Proceedings of the Nineteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2023)

Level Building Sidekick: An AI-Assisted Level Editor Package for Unity

Camila Aliaga, Cristian Vidal, Gabriel K. Sepulveda, Nicolas Romero, Fernanda Gonzalez, Nicolas A. Barriga*

Faculty of Engineering, Universidad de Talca, Talca, Chile
camila.aliaga@utalca.cl, cvidal@utalca.cl, gsepulveda17@alumnos.utalca.cl, nromero17@alumnos.utalca.cl, feravila.fga@gmail.com, nbarriga@utalca.cl*

Abstract

Developing an original video game requires high investment levels, market research, cost-effective solutions, and a quick development process. Game developers usually reach for commercial off-the-shelf components often available in the engine's marketplace to reduce costs. Mixed-initiative authoring tools allow us to combine the thoughtful work of human designers with the productivity gains of automated techniques. However, most commercial AI-assisted Procedural Content Generation tools focus on generating small independent components, and standalone research tools available for generating full game levels with state-of-the-art algorithms usually lack integration with commercial game engines. This article aims to fill this gap between industry and academia.

The Level Building Sidekick (LBS) is a mixed-initiative procedural content generation tool built by our research lab in association with four small independent game studios. It has a modular software architecture that enables developers to extend it for their particular projects. The current version has two working modules for building game maps, an early version of a module for populating the level with NPCs or items, and the first stages of a quest editor module. An automated testing module is planned. LBS is distributed as an AI-Assisted videogame-level editor Unity package.

Usability testing performed using the "Think-Aloud" methodology indicates LBS has the potential to improve game development processes convincingly. However, at this stage, the user interface and the AI recommendations could improve their intuitiveness. As a general comment, the tool is perceived as a substantial contribution to facilitating and shortening development times, compared to only using the base game engine.

There is an untapped market for mixed-initiative tools that assist the game designer in creating complete game levels. We expect to fill that market for our partner development studios and provide the community with an open research and development platform in a standard game engine.

Introduction

Original game development requires high levels of investment, market research, cost-effective solutions, and short development cycles to catch market trends and effectively bring a differentiated product to the consumer. Software engineering emphasizes using and reusing modular components, which is a tendency in software solutions in the market. For example, web development projects usually share open-source servers and tools. Many companies have the edge over their competitors by being highly proficient in using specific tools, having access to proprietary custom tools built on top of this configuration, or both.

The videogame development ecosystem also follows a modular approach: a base game engine, standard graphic design tools, and modular components on top of the game engine, such as art assets, small functional features, and development tools to modify and extend fully-fledged games. Established companies have many systems and tools from previous games to get shorter development cycles while still improving the quality of their products, effectively differentiating from other companies in the market.

Several tools implement state-of-the-art procedural content generation. However, most of these a) are standalone tools, not integrated into modern game engines; b) are not mixed-initiative, they provide an automated generation with little user control; and c) they don't assist with the generation of full game levels, only specific items, such as vegetation.

Standalone tools are excellent for research projects. They allow for quick iteration and easy user testing because the users do not need to familiarize themselves with extensive pre-existing tools. However, this makes them rather limited in their real-world usage as most development companies have an established software pipeline they use to develop their projects.

Fully automatic procedural content generation, or PCG, is a widespread technique used in many genres of video games. Nevertheless, most video games benefit from carefully crafted levels. Mixed-initiative or AI-assisted authoring tools enable users to combine the thoughtful work of human designers with the productivity gains of automated techniques. Most commercial AI-assisted or mixed-initiative PCG tools focus on generating game bits (small independent components in games). There is an untapped market for mixed-initiative tools that assist the game designer in creating complete game levels. Mixed-initiative techniques that produce high-quality content have been proposed and validated in laboratory settings. Automated techniques have been successfully integrated into game engines and develop-

*Corresponding author.

ment workflows. We are at the right time to integrate both features and create a competitive advantage for our associates.

In this article, we introduce the Level Building Sidekick (LBS), an Artificial Intelligence (AI) assisted videogame design tool that aims to increase the productivity of development teams and the quality of their products. LBS includes AI modules to design game maps, populate them and define quests. It executes seamlessly integrated into the Unity game engine. This integration implies adding special menu options and custom windows to avoid using external tools. Future plans include an automated testing module.

This article is structured as follows. The following section provides some background regarding Mixed-Initiative Procedural Content Generation (MI-PCG). The Software Description section describes the main features and design of LBS, as well as the algorithms it uses. Then, we summarize preliminary usability test results. We finalize the article with our conclusions and planned future work. Appendix A has full usability test results.

Background

PCG (Shaker, Togelius, and Nelson 2016; Hendrikx et al. 2013; Barriga 2019) has uncovered an exciting realm of possibilities for the future of videogames. A potential advantage of PCG is the ability to reduce the costs of follow-up products significantly. This is especially evident with mobile games that employ the technique of reskinning, where multiple games share mechanics but differ in visual components such as textures or maps. By generating these visual elements automatically, the cost of reskinning becomes nearly non-existent.

PCG has the potential to generate virtually unlimited content, as demonstrated in games like No Man's Sky or those that create new environments every time the game is restarted, like Sim City and Civilization (Volkmar et al. 2022). One of the most exciting potentials of PCG is its ability to create personalized content tailored to individual players (Khoshkangini et al. 2021). While this feat is impossible for human game designers, algorithms that learn and act on implicit feedback from players could transform the gaming industry, making for an even more immersive and enjoyable gaming experience.

Over the last few years, research on mixed-initiative content generation tools has surged. Proof-of-concept tools have been successfully built for Real-Time Strategy (RTS) games (Liapis, Yannakakis, and Togelius 2014), puzzle games (Charity, Khalifa, and Togelius 2020), dungeons (Alvarez et al. 2018), narrative (Kybartas, Verbrugge, and Lessard 2020) and mechanics (Saini and Guzdial 2020), among others. However, most projects never move past basic research into applied and industrial applications. One notable example is Speedtree[1], the industry-standard vegetation modeling tool for videogames and cinema. The LBS project aims to bridge the academia-industry gap by developing a mixed-initiative design tool with industry partners, who will develop and use it for their projects.

[1]https://store.speedtree.com/

Basic research in mixed-initiative design and generation for several video game genres and content types is fairly mature. The most common approach is Evolutionary Algorithms, used in Sentient Sketchbook (Liapis, Yannakakis, and Togelius 2014), Baba is y'all (Charity, Khalifa, and Togelius 2020), and Evolutionary Dungeon Designer (Alvarez et al. 2018). Due to their incremental optimization architecture, they provide a natural endpoint for inserting human interaction between evolutionary generations. Machine learning approaches have been gaining popularity, as in Mechanic Maker (Saini and Guzdial 2020). This project will use evolutionary algorithms, which have been more thoroughly explored in mixed-initiative generation and are closer to being ready for application in an industrial setting. Finally, other tools, such as Procedural Generation Grid[2] are integrated into modern game engines but use elementary rule-based generation methods and are not easily extensible.

Software Description

LBS represents the outcome of an academia-industry collaboration project for constructing an authoring tool to improve product quality, development time, and developer and players' experience through AI assistants for defining game maps, scenarios, quests, and visual design. LBS lets software developers define restrictions and graphic elements to build a proposed level. Game designers and artists can collaborate and iterate on the design, with the tool offering suggestions and automating some parts of the process. Those features allow developers to deliver a higher-quality user experience without losing design details.

LBS is designed with a modular, extensible approach. The current version (diagram shown in Figure 1) has two working modules for building game maps, an early version of a module for populating the level with NPCs or items, and the first stages of a quest editor module. An automated testing module is planned.

Module 1: This module generates the space where all items in the game level reside. Most games would call this the game map.

- **Module 1A:** Assists with the design of level interiors. The system automatically translates a graph the user enters into a schematic plan using a local search algorithm for constraint satisfaction and optimization. This plan can be edited by the user and automatically exported to a 3D environment.
- **Module 1B:** Assists with the design of level exteriors. The system generates tile-based maps using a constraint satisfaction algorithm. The user can edit or regenerate parts of the map.

Module 2: The system allows the user to populate the generated level with different configurable elements, such as items, characters, and enemies. The tool provides suggestions through a quality diversity (Gravina et al. 2019) algorithm.

[2]https://assetstore.unity.com/packages/tools/utilities/procedural-generation-grid-beta-195535

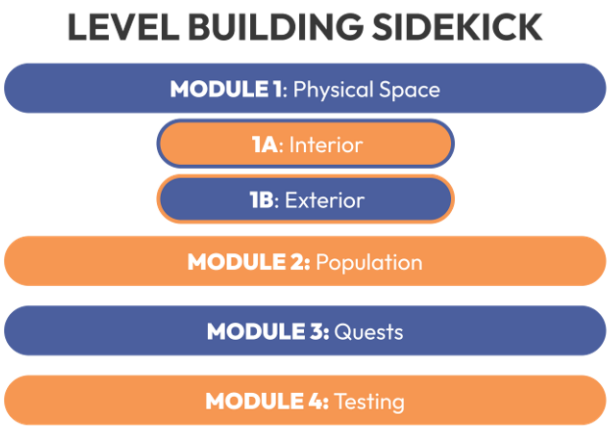

Figure 1: Current and in-development LBS modules.

Module 3: The Quests, or Missions, module is currently in development. The tool will have a fully configurable mission editor based on free context grammars. The editor allows you to expand expressions manually or automatically. Currently, the UI for this module is working, but no assistance is provided, and nothing is exported to the Unity scene.

Module 4: A future fourth module will semi-automatically test the generated levels. This testing can range from basic connectivity checks via Flood-fill or A* to gameplay checks using behavior trees, utility systems, or agents trained by reinforcement learning.

The current LBS version integrates seamlessly into Unity Game Engine, adding special menu options and custom windows to use the tools without needing external executables or installing special GUIs. Figure 2 shows a sample workflow for creating a game level that includes indoor and outdoor zones and some enemies.

Extending LBS

To add a new module to the LBS tool, the programmer must create a new class extending from LBSModule. The new class must contain the data to be modified and the methods to do so. To connect the new module with the LBS interface, the programmer must create a new drawing class extending from Drawer. The drawing class must specify the VisualElement representation of each element in the data and add them to the MainView. Additionally, the programmer has to extend the LBSManipulator class to modify the data. Finally, to use the module in the tool, the user has to go to the class LayerTemplateEditor and add a new method that creates a new Layer and adds the module to the layer, then create a new Tool for each manipulator to use and add them to a new Mode together with the layer and the corresponding drawing class and add this new Mode to a new LayerTemplate. Then on the method called OnInspectorGUI, the programmer needs to create a new button to set the current target of the editor to the new layer template. Once these steps are done, the user can create a layer template and set it as the layer with the user's new module to use in the LBSTool.

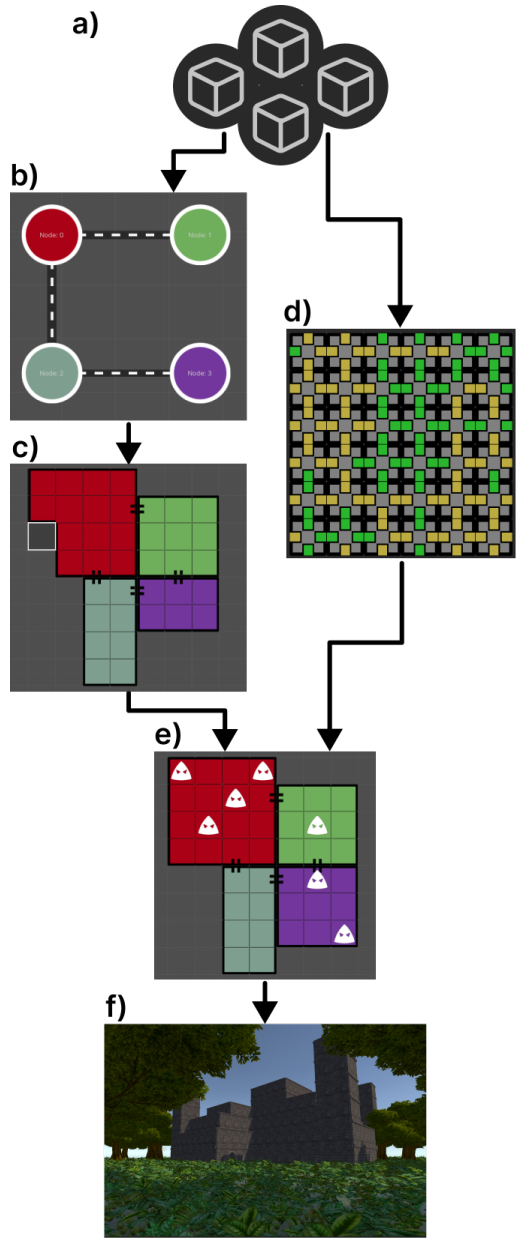

Figure 2: Sample LBS workflow: a) user configures assets (floor tiles, enemy 3D models, etc.); b) user designs an abstract connectivity graph for a building; c) tool creates a blueprint from the connectivity graph, which the user can customize; d) user defines constraints for the exterior environment and tool provides suggestions; e) user populates level, possibly incorporating suggestions from LBS; f) internal level representation is exported to a Unity scene.

Algorithms

The Level Building Sidekick uses several search-based procedural content generation (Togelius et al. 2011) algorithms. In module 1A, the user-defined connectivity graph (or bubble diagram) is translated into a dungeon blueprint. Inspired by work on generating residential building layouts (Merrell,

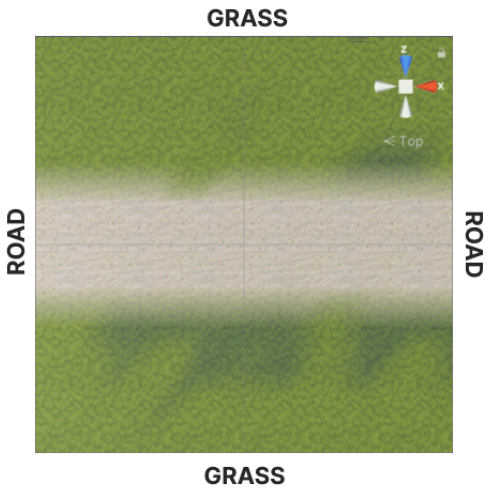

Figure 3: WFC sample tile. Each one is tagged on the four sides. Tags are used to check for legal neighbors.

Schkufza, and Koltun 2010), our algorithm generates disconnected rooms with the sizes and locations specified in the graph. Then it runs an optimization algorithm to minimize unsatisfied constraints defined by the user (room connections and sizes), implicit constraints (rooms cannot overlap), and some predefined aesthetic criteria (empty space between rooms). Currently, we are using a simple steepest ascent hill-climbing algorithm, but it can be easily swapped if testing shows it is not performing as needed.

Module 1B uses Wave Function Collapse (Gumin 2016; Karth and Smith 2017) to assign tiles from a predefined set to cells in a grid. The tiles have tags on each of the four sides, as seen in Figure 3. Tags are used to match tiles to legal neighbors. The user can manually assign tags to cells in the grid, lock tiles suggested by WFC, or erase and re-generate them. Finally, the system can have multiple tiles with the same four tags, and the user can assign a weight to each of them, allowing the system to generate varied environments while maintaining functional coherence.

Module 2 uses a MAP-Elites quality diversity algorithm inspired by the Evolutionary Dungeon Designer (EDD) (Alvarez et al. 2018) and Sentient Sketchbook (Liapis, Yannakakis, and Togelius 2014). It is a classification algorithm that works along an optimization algorithm to present diverse options rather than a single result. We implemented a lightweight, "simple, default version" MAP-Elites (Mouret and Clune 2015). Instead of running an optimization procedure on each MAP-Elite cell, as is usually done, we run a single genetic algorithm and use MAP-Elites as a visualization layer on top of it. This was needed due to the large size of the chromosomes and the need for quick response times. Our implementation uses a 2-dimensional map, with X and Y axes, as shown in Figure 4. Each axis is divided into N partitions, and an evaluation function is assigned to it. In each iteration of the optimization algorithm, the results are placed in the map partition corresponding to the score obtained by the evaluation function of the respective axes. The level of

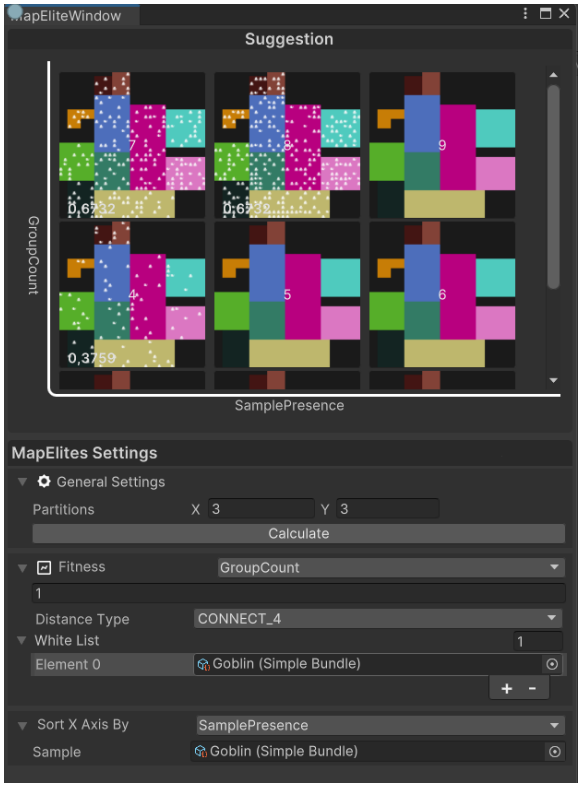

Figure 4: MAP-Elites window. In this example, the user selected *GroupCount* as the function to optimize and *SamplePresence* as the function for the X-axis of the MAP-Elites. The Y-axis is fixed to the same function as the optimized one, as per user feedback, who found dealing with three functions to be too cumbersome. We plan to have an "expert" mode that enables the user to select the Y-axis function.

granularity depends on the number of partitions chosen.

Module 3 is inspired by EDD's Questgram (Alvarez et al. 2021). Though currently just a UI, it will allow users to define a grammar for their quests (Doran and Parberry 2011; Machado, Santos, and Dias 2017), help with quest creation by filtering actions, targets, and rules to what is applicable in the current context, and automatically expand rules if the user chooses to do so.

We are in the process of defining the capabilities that Module 4 will offer. We expect to have at least simple automatic reachability checks and quest legality checks. We are exploring the possibility of adding automated players based on Behavior Trees.

Software Validation

"Think-Aloud" is a testing methodology in which the participants are asked to use a system while continuously thinking aloud, verbalizing their thoughts as they move through the user interface (Nielsen 1994). It has been used in several areas of qualitative research to understand cognitive processes that cannot be directly observed. The objective of the test

is to gather information regarding the interpretations of the elements present in the developed interface and the transformation of this information into improvement proposals for a possible redesign. This test allows us to discover why users make mistakes and which parts are easy for them to use.

The number of participants involved in the study was 9, all males between 22 and 36 years old, corresponding to the target user profile of the project. The selection criterion was that they had advanced experience using the Unity game engine and did not have previous experience with the tool being evaluated. Participants were given two specific tasks to perform related to the construction of rooms, exteriors, and positioning of population elements. This covers modules 1A, 1B, and 2.

The test was carried out at the project development offices and through virtual videoconferencing platforms. An exhibitor explained the software's general operation to the participant before the examiner started the test procedure. Each participant signed an informed consent form accepting voluntary participation in the procedure. The tasks assigned were:

Task 1: Create a 3D dungeon containing:

- 5 rooms
- 3 types of enemies
- 2 prizes
- 2 weapons

Task 2: Create an outdoor environment containing:

- 3 groups of enemies
- 2 points of interest

While working on the tasks, participants were asked to express their thoughts, feelings, and opinions aloud while interacting with the tool. They were not given a time limit for completing both tasks. Still, if they could not complete either task or needed to resolve any concerns regarding the test, they were urged to contact the examiner in charge. For evaluation purposes, the software screen and the participants' facial reactions while they were performing the task were recorded so that this information could be analyzed and contrasted with the verbalization of the process. The duration of the procedure averaged 40 minutes for each evaluation. For detailed results, see Appendix A, Table 2.

Although some participants did not complete the tasks fully, all agreed that the tool contributes to efficiency in development time. All participants agree on the phrase, "I like the tool, I feel it has potential, but there are certain things that I still find difficult to visualize." This difficulty in visualizing the elements and their functionalities is expected due to the learning curve of any tool a user encounters for the first time. Most of the problems encountered are interface development errors affecting the tool's intuitiveness. Table 3 of Appendix A details the most frequent problems encountered in the test.

As a general comment, the participants say that they like the tool, that it has great potential to reduce level development times, and want to continue using it in the future. Table 1 in Appendix A shows some suggestions gathered from verbalizing the process. The limitations of this test are related to the small sample size and the level of progress of the tool at the time of applying the method, which is still at the laboratory validation stage. The most important issues found in testing have already been fixed.

Conclusions

We have presented the Level Building Sidekick (LBS), a mixed-initiative procedural content generation tool built by our research lab in association with four small independent game studios. It has a modular software architecture that enables developers to extend it for their particular projects. The current version has two working modules for building game maps, an early version of a module for populating the level with NPCs or items, and the first stages of a quest editor module. An automated testing module is planned. LBS is distributed as an AI-Assisted videogame-level editor Unity package.

Usability testing suggests that the tool shows promise in assisting game-level designers, even though the user experience still needs work. The strengths found in the tool are that it is perceived as a contribution to the development flow, the functionalities implemented are coherent with the work processes in the development of levels, and it allows to do the job in a short time, compared to the use of a game engine alone. The functionalities that stand out for their ease of use are the positioning of elements in the schema, creating nodes, and optimizing indoor maps. The main points to improve are the UI's intuitiveness, the association of the graphic visualization of nodes and schema with the sizes that correspond to each one, and the feedback delivered by the tool regarding the active processes within it.

Future Work

We plan on improving LBS in several ways:

Algorithm change in module 1A: Evaluation of the performance of alternative optimization algorithms, such as Simulated Annealing or Evolutionary Algorithms, and implement the best performing one for the final version.

Add support for more room layouts in module 1A: Multiple height-levels, multiple floors, or other extensions will allow the tool to be helpful on a broader range of games and applications.

Improve the fitness functions in module 2: Better and more varied suggestions from the tool will improve the overall usefulness and user experience.

Add an alternative algorithm to MAP-Elites in module 2: Algorithms such as novelty search (a quality diversity algorithm), MOEA/D, or NSGA-II (Multi-Objective Optimization Algorithms) might make the tool easier to use for novice users or faster for rapid prototyping.

Add a new module to the tool: Module 4 will semi-automatically test the generated levels. This testing can range from basic connectivity checks via Flood-fill or A* to gameplay checks using behavior trees, utility systems, or agents trained by reinforcement learning.

Add support for lock&key puzzle generation: This support includes changes to modules 1, 2, and 3.

Improve the general usability of the tool: Integrate LBS seamlessly into Unity and each company's current toolset. Documentation and training are critical elements for this aspect.

A Usability Test Results

This appendix contains details about LBS usability tests. Table 1 mentions suggestions made spontaneously by test participants. Table 2 shows task time and completion levels. Table 3 shows problems that were found by more than one participant.

"The distributions that Map Elites yields are very similar, I would like a little more variability."
"I would like the size of the graphs to have a default because I don't know what size I have to put in."
"I would like there to be default dimensions of enemies with respect to the rooms I already generated."
"I would like it specified what each function does in Map Elites."
"I would like to be able to visualize the tools as a toolbar and have it specified what the function of each tool is."
"Artificial Intelligence balancing is important."
"The AI section is very technical, it lacks some feedback on what to do or what each of the functions does."
"I think the tool is very good, it is interesting, but I think it is too oriented to games that are open world. I would like to have more control of where the elements go in games that are more linear."
"It's already useful enough to be able to save time, it takes me little time to make a map that takes me a day or a week. You can already test it and you can have a demo."

Table 1: Suggestions made by test participants.

	Task 1		Task 2		Total Task
Participant	Achievement rate (%)	Time (min)	Achievement rate (%)	Time (min)	Time (min)
n1	100	21:41	100	16:04	37:45
n2	100	22:06	100	09:19	31:25
n3	100	15:50	100	08:23	33:33
n4	80	23:12	20	11:13	24:25
n5	100	11:12	100	14:00	25:12
n6	100	21:42	100	17:15	38:57
n7	100	15:09	0	11:42	27:51
n8	70	27:00	0	14:00	41:00
n9	0	08:37	20	44:20	53:57
Average	83.3	18:42	60	16:15	34:57

Table 2: Task fulfillment percentage for each participant and time spent.

N°	Problems	1	2	3	4	5	6	7	8	9	total
1	Lack of use of *shortcuts*	1									**1**
2	Node inspector cannot be associated with the direct functionality of the tool.	1									**1**
3	Default options were expected in the creation of nodes and AI.	1					1				**2**
4	Assignment of tags that do not correspond to the node configuration.	1									**1**
5	Cannot locate the map within the window space.	1					1		1	1	**2**
6	It is not clear at first glance the order of the layers or which one is being worked on.	1			1			1	1		**4**
7	It could not eliminate elements of the outer layer having the inner layer on top of it.	1									**1**
8	Could not remove the base layer.	1			1		1				**3**
9	Could not save presets after closing the Map elites panel.	1				1					**2**
10	When returning to Graph view, it was expected to be able to reconfigure the nodes to modify the schema, but this was not achieved.		1		1		1				**3**
11	Cannot read or have difficulty identifying tile icons.		1	1							**2**
12	Map Elites functionalities are not understood.	1	1	1	1	1	1	1	1	1	**9**
13	Tiles are expected to be able to rotate.		1				1				**2**
14	Failure to identify which AI tool is in the outer layer.		1				1				**2**
15	Z fighting or interference between textures makes the tool feel unfinished or unoptimized.	1	1								**2**
16	The name of the tool buttons is confusing.			1							**1**
17	Would like to see all graphics at once in Map elites.			1							**1**
18	The creation relationship between graph and schema is not yet clear.			1	1	1	1	1	1	1	**7**
19	Attempts to leave a separate room and can't.				1	1					**2**
20	Expects to be able to move schema tiles.				1	1	1				**3**
21	Tried to select and delete items in a group.				1		1				**2**
22	Did not see the need to use artificial intelligence.				1		1				**2**
23	Didn't know how to find the tiles corresponding to population.				1						**1**
24	Unity in its light-themed version makes the tool look decontextualized.					1					**1**
25	Wanted to use Z-control					1	1				**2**
26	Expect that pressing Hill climbing more than twice will change the results displayed.					1	1	1	1	1	**5**
27	An attempt was made to put a connection from the door to the outside.					1	1				**2**
28	Some feedback was expected at the time of 3D generation.		1			1					**2**
29	Random tree generation is confusing.			1		1	1				**3**
30	An attempt was made to remove a connection and it could not be removed.						1				**1**
31	An attempt was made to assign tags as a group and it was not possible.						1				**1**
32	There are elements that do not correspond to the current functionality being worked on that can be modified: Inspector, Tags, AI.	1	1	1	1	1	1				**6**
33	Try to connect and delete nodes with left click.							1	1	1	**3**
34	Try to define the connection type of several tiles at the same time in the outer layer.							1	1	1	**3**
35	Confusion is generated due to the difference in tile size which is not reflected in the visualization of the generated space.							1	1	1	**3**
36	The brush tool does not save the last selected option.							1	1	1	**3**
37	Attempts are made to use generators that do not correspond to the working layer.							1	1	1	**3**
38	The toolbar is lost when a layer is deselected.							1	1	1	**3**

Table 3: Main problems detected and their incidence per participant.

Acknowledgments

This research is partially funded by the National Agency for Research and Development (Agencia Nacional de Investigación y Desarrollo, ANID), Sub-directorate of Applied Research (Subdirección de Investigación Aplicada, SIA), grant number ID21I10363.

We want to thank our industry partners: Abstract Digital[3], Cienart Studios[4], Altair Films[5] and Frisson Games[6]. for their support throughout this project.

References

Alvarez, A.; Dahlskog, S.; Font, J.; Holmberg, J.; Nolasco, C.; and Österman, A. 2018. Fostering Creativity in the Mixed-Initiative Evolutionary Dungeon Designer. In *Proceedings of the 13th International Conference on the Foundations of Digital Games*. New York, NY, USA: ACM. ISBN 9781450365710.

Alvarez, A.; Grevillius, E.; Olsson, E.; and Font, J. 2021. Questgram [Qg]: Toward a Mixed-Initiative Quest Generation Tool. In *The 16th International Conference on the Foundations of Digital Games (FDG) 2021*. New York, NY, USA: Association for Computing Machinery.

Barriga, N. A. 2019. A short introduction to procedural content generation algorithms for videogames. *International Journal on Artificial Intelligence Tools*, 28(02): 1930001.

Charity, M.; Khalifa, A.; and Togelius, J. 2020. Baba is Y'all: Collaborative Mixed-Initiative Level Design. In *2020 IEEE Conference on Games (CoG)*, 542–549. Osaka, Japan: IEEE.

Doran, J.; and Parberry, I. 2011. A Prototype Quest Generator Based on a Structural Analysis of Quests from Four MMORPGs. In *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games*, PCGames '11. New York, NY, USA: Association for Computing Machinery. ISBN 9781450308724.

Gravina, D.; Khalifa, A.; Liapis, A.; Togelius, J.; and Yannakakis, G. N. 2019. Procedural Content Generation through Quality Diversity. In *2019 IEEE Conference on Games (CoG)*, 1–8.

Gumin, M. 2016. Wave Function Collapse Algorithm. https://github.com/mxgmn/WaveFunctionCollapse. Accessed: 2023-05-26.

Hendrikx, M.; Meijer, S.; Van Der Velden, J.; and Iosup, A. 2013. Procedural Content Generation for Games: A Survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 9(1).

Karth, I.; and Smith, A. M. 2017. WaveFunctionCollapse is Constraint Solving in the Wild. In *Proceedings of the 12th International Conference on the Foundations of Digital Games*, FDG '17. New York, NY, USA: Association for Computing Machinery. ISBN 9781450353199.

Khoshkangini, R.; Valetto, G.; Marconi, A.; and Pistore, M. 2021. Automatic generation and recommendation of personalized challenges for gamification. *User Modeling and User-Adapted Interaction*, 31: 1–34.

Kybartas, B.; Verbrugge, C.; and Lessard, J. 2020. A sketch-based tool for authoring and analyzing emergent narratives. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 16, 319–321.

Liapis, A.; Yannakakis, G. N.; and Togelius, J. 2014. Designer modeling for sentient sketchbook. In *2014 IEEE Conference on Computational Intelligence and Games*.

Machado, A.; Santos, P.; and Dias, J. 2017. On the structure of role playing game quests. *Revista de Ciências da Computação*.

Merrell, P.; Schkufza, E.; and Koltun, V. 2010. Computer-Generated Residential Building Layouts. In *ACM SIGGRAPH Asia 2010 Papers*, SIGGRAPH ASIA '10. New York, NY, USA: Association for Computing Machinery. ISBN 9781450304399.

Mouret, J.-B.; and Clune, J. 2015. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*.

Nielsen, J. 1994. *Usability Engineering*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. ISBN 9780080520292.

Saini, V.; and Guzdial, M. 2020. A Demonstration of Mechanic Maker: An AI for Mechanics Co-Creation. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Shaker, N.; Togelius, J.; and Nelson, M. 2016. *Procedural Content Generation in Games*. Springer.

Togelius, J.; Yannakakis, G. N.; Stanley, K. O.; and Browne, C. 2011. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3): 172–186.

Volkmar, G.; Alexandrovsky, D.; Eilks, A. E.; Queck, D.; Herrlich, M.; and Malaka, R. 2022. Effects of PCG on Creativity in Playful City-Building Environments in VR. *Proceedings of the ACM on Human-Computer Interaction*, 6(CHI PLAY): 1–20.

[3]https://abstractdw.com/

[4]http://www.cienartstudios.com/

[5]https://www.altairfilms.cl/

[6]https://frissongames.com/