. AI can suggest, accelerate, and fill in details while leaving the creative decisions in human hands.

5.3 Adjacent Symphonies

Academic research and industry challenges do not always align. To explore this relationship, we constructed a rough approximation of academic focus areas and compared them to the developer concerns captured in our survey. We selected the PCG Workshop [38] as our reference corpus: the longest-running workshop dedicated to procedural content generation in games, with 152 openly accessible papers spanning 16 years.

The PCG Workshop covers broader topics than level generation alone, including storytelling, music, and game rules. We filtered papers focusing on level generation, narrowing to 59 papers. Using term frequency-inverse document frequency (TF-IDF)[32] on titles and abstracts, we extracted the 200 highest-scoring terms (Appendix Table 10). We integrated these terms into our thematic coding matrix from Subsection 4.3.13, adding two themes (Algorithms & Models, and Tooling & Methods) and expanding existing categories. This comparison suggests that academic research concentrates on methods, algorithms, and models, while industry concerns distribute across control, efficiency, integration, and accessibility (Figure 22).

This does not argue that academic research should serve industry needs. Curiosity-driven research remains essential for fundamental advances [6]. Rather, it identifies substantial opportunity: the problems developers face daily represent rich research territory with immediate practical impact.

5.4 Limitations

5.4.1 Survey Design and Interpretation. We designed survey questions and response options to capture a neutral picture of PCG adoption challenges. We employed statistical methods for textual analysis (Subsection 5.1), but human language is malleable; interpretation and meaning vary between individuals. We acknowledge that we introduced our own interpretive bias during question formulation and data categorization. We designed this study with open data as a core principle. The complete dataset, including an interactive analysis tool, enables independent researchers to explore alternative interpretations and derive their own insights [31].

5.4.2 Role Categorization. Combining Programmers and Technical Designers into a single role category complicated our analysis. We made this decision to focus on professionals directly involved in level creation, anticipating programmers would be a minority

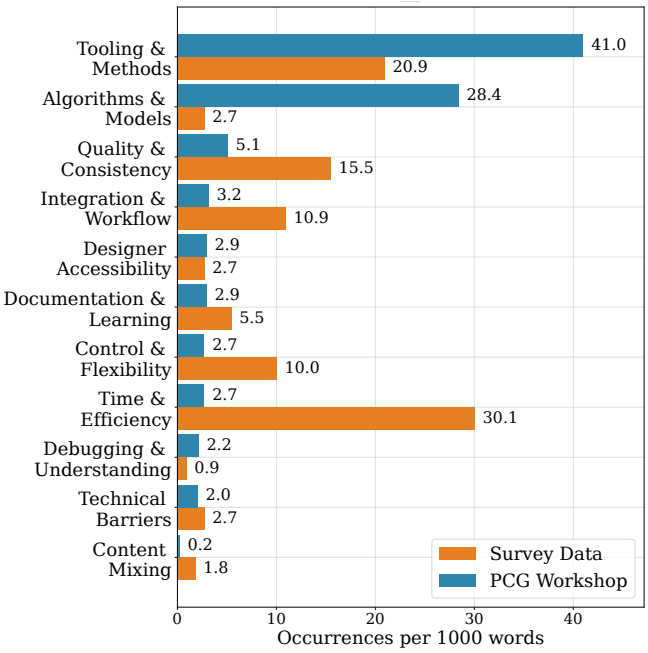


Figure 22: Theme occurrence comparison between PCG Workshop papers and industry survey responses, normalized per 1000 words

among our respondents. Studio-to-studio variation in how these roles are defined supported the consolidation. In retrospect, this obscured meaningful distinctions between technical implementation (programming) and technical design (systems and rules). The data suggests separating these roles would improve our understanding of how technical versus creative orientations shape PCG tool preferences. Future surveys should treat these as distinct categories.

5.4.3 Academic Comparison Scope. The comparison in Section 5.3 is limited in scope. On the industry side, we use responses from a single open-ended question; on the academic side, a single venue stands in for the broader research landscape. Broadening to multiple questions, venues, or full paper bodies would sharpen the picture, but remains outside the scope of this study.

5.4.4 AI Terminology. Subsections 4.3.9, 4.3.10, and 4.3.11 used *AI* as an umbrella term, reflecting how the word is commonly used among game developers to refer to generative AI and LLMs. This framing implicitly sets aside the fact that many established PCG techniques also fall under *AI*.

5.5 Future Considerations

The data presented in this paper clearly highlights a discrepancy in the usage of PCG tools between different industry roles, but the exact reasons behind this pattern remain somewhat elusive. Despite the data we collected, we can only speculate about these reasons on the basis of our interpretation of the data and our own experience working in and with the game industry. However, we feel that these speculations might inspire and direct future research.

One difference we see is that for technical artists the current PCG tools align much better with their professional ambitions. Technical artists are always pushing the envelope of the current technical affordances in regard to artistic expression; whether this is about increasing realism by creating evermore detailed models and animations, or by using advanced techniques to create novel, spectacularly stylized effects. Technical tools have always been an integral aspect of these developments, so it is not surprising that professionals in these roles experience fewer barriers to employ PCG tools.

In contrast, the core responsibility of game designers is to create pleasurable experiences for the player. For that, they traditionally rely on personal intuition, artistic vision, and a lot of play testing. Aspects of game development where technical tools are mostly not integral. Therefore, game designers have far less compelling reasons to employ PCG tools and techniques to realize their professional goals.

Following this line of reasoning, the real barrier of entry for most game designers is not technical. Rather, PCG tools should more clearly indicate what new opportunities they unlock. For better or worse, PCG impacts the way players experience games. Only when game designers incorporate the opportunities that PCG brings into their professional ambition, it becomes sensible to even try to navigate the relatively high barriers of entry that are associated with PCG tools. We suggest that one possible line of future research investigates this assumption, either by trying to prove or disprove the assumption in a future survey, or by tailoring PCG tools to foreground the beneficial impact they might have on future games.

5.6 Conclusion

Our survey among 120 game development professionals reveals two distinct patterns in PCG tool adoption. First, artists adopt procedural generation far more than designers (mean usage scores: 0.58 versus 0.33, p=0.001), leaving designers missing opportunities to unlock new workflows and creative possibilities. Technical roles compensate by expanding into design territory: around 60% of level layout generation in our sample is performed by non-designers.

Second, examining preferences across generative AI questions reveals consistent prioritization of creative control and transparency over automation. When evaluating AI capabilities, 84% emphasize control and transparency factors. When selecting AI-assisted tool features, 73% demand creative control over their final output. When identifying concerns about AI adoption, 93% cite control-related issues: unpredictable results, loss of agency, and lack of transparency.

These findings point toward a clear design principle: successful PCG tools should amplify designer agency rather than replace it. Current technical barriers do not stimulate designers to adopt procedural generation. Yet, most developers favor tools enhancing their capabilities. We can therefore expect more research effort to focus on this promising direction, as bridging this gap will empower designers to achieve the creative workflows they envision.

References

[1] Shaad Alaka and Rafael Bidarra. 2023. Hierarchical Semantic Wave Function Collapse. In *Proceedings of the 18th International Conference on the Foundations of Digital Games (FDG '23)*. Association for Computing Machinery, New York, NY, USA, 1–10. doi:10.1145/3582437.3587209

[2] Aleksandre Asatiani, P. Malo, Per Rådberg Nagbøl, Esko Penttinen, Tapani Rinta-Kahila, and Antti Salovaara. 2020. Challenges of Explaining the Behavior of Black-Box AI Systems. *MIS Q. Executive* 19 (2020). doi:10.17705/2MSQE.00037

[3] Blender. 2025. Geometry Nodes - Blender 4.3 Manual. https://docs.blender.org/manual/en/latest/modeling/geometry_nodes/index.html

[4] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology* 3, 2 (Jan. 2006), 77–101. doi:10.1191/1478088706qp063oa Publisher: Routledge _eprint: https://doi.org/10.1191/1478088706qp063oa.

[5] BUAS, Bojan Endrovski, and Ronny Franken. 2025. Everything Procedural Conference. https://www.everythingprocedural.com/

[6] Vannevar Bush. 1945. *Science, the Endless Frontier*. US Government Printing Office, Washington, DC.

[7] Kate Compton. 2016. So you want to build a generator... https://proceduralgeneration.tumblr.com/post/139979646183/so-you-want-to-build-a-generator

[8] Kate Compton, Ben Kybartas, and Michael Mateas. 2015. Tracery: An Author-Focused Generative Text Tool. In *Interactive Storytelling*, Henrik Schoenau-Fog, Luis Emilio Bruni, Sandy Louchart, and Sarune Baceviciute (Eds.). Springer International Publishing, Cham, 154–161.

[9] Kate Compton and Michael Mateas. 2015. Casual Creators. In *Proceedings of the Sixth International Conference on Computational Creativity (ICCC 2015)*, Hannu Toivonen, Simon Colton, Michael Cook, and Dan Ventura (Eds.). Brigham Young University, Park City, Utah, 228–235. http://computationalcreativity.net/iccc2015/proceedings/10_2Compton.pdf

[10] Wikipedia contributors. 2026. Rogue (1980). https://en.wikipedia.org/w/index.php?title=Rogue_(video_game)&oldid=1337873899 Page Version ID: 1337873899.

[11] Joris Dormans. 2010. Adventures in level design: generating missions and spaces for action adventure games. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games (PCGames '10)*. Association for Computing Machinery, New York, NY, USA, 1–8. doi:10.1145/1814256.1814257

[12] Joris Dormans and Sander Bakkes. 2011. Generating Missions and Spaces for Adaptable Play Experiences. *IEEE Transactions on Computational Intelligence and AI in Games* 3, 3 (Sept. 2011), 216–228. doi:10.1109/TCIAIG.2011.2149523 Conference Name: IEEE Transactions on Computational Intelligence and AI in Games.

[13] Bojan Endrovski, Rafael Bidarra, and Joris Dormans. 2025. Procedural Level Generation Survey. https://enci.github.io/plg-survey/

[14] Epic. 2025. Procedural Content Generation Overview | Unreal Engine 5.5. https://dev.epicgames.com/documentation/en-us/unreal-engine/procedural-content-generation-overview

[15] Gamebadges Project. 2025. Gamebadges Competence Map. https://map.gamebadges.eu/

[16] Hello Games. 2025. No Man's Sky - No Man's Sky. https://www.nomanssky.com/

[17] Supergiant Games. 2025. Hades on Steam. https://store.steampowered.com/app/1145360/Hades/

[18] Maxim Gumin. 2016. Wave Function Collapse Algorithm. https://github.com/mxgmn/WaveFunctionCollapse original-date: 2016-09-30T11:53:17Z.

[19] Tyler Hamby and Daniel S. Levine. 2016. Response-Scale Formats and Psychological Distances Between Categories. *Applied Psychological Measurement* 40, 1 (Jan. 2016), 73–75. doi:10.1177/0146621615597961

[20] SideFX Houdini. 2025. Houdini - 3D modeling, animation, VFX, look development, lighting and rendering | SideFX. https://www.sidefx.com/

[21] Housemarque. 2025. Returnal™ on Steam. https://store.steampowered.com/app/1649240/Returnal/

[22] Daniël Karavolos, Anders Bouwer, and Rafael Bidarra. 2015. Mixed-Initiative Design of Game Levels: Integrating Mission and Space into Level Generation. (2015).

[23] Thijmen S. L. Langendam and Rafael Bidarra. 2022. miWFC - Designer empowerment through mixed-initiative Wave Function Collapse. In *In Proceedings of FDG*. ACM, ACM Press, 66:1–66:8. doi:10.1145/3555858.3563266

[24] Antonios Liapis. 2020. 10 Years of the PCG workshop: Past and Future Trends. In *International Conference on the Foundations of Digital Games*. 1–10. doi:10.1145/3402942.3409598 arXiv:2104.11037 [cs].

[25] Antonios Liapis, Gillian Smith, and Noor Shaker. 2016. Mixed-initiative Content Creation. In *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*, Noor Shaker, Julian Togelius, and Mark J. Nelson (Eds.). Springer, 195–214.

[26] Antonios Liapis, Georgios N. Yannakakis, Mark J. Nelson, Mike Preuss, and Rafael Bidarra. 2019. Orchestrating Game Generation. *IEEE Transactions on Games* 11, 1 (March 2019), 48–68. doi:10.1109/TG.2018.2870876 Conference Name: IEEE Transactions on Games.

[27] Antonios Liapis, Georgios N. Yannakakis, and Julian Togelius. 2013. Sentient Sketchbook: Computer-aided game level authoring. In *International Conference on Foundations of Digital Games*. SASDG, 213–220. https://api.semanticscholar.org/CorpusID:2103833

[28] Ricardo Lopes and Rafael Bidarra. 2011. Adaptivity Challenges in Games and Simulations: A Survey. *IEEE Trans Comput Intell AI Games* 3 (2011), 85–99. doi:10.1109/TCIAIG.2011.2152841

[29] Paul Merrell and Dinesh Manocha. 2011. Model Synthesis: A General Procedural Modeling Algorithm. *IEEE Transactions on Visualization and Computer Graphics* 17, 6 (June 2011), 715–728. doi:10.1109/TVCG.2010.112 Conference Name: IEEE Transactions on Visualization and Computer Graphics.

[30] Mossmouth. 2024. Spelunky 2 on Steam. https://store.steampowered.com/app/418530/Spelunky_2/

[31] Redacted Review. 2025. Procedural Level Generation Survey. https://figshare.com/s/179a4c324962d110ebf7

[32] Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing & Management* 24, 5 (Jan. 1988), 513–523. doi:10.1016/0306-4573(88)90021-0

[33] Ruben M. Smelik, Tim Tutenel, Rafael Bidarra, and Bedrich Benes. 2014. A survey on procedural modeling for virtual worlds. *Computer Graphics Forum* 33, 6 (2014), 31–50. doi:10.1111/cgf.12276

[34] Oskar Stålberg. 2021. Townscaper on Steam. https://store.steampowered.com/app/1291340/Townscaper/

[35] Roland van der Linden, Ricardo Lopes, and Rafael Bidarra. 2014. Procedural Generation of Dungeons. *IEEE Transactions on Computational Intelligence and AI in Games* 6, 1 (March 2014), 78–89. doi:10.1109/TCIAIG.2013.2290371 Conference Name: IEEE Transactions on Computational Intelligence and AI in Games.

[36] Pauli Virtanen, Ralf Gommers, and Travis E. Oliphant. 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods* 17, 3 (March 2020), 261–272. doi:10.1038/s41592-019-0686-2

[37] Oleg Dolya aka Watabou. 2025. Watabou's Procgen Arcana. https://watabou.github.io/

[38] PCG Workshop. 2025. PCG Workshop. https://www.pcgworkshop.com/

[39] Jichen Zhu, Antonios Liapis, Sebastian Risi, Rafael Bidarra, and G. Michael Youngblood. 2018. Explainable AI for Designers: A Human-Centered Perspective on Mixed-Initiative Co-Creation. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, Maastricht, 1–8. doi:10.1109/CIG.2018.8490433

A Field Evidence

Table 4: PCG adoption patterns in the top 20 best-selling roguelikes on Steam (February 2026, sorted by sales rank). Level Structure categories: Full Gen = algorithmic terrain/geometry generation; Layout Gen = procedural room/node graphs; Chunk Assembly = hand-crafted pieces connected procedurally; Fixed/Minimal = static or arena-based levels.

Game	Level Structure	Builds
The Binding of Isaac: Rebirth	Layout Gen	Deep
Balatro	Fixed (card game)	Deep
Risk of Rain 2	Fixed + variants	Deep
Hades II	Chunk Assembly	Deep
Deadzone: Rogue	Layout Gen	Moderate
Elden Ring Nightreign	Fixed + modifiers	Moderate
Megabonk	Fixed (arena)	Deep
BALL x PIT	Fixed (arena)	Deep
Barony	Layout Gen	Moderate
Hades	Chunk Assembly	Deep
Gunfire Reborn	Chunk Assembly	Deep
Slay the Spire	Layout Gen	Deep
Dwarves: Glory, Death and Loot	Layout Gen	Moderate
Vampire Survivors	Fixed	Moderate
Monster Train 2	Fixed	Deep
Brotato	Fixed (arena)	Deep
CloverPit	Fixed (slots)	Moderate
Ravenswatch	Chunk Assembly	Moderate
Returnal	Chunk Assembly	Moderate
Darkest Dungeon	Layout Gen	Moderate

Summary:
Full procedural level geometry: 0/20 (0%)
Any level generation (Layout Gen + Chunk Assembly): 12/20 (60%)
Fixed or arena-based only: 8/20 (40%)

Table 5: EPC Submissions by Topic

	Art	Design	Other	Art to Design Ratio	Art to Other Ratio
EPC2022	9	1	2	9	4.5
EPC2023	11	2	2	5.5	5.5
EPC2024	22	5	4	4.4	5.5
EPC2025	23	4	6	5.75	3.83
EPC2026	25	2	2	12.5	12.5

B Additional Charts

Additional charts from the survey data are presented here.

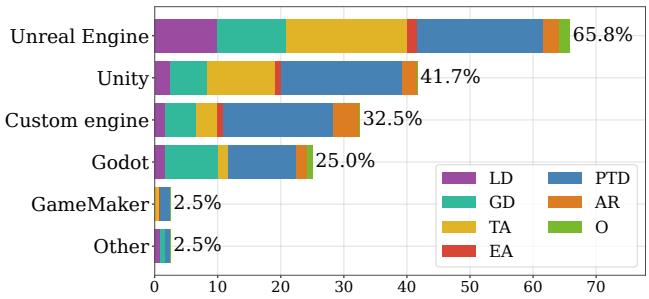


Figure 23: Which game engine(s) do you primarily work with? (Select all that apply)

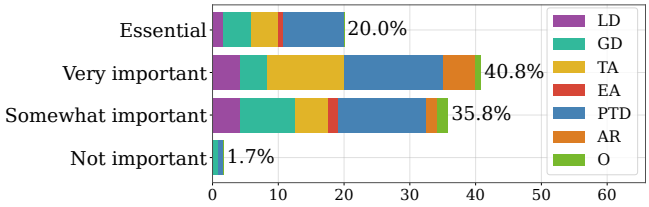


Figure 24: How important is real-time feedback when configuring a procedural level generator?

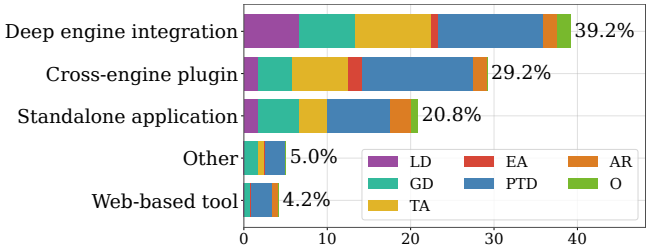


Figure 25: What level of integration would you prefer for a PCG level generation tool?

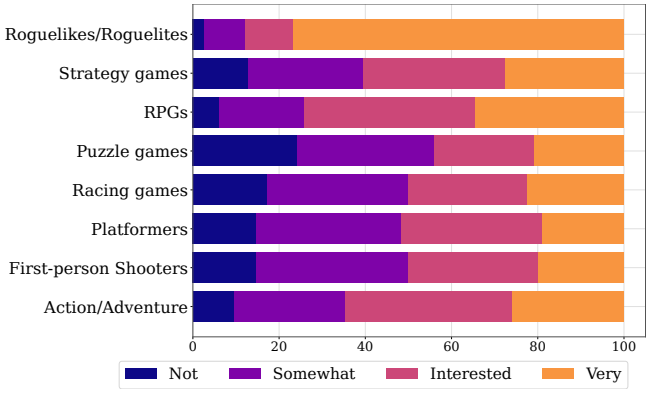


Figure 26: If your project were of the following game genre, how interested would you be in using procedural level generation?

C Qualitative Analysis

Table 6: Thematic codebook with example keywords

Theme	Example Keywords
Control & Flexibility	control, customize, adjust, tweak, precision
Time & Efficiency	time, fast, speed, efficient, iteration, rapid
Integration & Workflow	integration, workflow, pipeline, compatible, export
Technical Barriers	complexity, learning curve, barrier, difficult
Designer Accessibility	designer, non-programmer, user-friendly, intuitive
Debugging & Understanding	debug, transparent, explainable, black box
Quality & Consistency	quality, consistent, reliable, predictable, stable
Content Mixing	mix, combine, hybrid, hand-crafted, blend
Documentation & Learning	documentation, tutorial, guide, examples

Table 7: Theme Frequencies in Open-Ended Responses

Theme	Count	Percentage
Time & Efficiency	28	51.9%
Quality & Consistency	14	25.9%
Control & Flexibility	10	18.5%
Documentation & Learning	6	11.1%
Integration & Workflow	6	11.1%
Technical & Barriers	5	9.3%
Designer & Accessibility	4	7.4%
Content & Mixing	3	5.6%
Debugging & Understanding	2	3.7%

Table 8: Top Co-occurring Theme Pairs

Theme 1	Theme 2	Count
Quality & Consistency	Time & Efficiency	6
Time & Efficiency	Control & Flexibility	4
Technical Barriers	Control & Flexibility	3
Content Mixing	Quality & Consistency	2
Technical Barriers	Designer Accessibility	2
Technical Barriers	Time & Efficiency	2
Control & Flexibility	Designer Accessibility	2
Content Mixing	Integration & Workflow	1
Content Mixing	Time & Efficiency	1
Quality & Consistency	Control & Flexibility	1

Table 9: Open Question Response Statistics

Statistic	Value
Total responses	78
Coded responses	54 (69.2%)
Average themes per response	1.4
Number of themes	9
Number of co-occurrence edges	36
Most central theme	Content Mixing
Connections of most central	8

Table 10: Unigram TF–IDF ranks for PCG workshop papers (term, mean TF–IDF).

1	game	0.077389	51	expressive	0.015517	101	trained	0.010364	151	structure	0.008484
2	level	0.065126	52	methods	0.015448	102	pcg	0.010353	152	wavefunctioncollapse	0.008477
3	levels	0.058514	53	players	0.015338	103	demonstrate	0.010337	153	high	0.008404
4	generation	0.051238	54	semantic	0.015269	104	problem	0.010310	154	elements	0.008356
5	design	0.044961	55	language	0.015082	105	variety	0.010285	155	map	0.008340
6	procedural	0.039475	56	designer	0.015064	106	initial	0.010232	156	collapse	0.008297
7	based	0.038815	57	study	0.014761	107	features	0.010082	157	introduces	0.008281
8	approach	0.035416	58	generative	0.014569	108	experience	0.010038	158	proposed	0.008271
9	games	0.035173	59	3d	0.014379	109	introduce	0.009905	159	allows	0.008268
10	content	0.032329	60	control	0.014259	110	layout	0.009879	160	enjoyment	0.008256
11	wfc	0.028059	61	user	0.014026	111	analysis	0.009845	161	providing	0.008232
12	generate	0.027223	62	spatial	0.013816	112	tasks	0.009833	162	spaces	0.008231
13	learning	0.027209	63	techniques	0.013813	113	number	0.009732	163	test	0.008214
14	player	0.027203	64	new	0.013725	114	interactive	0.009709	164	super	0.008189
15	paper	0.025758	65	difficulty	0.013561	115	time	0.009681	165	solving	0.008184
16	generating	0.025102	66	structures	0.013485	116	existing	0.009644	166	novel	0.008174
17	designers	0.024212	67	quality	0.013394	117	adventure	0.009643	167	skill	0.008107
18	algorithm	0.024185	68	step	0.013315	118	issues	0.009582	168	automated	0.008082
19	process	0.022541	69	evolution	0.013315	119	creative	0.009547	169	rewrite	0.008076
20	model	0.022250	70	representation	0.013313	120	tiles	0.009505	170	node	0.008068
21	generated	0.022034	71	rooms	0.013219	121	end	0.009378	171	editor	0.008054
22	models	0.022018	72	grammar	0.013144	122	capable	0.009373	172	softlocks	0.008043
23	different	0.021902	73	semantics	0.013004	123	instead	0.009284	173	heuristics	0.007991
24	constraint	0.021618	74	patterns	0.012735	124	example	0.009272	174	concept	0.007983
25	exploration	0.021577	75	paths	0.012396	125	mechanics	0.009241	175	purpose	0.007968
26	use	0.021568	76	used	0.012379	126	action	0.009198	176	grammars	0.007950
27	worlds	0.021218	77	automatically	0.012217	127	challenge	0.008932	177	debugging	0.007879
28	initiative	0.021025	78	approaches	0.011912	128	evaluate	0.008904	178	rt	0.007875
29	mixed	0.021025	79	tile	0.011868	129	maze	0.008904	179	strategy	0.007832
30	virtual	0.020972	80	dungeons	0.011670	130	development	0.008871	180	objectives	0.007799
31	world	0.020952	81	large	0.011621	131	bros	0.008828	181	creating	0.007791
32	tool	0.020763	82	create	0.011543	132	solution	0.008812	182	creation	0.007785
33	gameplay	0.020498	83	discuss	0.011435	133	key	0.008812	183	way	0.007769
34	generator	0.020308	84	space	0.011419	134	llms	0.008783	184	edrl	0.007764
35	graph	0.020282	85	propose	0.011370	135	function	0.008782	185	given	0.007704
36	method	0.020225	86	results	0.011325	136	tools	0.008756	186	videos	0.007704
37	using	0.018640	87	segments	0.011304	137	environments	0.008743	187	research	0.007674
38	constraints	0.018498	88	mario	0.011301	138	infinite	0.008733	188	possible	0.007613
39	2d	0.017975	89	challenges	0.011276	139	diversity	0.008725	189	designed	0.007565
40	present	0.017925	90	search	0.011195	140	maintenance	0.008710	190	implementation	0.007565
41	generators	0.017497	91	driven	0.011146	141	knowledge	0.008646	191	rehabilitation	0.007561
42	range	0.017384	92	evolutionary	0.011129	142	minecraft	0.008641	192	designing	0.007532
43	machine	0.017292	93	properties	0.011056	143	including	0.008612	193	racetrack	0.007531
44	data	0.017028	94	able	0.010898	144	path	0.008612	194	runner	0.007496
45	framework	0.016790	95	ai	0.010722	145	rules	0.008575	195	ensuring	0.007492
46	maps	0.016694	96	set	0.010636	146	pcgml	0.008563	196	controllable	0.007459
47	work	0.016038	97	modeling	0.010551	147	generic	0.008537	197	train	0.007444
48	video	0.015648	98	training	0.010547	148	agent	0.008506	198	individual	0.007425
49	procedurally	0.015598	99	plgml	0.010500	149	simple	0.008494	199	need	0.007400
50	dungeon	0.015534	100	output	0.010402	150	grid	0.008490	200	embeddings	0.007398

D Survey

Table 11: Survey Questions

Question	Question
1. How would you primarily describe your professional role? *(Single choice)* • Level Designer • Game Designer • Technical Artist • Environment Artist • Programmer/Technical Designer • Academic/Researcher • Other *Motivation: Understanding role distribution enables analysis of how PCG needs vary across development disciplines.*	**2.** How many years of experience do you have in game development? *(Single choice)* • 0-2 years • 3-5 years • 6-10 years • 10+ years *Motivation: Experience level reveals whether PCG adoption and preferences vary across career stages.*
3. Which game engine(s) do you primarily work with? *(Multiple choice)* • Unity • Unreal Engine • Godot • Custom in-house engine • GameMaker • Other *Motivation: Engine usage informs targeting decisions for PCG tool development and integration.*	**4.** How would you rate your current experience with the following procedural tools? *(Matrix)* • Houdini • Unreal Engine PCG tools • Blender Geometry Nodes • Plugins/Tools that use Wave Function Collapse • Plugins/Tools that use other methods • Custom code-based PCG solutions Scale: No Experience / Limited / Moderate / Extensive *Motivation: Current tool proficiency reveals adoption patterns and gaps in the PCG ecosystem.*
5. If you use procedural generation, what aspects do you currently use it for? *(Multiple choice)* • I don't currently use procedural generation • World building (terrain, vegetation, etc.) • Level layout/structure generation • Mission/quest generation • Enemy/NPC placement • Puzzle generation • Other *Motivation: Identifying where PCG is well-utilized versus underutilized reveals research opportunities.*	**6.** How frequently do you incorporate procedural level generation (not just world building) in your design workflow? *(Single choice)* • Always (essential part of workflow) • Often (most projects) • Sometimes (about half of projects) • Rarely (a few projects) • Never *Motivation: This core question directly measures the adoption gap our research addresses.*
7. What are your primary concerns when considering procedural level generation? *(Multiple choice, up to 3)* • Lack of precise artistic control • Technical complexity/steep learning curve • Difficulty in debugging unexpected outputs • Integration with existing workflows • Time investment to set up compared to potential benefits • Unpredictable results affecting game balance • Unclear design methodology for procedural systems • Other *Motivation: Understanding specific pain points guides future research directions and tool design.*	**8.** Which statement best describes your view on procedural level generation tools? *(Single choice)* • I'm satisfied with the current PCG tools available • Existing PCG tools are too limited in what they can generate • I prefer handcrafting levels and don't see benefits in PCG tools • Most PCG tools are built for programmers, not designers • PCG tools don't give me enough control over the final output • PCG tools are too complex to integrate into my workflow • Other *Motivation: Perception of existing tools validates whether current solutions meet developer needs.*

Continued on next page

Table 11 – *Continued from previous page*

Question	Question
9. What do you consider the most critical factor when evaluating a new design tool? *(Multiple choice, up to 3)* • Flexibility (ability to adapt to various use cases) • Familiarity (resemblance to tools you already know) • Simplicity (low barrier to entry) • Reliability (predictable, stable results) • Feature completeness (comprehensive capabilities) • Documentation and learning resources • Integration with existing workflows • Community support • Other *Motivation: Understanding what convinces developers to adopt new tools reveals threshold requirements for successful design.*	**10.** If using a node-based tool for creating a procedural level generator, which features would be most important to you? *(Ranking, top 3)* • Visual previews of generation steps • Easy debugging of unexpected results • Pre-built common PCG patterns/techniques (WFC, graph grammars, etc.) • Ability to mix procedural and hand-crafted content • Control over generation constraints and rules • Limited or no programming required • Runtime vs. offline generation options • Support for mission/gameplay integration *Motivation: Given node-based interfaces dominate DCC packages, we assess which features matter most within this paradigm.*
11. How important is real-time feedback when configuring a procedural level generator? *(Single choice)* • Essential • Very important • Somewhat important • Not important *Motivation: Understanding feedback importance determines whether tools should prioritize real-time preview capabilities.*	**12.** Which approach to creating procedural generators would you prefer? *(Single choice)* • Building generators from programming primitives (maximum flexibility) • Assembling generators from pre-built, configurable components (balanced approach) • Using templates with limited parameters to adjust (simpler, less flexible) • Mixed-initiative approach where the tool learns from my examples • Assembling generators from pre-built components, with the option to write custom components • Other *Motivation: Different philosophies balance flexibility against complexity; we assess developer preferences.*
13. What level of integration would you prefer for a PCG level generation tool? *(Single choice)* • Deep integration within existing engine (like Unreal Blueprint) • Plugin that works across multiple engines • Standalone application that exports to game engines • Web-based tool accessible from anywhere • Other *Motivation: Integration requires significant development time, making actual user preferences essential before committing to approaches.*	**14.** If your project were of the following game genre, how interested would you be in using procedural level generation? *(Matrix)* • Action/Adventure • First-person Shooters • Platformers • Racing games • Puzzle games • RPGs • Strategy games • Roguelikes / Roguelites Scale: Not Interested / Somewhat / Interested / Very Interested *Motivation: Genre-specific interest identifies underserved areas and research opportunities.*
15. How are levels typically represented and stored in your projects, especially when procedurally generated? *(Multiple choice)* • Rectangular grid/tile-based • Hexagonal grid/tile-based • Free-form geometry • Scene graph/hierarchical structure • Graph-based (nodes and connections) • Hierarchical/nested structures • Node-based graphs (mission/flow graphs) • Navigation mesh • Constraint-based representations • Voxel-based • Other *Motivation: Dominant representation formats directly impact technical architecture and algorithm selection.*	**16.** Which of these approaches to level generation would be most useful to your workflow? *(Single choice)* • Mission-driven generation (gameplay goals determine level structure) • Space-driven generation (spatial layout determines gameplay possibilities) • Balanced approach (iterative refinement between mission and space) • Context-dependent (different approaches for different game sections) • Not sure/would need to experiment *Motivation: Understanding preferences between spatial-first and gameplay-first generation informs tool design philosophy.*

Continued on next page

Table 11 – *Continued from previous page*

Question	Question
17. What role would you prefer AI to play in your procedural level generation workflow? *(Multiple choice, up to 2)* • Assistant-based (AI helps implement your design intentions) • Suggestion-based (AI proposes level designs for you to select and modify) • Learning-based (AI learns from your design patterns and mimics your style) • Analysis-based (AI analyzes and provides feedback on generated levels) • I prefer traditional rule-based PCG without AI involvement • Tool-based (AI enhances specific components of your manual design process) • Full automation (AI generates complete levels with minimal input) • I have no opinion/not sure *Motivation: Current attitudes toward AI predict adoption patterns for AI-assisted PCG tools.*	**18.** When considering AI-assisted procedural level design, which is most important to you? *(Multiple choice, up to 2)* • Maintaining creative control over the final output • Understanding how the AI makes its decisions • Speed of generation compared to traditional methods • Novelty/uniqueness of the generated content • Consistency with existing game assets and style • Learning from my design preferences over time *Motivation: Identifying priority factors guides AI-powered tool development focus.*
19. What concerns you most about using AI in procedural level generation? *(Multiple choice, up to 2)* • Unpredictable or inconsistent results • Lack of control over the generation process • Difficulty integrating with existing tools/workflows • Potential copyright/IP issues with AI-generated content • Performance and computational requirements • Lack of specialized AI tools for level design specifically • Intellectual property/ownership concerns • Potential black box nature (lack of transparency) • Loss of designer agency/control • Other *Motivation: Understanding perceived obstacles helps address barriers to AI-assisted PCG adoption.*	**20.** Which problems do you wish a procedural level generation tool could solve for you? *(Multiple choice)* • Time savings compared to manual design • Ability to create more content variations with consistent quality • Improved iteration speed on level designs • Reduced technical barriers to procedural generation • Better integration with existing workflows • Community/marketplace of shareable generator components • Learning resources and examples for different game genres • Other *Motivation: Identifying which problems developers most want solved prioritizes development focus for future PCG tools.*
21. What is the single most important problem you would want a procedural level generation tool to solve in your workflow? *(Open text)* *Motivation: Open-ended responses capture priorities and pain points that predefined options may have missed.*	

Table 12: Survey response counts per question

Question	Responses	Rate
1. How would you primarily describe your professional role?	120/120	100.0%
2. How many years of experience do you have in game development?	120/120	100.0%
3. Which game engine(s) do you primarily work with? (Select all that apply)	120/120	100.0%
4. How would you rate your current experience with the following procedural tools?	119/120	99.2%
5. If you use procedural generation, what aspects do you currently use it for? (Select all that apply)	119/120	99.2%
6. How frequently do you incorporate procedural level generation (not just world building) in your design workflow?	119/120	99.2%
7. What are your primary concerns when considering procedural level generation? (Select up to 3)	118/120	98.3%
8. Which statement best describes your view on procedural level generation tools?	115/120	95.8%
9. What do you consider the most critical factor when evaluating a new design tool? (Select up to 3)	117/120	97.5%
10. If using a node-based tool for creating a procedural level generator, which features would be most important to you? (Rank top 3)	113/120	94.2%
11. How important is real-time feedback when configuring a procedural level generator?	118/120	98.3%
12. Which approach to creating procedural generators would you prefer?	115/120	95.8%
13. What level of integration would you prefer for a PCG level generation tool?	118/120	98.3%
14. If your project were of the following game genre, how interested would you be in using procedural level generation?	116/120	96.7%
15. How are levels typically represented and stored in your projects, especially when procedurally generated? (Select all that apply)	106/120	88.3%
16. Which of these approaches to level generation would be most useful to your workflow? (Select one)	117/120	97.5%
17. What role would you prefer AI to play in your procedural level generation workflow? (Select up to 2)	118/120	98.3%
18. When considering AI-assisted procedural level design, which is most important to you? (Select up to 2)	114/120	95.0%
19. What concerns you most about using AI in procedural level generation? (Select up to 2)	117/120	97.5%
20. Which problems do you wish a procedural level generation tool could solve for you?	117/120	97.5%
21. What is the single most important problem you would want a procedural level generation tool to solve in your workflow?	78/120	65.0%

Table 13: "Other" responses to Question 8

#	Response
1	Well, I'm making my own PCG tools, but interested in others' too
2	Houdini – too complex/built for programmers. UE PCG – built the same way but with less features. I use Apparance partly for the designer-friendly approach, but it's not well known enough yet.
3	As a programmer I'd normally implement PCG in code, not using an external tool
4	Not many PCG tools available for Godot and making one usually takes too much time for the extent of the project
5	PCG tools are neat, but most often not as accessible to the general public and indies.
6	I do not do games
7	PCG for common elements can allow more time for handcrafting gameplay
8	I don't know enough about them to know where to begin
9	I cannot answer
10	Most PCG tools are bullshit
11	I do see the value of PCG tools, but as a level designer I only use them for set dressing because the level layout itself is superior if handcrafted
12	I am not too familiar with PCG tools honestly, all procedural generation I do, I'm coding myself and ChatGPT together
13	I would like to learn how to use PCG tools for a personal project, but I have not started it yet due to lack of time.
14	With great power comes great responsibility
15	I like to make my own tools, only way to truly understand what you are doing
16	PCG in house are too custom, PCG in open engines are too loose-form, not goal-oriented enough and finicky.
17	While PCG allows for a lot more content to be made, proc levels are often worse designed than hand made
18	Too often I need to reinvent the wheel for a new project because the PCG solution slightly differs from the previous project
19	The time saved isn't sufficient to match the effort expended to reach quality.
20	PCG tools have their benefits but I often find them more time consuming to set up and use than building something myself
21	PCG is powerful but new and growing.
22	My current role and activities require little to no PCG tool use though I am interested in the subject
23	I often need the time to figure out the design setup and constraints, and then there's not enough time to invest in a procedural setup
24	There is always room for improvement.
25	The real benefits of PCG usually come as niche implementations, to take full advantage custom solutions have to be made often