

AI-assisted Scene Synthesis in the 3D Educational Game Authoring Tool GameTULearn

Florian Horn, Alessandro Vincenti and Stefan Göbel

Technical University of Darmstadt, Germany

florian.horn@tu-darmstadt.de

stefan_peter.goebel@tu-darmstadt.de

Abstract:

Educational games can improve motivation and learning outcomes but authoring them is still difficult for educators without programming experience. This problem is particularly visible in 3D scene creation, where users must select assets, arrange objects, and maintain spatial consistency inside a complex editor. GameTULearn addresses this challenge as a no-code authoring environment for interactive 3D educational games. This paper presents an AI-assisted scene-synthesis workflow that translates prompt-based natural-language descriptions into editable 3D room layouts inside GameTULearn. The workflow is designed as an assistive layer rather than an automatic generator. A multi-stage pipeline interprets the user request, selects scene-editing functions, retrieves suitable assets and objects, places them on a structured room grid, and executes the resulting operations in Unity. Low-level implementation details such as object instantiation, placement constraints, and validation remain inside the engine. This separation reduces the decision burden for the language model and keeps the generated scene fully editable. We evaluated the workflow in a within-subject user study with 13 participants. Each participant recreated the same educational game scene once with the AI-assisted workflow and once with a traditional drag-and-drop editor. The AI-assisted workflow reached a System Usability Scale score of 80.6, compared with 69.4 for the traditional condition. Participants particularly valued the lower entry barrier and the reduced interaction effort. At the same time, they criticised mismatches between their descriptions and the generated layouts and did not want to give up manual control. The results therefore support a hybrid authoring model in which AI accelerates scene creation, but users remain responsible for review and refinement.

Keywords: Educational games, Authoring tool, Scene synthesis, Large language models, Usability

1. Introduction

Game-based learning (GBL) has long been positioned to address learner disengagement and superficial knowledge acquisition, since play mechanics – challenge, curiosity, control, and fantasy – can foster deep, meaningful learning (Plass, Homer and Kinzer 2015). Yet creating such games remains difficult, especially when educators want domain-specific content rather than generic software (Sousa, Neves and Luz 2023), since building interactive 3D learning environments demands technical expertise, software familiarity, and time most teachers lack (Horn, Vogt and Göbel 2023). This leaves a gap between GBL's pedagogical promise and educators' practical ability to create or adapt games for their own teaching.

GameTULearn was developed to address this gap. Earlier work introduced it as a no-code authoring tool for interactive 3D educational games, aimed at educators without programming expertise, reducing technical barriers while supporting structured game design with generative AI assistance. That work established the tool's concept, usability focus, and role as an authoring environment rather than a low-level development framework (Horn, Vogt and Göbel 2023).

A subsequent study examined how AI could support educators during authoring, finding particular interest in features that reduce content-creation effort, such as generating room layouts from natural-language descriptions – motivating GameTULearn's extension with AI-supported authoring features (Horn and Göbel 2024).

This paper focuses on that extension: AI-assisted scene synthesis, which translates textual descriptions into editable 3D scenes to address authoring barriers.

The contributions are:

- an AI-assisted workflow translating natural-language scene descriptions into editable 3D room layouts inside GameTULearn;
- its technical design, including staged prompt processing, structured world representations, asset/object selection, grid-based placement, and Unity-side validation;
- empirical evidence from a within-subject study showing improved perceived usability, while users prefer AI support that stays open to manual refinement.

2. Related Work

Educational game authoring tools have consistently sought to make content creation accessible to non-programmers. Earlier work, including *StoryTec*, emphasised templates, structured authoring models, and reuse (Mehm et al. 2013; Mehm 2013). Our own prior work extended this direction with *GameTULearn* (Horn et al. 2023). General-purpose tools like *Scratch* and *Construct* show the value of accessible visual environments but are not designed for 3D educational authoring in formal learning contexts.

A second, more recent line of research offers the technical building blocks for AI-assisted authoring: language-driven scene synthesis. Many systems now translate text into 3D scenes or object layouts, but none treats the host authoring environment as a first-class concern. We organise the discussion around four design tensions that clarify what educational authoring requires differently.

Asset space. Many systems operate over open, uncurated object vocabularies – *Open-Universe*, *AnyHome* and *Holodeck* maximise diversity through large object databases (Aguina-Kang et al. 2024; Fu et al. 2024; Yang, Yue et al. 2024). This is reasonable for general scene generation but not for education, where the object catalogue is itself curricular content requiring control, age-appropriateness, and pedagogical alignment. *LayoutGPT*, *InstructScene* and *LLplace* constrains generation to the 3D-Front dataset (Feng et al. 2023; Lin and Mu 2024; Yang, Yixuan et al. 2024), but this remains a research benchmark rather than an educational catalogue. *GameTULearn* instead grounds asset selection in a closed, educationally curated catalogue encoding attributes like name, form, material, and room context.

Interaction model. Several systems produce a complete scene in one generation step, meant to be inspected rather than built collaboratively (Feng et al. 2023; Lin and Mu 2024; Aguina-Kang et al. 2024; Fu et al. 2024; Yang, Yue et al. 2024). Others support iterative editing – *Chat2Layout* via dialogue, *LLplace* via layout editing, and *SceneCraft* via editable Blender code (Yang, Yixuan et al. 2024; Wang et al. 2026; Hu et al. 2024) – but still as self-contained tools with their own interfaces. *GameTULearn* reverses this: the host authoring environment is the primary workspace, and AI assistance is one input modality among others, with proposed changes immediately editable through the regular editor.

Intended output. Many systems target photorealism or research use rather than editability. *I-Design* produces photorealistic renderings via a multi-agent pipeline (Çelen et al. 2025); *Holodeck* targets embodied-AI research (Yang, Yue et al. 2024); *AnyHome* emphasises realistic geometry and textures (Fu et al. 2024). For educational authoring, a scene's value lies in editability, narrative role, and domain relevance, not photorealism – so *GameTULearn* produces editable, structured layouts meant to combine with further game content, not photorealistic interiors.

Locus of geometric control. Most systems place precise spatial reasoning in the model itself: *LayoutGPT* predicts bounding-box coordinates (Feng et al. 2023), *InstructScene* conditions diffusion on scene graphs with spatial relations (Lin and Mu 2024), *Chat2Layout* prompts an MLLM visually over a grid (Wang et al. 2026), and *SceneCraft* has the model emit Blender code that directly manipulates transforms (Hu et al. 2024) – so placement errors surface as visible layout artefacts. *GameTULearn* avoids this: the LLM reasons only over an abstract, grid-based scene representation – which objects belong in which cell, and their high-level relations – never predicting granular coordinates or alignment offsets. Translation into a concrete Unity scene (vertical placement, alignment, collision-free positioning) is handled entirely by deterministic host-engine logic. This is a deliberate division of labour: it keeps the LLM's task within its reliable range while keeping metric precision under the engine's control.

Taken together, these tensions show that *GameTULearn* is not a competitor aiming for more realistic, diverse, or autonomous scene generation. Rather, it integrates scene synthesis into a no-code 3D educational authoring environment under four constraints: full editability within the host environment, asset grounding in a closed educational catalogue, support for iterative human-AI refinement, and engine-controlled spatial precision with the LLM confined to abstract, grid-level reasoning. The next section describes how *GameTULearn* implements them.

3. AI-Assisted Scene Synthesis in GameTULearn

The workflow presented in this paper addresses one specific authoring problem: how to translate a user's textual scene description into an editable 3D room layout inside *GameTULearn*. The implementation does not ask the language model to construct the entire scene directly in continuous 3D space. Instead, it divides the

task into smaller steps and delegates low-level execution to Unity. This keeps the model focused on creative and semantic decisions while the engine handles implementation details and validation.

3.1 Design Rationale

Three design decisions structure the workflow. First, the AI acts as an authoring assistant, not as a replacement for the editor. Users can describe a room in natural language, inspect the result, and continue editing the scene manually. The generated output therefore functions as a starting point, not as a final artefact.

Second, the workflow reduces the numerical and spatial burden placed on the language model. Large language models can reason well about object types, relations, and scene intentions, but they are less reliable when asked to produce precise coordinates or maintain geometric consistency over many objects. *GameTULearn* addresses this by exposing structured representations of the room and by using grid-based placement rather than free continuous positioning.

Third, the workflow separates semantic planning from scene execution. The model decides what should be created and how scene-editing functions should be parameterised. Unity executes those functions, applies internal rules, and validates the result. This division makes the system more robust and keeps the generated scene compatible with normal editor operations.

3.2 Pipeline Overview

The workflow uses a multi-stage pipeline, displayed in Figure 1. As the figure shows, a task determiner first analyses the user's natural-language request and routes it to the appropriate agent: the function list creator for scene-editing requests, the oracle for knowledge questions, or the chatter for general conversation, while the evaluator can review intermediate results and process system error messages, for example when Unity reports an invalid operation. For scene editing, the function list creator produces a list of scene-editing functions, which the parameter extractor, the asset fetcher, and the object chooser then enrich with concrete parameters, assets, and objects. The grid placer assigns the selected objects to valid grid regions and hands the completed function list to Unity for execution. Finally, the summariser reports the performed changes back to the user. Every stage receives the user's natural-language request and the structured context required for its task. The output of one stage becomes the input for the next. The stages are narrower than a single large generation step, which makes the behaviour easier to constrain and inspect.

The first stage identifies the editing actions required to fulfil the user request. Rather than producing direct object coordinates, it generates a list of scene-editing functions such as changing room size, adding architectural elements, placing furniture, or attaching wall-mounted items. This keeps the intermediate representation close to executable editor operations.

The next stages enrich this function list. Parameters are extracted from the user request, suitable assets and objects are selected from the available database, and the resulting instructions are prepared for placement. Asset retrieval is intentionally constrained. *GameTULearn* does not operate on an unrestricted object space. It selects from an existing catalogue of architectural assets and scene objects. For furniture and accessories, object names encode meaningful attributes such as form, material cue, or room context. This allows the model to use language-level semantics during selection without requiring a complex symbolic ontology.

Objects that require placement are then passed to the placement stage. At this point, the system already knows which concrete objects will be used and can therefore expose their sizes and categories to the model. This is an important difference from approaches that first generate abstract bounding boxes and only later retrieve assets. In our case, the restricted asset base makes early object selection more practical and keeps later placement decisions grounded in real object dimensions.

3.3 Structured Scene Representations

The workflow uses three levels of scene representation, visualised in Figure 2. At the highest level, the world representation stores global context such as the scene settings, the available rooms, and the active room under edit. At the room level, the system represents architectural properties and room-specific assets such as walls, floors, ceilings, and windows. At the most detailed level, the interior representation stores the room as a grid of cells with object occupancy information. As Figure 2 illustrates, the three levels nest into one another: each room contains its metadata, its room assets, and its interior grid, and each grid cell stores its coordinates, its cell type, and the objects placed on it.

This hierarchy helps in two ways. It limits the amount of information each stage has to process, and it allows different tasks to receive different levels of detail. Object selection does not need the same context as object placement. Placement, on the other hand, depends on a more explicit spatial representation of the current room state.

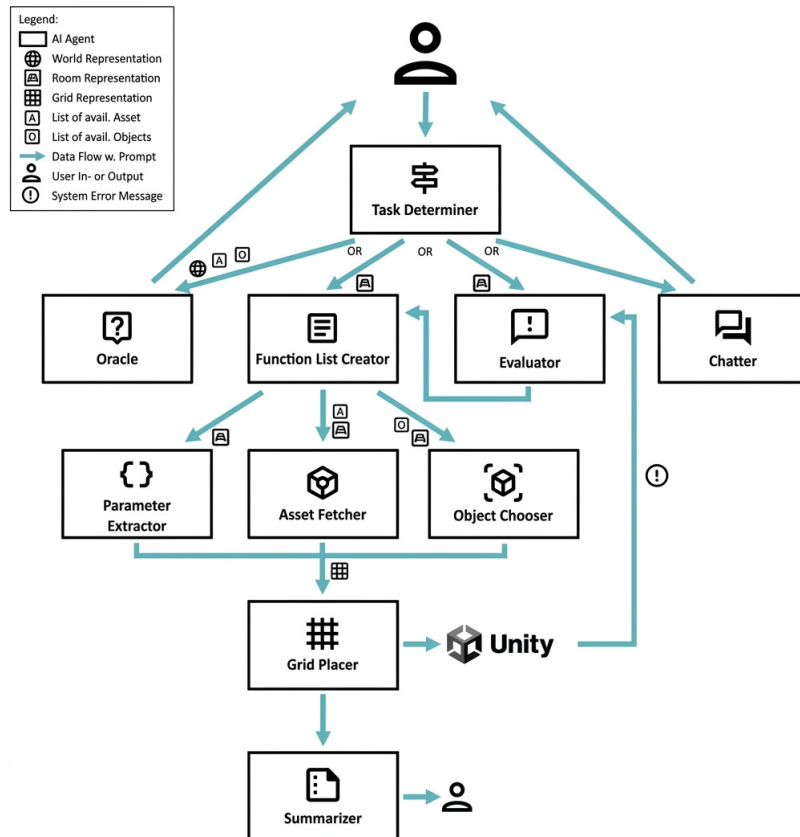


Figure 1: Overview of the multi-stage pipeline for scene synthesis

All internal communication is serialised as JSON. This choice fits the hierarchical structure of the problem and gives the model a stable format for reading and generating intermediate results. The representation includes the current grid, available tools, selected objects, and internal summaries that later help explain the final result to the user.

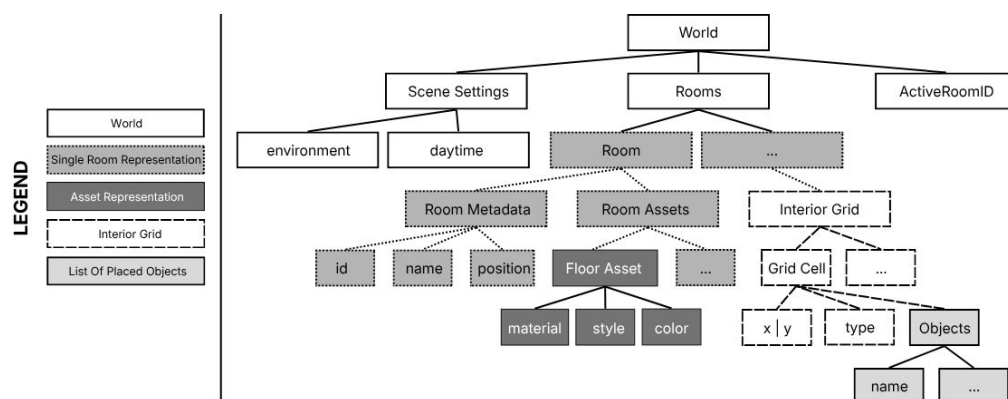


Figure 2: The scene representation with three distinct levels: world representation, room representation and detailed grid placement information

3.4 Grid-Based Placement and Execution

The placement stage uses a grid that integrates the floor, walls, and ceiling into one shared coordinate system. Each grid cell has a type, and placement rules determine which object categories are allowed on which cell types. The grid therefore simplifies spatial reasoning without removing all constraints. Figure 3 shows a typical

result: from a short textual description, the assistant generated a living room in which sofas are arranged around a coffee table and complemented by a fireplace, a television, windows with curtains, and ceiling lighting. All of these elements remain regular editor objects that the user can move, replace, or delete after generation.



Figure 3: Example of an AI-created room layout

This design reduces one of the main weaknesses of language-model-based scene generation: precise spatial reasoning. The model does not have to generate arbitrary coordinates for every object. Instead, it selects valid grid regions and orientations. Some placement details are resolved by the engine. For example, the height of a wall picture or the exact attachment of a curtain (see Figure 4) can be handled through category-specific Unity logic rather than through additional model output. Figure 4 shows this for the case of curtains: the model only decides that the two windows should be covered; Unity dynamically adjusts the size and attachment of each curtain to its window.



Figure 4: Dynamic size adjusting of curtains depending on the window size

Once the final function list has been completed, Unity executes the actions in the editor. This includes object instantiation, room modification, resizing, and attachment of dependent elements. When executing placements, the engine can also position objects inside valid containers: Figure 5 shows objects that were placed and stacked on a shelf wall, either with slight randomness in their arrangement (right) or without (left). The same editor functions used by the AI remain available to the user through normal authoring operations. This is important because it keeps the AI workflow aligned with the rest of the tool rather than creating a separate hidden subsystem.



Figure 5: Objects can be placed and stacked in valid containers, either with randomness (right) or without (left)

3.5 Validation and Editability

Validation is handled in Unity rather than inside the language model. Two checks are particularly important. First, the engine detects invalid placements, such as collisions or attempts to place an object on an incompatible grid cell. Second, it checks object references. If the model refers to an object that cannot be resolved in the current scene state, the request can be rejected or corrected before it damages the scene structure.

This validation strategy supports the central design goal of the workflow: the generated scene must remain editable. Users are not locked into a one-shot generation result. They can inspect the output, move objects,

change assets, or continue prompting the assistant. Figure 6 illustrates how the editor adapts objects to the room layout: wall objects such as the wardrobe are aligned with the wall, while corner objects such as the bed are placed into the corner. The assistance therefore accelerates scene creation, but the authoring environment remains the primary place for control and refinement.



Figure 6: Layout adaptation for wall and corner objects

4. Evaluation

The main evaluation examines whether the AI-assisted workflow makes 3D scene authoring easier to use than a traditional drag-and-drop approach.

4.1 Study Design

We conducted a within-subject study with 13 participants. Each participant recreated the same educational game scene twice: once with the AI-assisted workflow and once with the traditional drag-and-drop editor. The order of the two conditions was counterbalanced to reduce order effects.

The task focused on a realistic authoring scenario. Participants had to construct the same target scene in both conditions, which made it possible to compare interaction styles directly. The AI condition allowed participants to describe the desired room and its contents in natural language and then refine the result. The traditional condition required manual scene editing through the existing editor controls.

Usability was measured with the System Usability Scale (SUS). In addition, task completion time was recorded, a short questionnaire captured perceived layout quality, and qualitative feedback was collected to understand how participants interpreted the two workflows.

We also distinguish between participant subgroups in the AI-assisted condition. Technical affinity was recorded as a self-report and analysed as a binary distinction between high and low technical affinity. Age was analysed as a sample-specific distinction between younger and older participants. This split was based on the two age clusters visible in the participant distribution, which were separated by roughly one generation. These labels should therefore be read as analytic groupings within this study rather than as universal age categories.

4.2 Results

The SUS score comparison between an AI-assisted workflow and a manual workflow can be seen in Figure 7. The AI-assisted workflow achieved a SUS score of 80.6, while the traditional drag-and-drop condition reached 69.4. This difference indicates a clear usability advantage for the AI-assisted condition. A t-test confirmed that the difference was statistically significant ($t = 6.1$, $p < 0.001$). Participants particularly valued the lower entry barrier and the reduction in interaction effort. Instead of constructing every detail manually, they could begin with a textual description and work from a generated first result.

The subgroup analysis depicted in Figure 8 makes this result more precise. Within the AI-assisted condition, participants with high technical affinity reached a mean SUS score of 84.4, compared with 76.9 for participants with low technical affinity. Younger participants reached 86.7, while older participants reached 74.6. Independent-samples t-tests showed statistically significant differences for both subgroup comparisons: high versus low technical affinity ($t = 5.44$, $p < 0.001$) and younger versus older participants ($t = 16.9$, $p < 0.001$). At the same time, the lower-scoring groups still remained in the above-average SUS range, which is important for interpreting the accessibility of the workflow.

The qualitative data adds an important nuance. Participants did not interpret the AI workflow as a substitute for the editor. They appreciated the speed with which they could create an initial layout, but they also reported that some results did not match their intentions precisely. Several participants mentioned mismatches between their descriptions and the generated arrangement, as well as inconsistencies during later modifications. This shows that usability gains do not automatically imply trust in full automation.

The short layout-quality questionnaire and the qualitative comments point in the same direction. Because this part of the questionnaire was a custom instrument rather than a standardised benchmark like SUS, it is best interpreted descriptively. The AI workflow helped users get started faster and reduced mechanical interaction effort, but quality depended on how well the system interpreted the prompt and how easily users could correct the output afterwards. In other words, the benefits of the workflow came mainly from assisted scene construction, not from replacing user judgement.

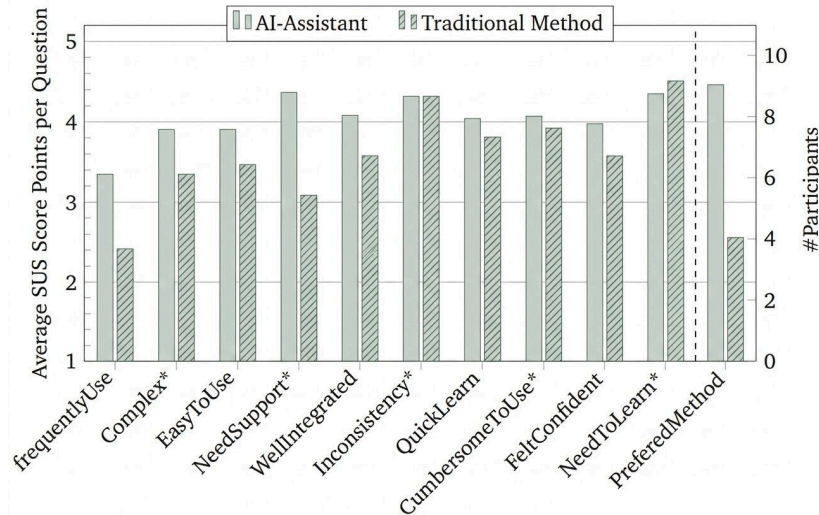


Figure 7: Comparison of the SUS questions between the AI-Assistant model and the traditional method. The respective points for each question have been calculated and negative questions (marked with *) have been inverted

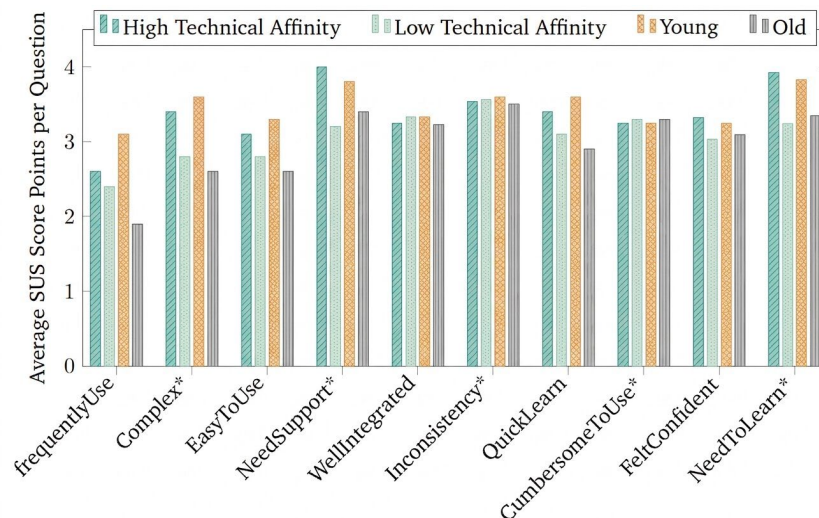


Figure 8: Comparison of the influence of the respective SUS questions on the SUS score per group. For this purpose, the respective points for each question have been calculated and negative questions (marked with *) have been inverted

4.3 Interpretation

The study suggests that AI-assisted scene synthesis is most useful when it is embedded in an editable authoring workflow. The statistically significant SUS difference between the AI-assisted and traditional conditions shows that the workflow improves perceived usability. The value of the system lies in making scene creation easier to begin. Users do not need to place every object from scratch, and they can express layout intentions in a more natural way. At the same time, they still expect to inspect, correct, and refine the scene.

The distinction between high and low technical affinity and between younger and older participants sharpens this interpretation. In this paper, technical affinity refers to the participant's self-reported level of technical familiarity, whereas younger and older refer to the two age clusters present in the sample. These subgroup

results do not mean that the workflow only works well for technically confident or younger users. Even the lower-scoring groups still rated the system above average. However, the consistent gap between the subgroup means suggests that onboarding, transparency, and correction support remain especially important if the workflow is intended for broader educational audiences with more heterogeneous technical backgrounds.

This result matters for educational authoring. Educational scenes are rarely generic interiors. They often need to reflect specific learning tasks, subject matter, or narrative sequences. A workflow that only produces fast layouts but offers poor correction options would not be sufficient. Our results instead support a hybrid model: AI helps users produce a meaningful first version of the scene, while the author remains in control of the final design or further refinement via additional prompts.

5. Discussion

The findings show that scene synthesis can ease authoring in a no-code 3D environment without becoming a fully automatic generator. This is the study's main contribution: the system reduces effort at the point where users struggle most – creating a first coherent spatial layout – by translating natural-language descriptions into structured editor operations.

The technical design explains why this works. The staged pipeline narrows each model decision; structured room representations reduce ambiguity; asset selection is grounded in the existing database rather than an open object space; grid-based placement simplifies spatial reasoning; and Unity-side execution and validation protect the editable scene from common errors. Together, these choices make the workflow usable in an educational authoring context.

The user study also reveals the approach's limits. Participants welcomed the lower interaction effort but still wanted control over the result, since educational authors often need to align scenes with specific teaching intentions, domain objects, or task sequences. AI assistance is therefore most convincing when it supports rather than replaces the authoring process – consistent with prior concerns that generative AI in educational game authoring can propagate bias and encourage over-reliance, risks best mitigated by keeping authors in the loop (Horn, Brandt Da Costa and Göbel 2025).

5.1 Limitations

The small sample ($n = 13$) limits generalisability and statistical power, so results should be read as exploratory feasibility evidence: the direction of the usability effect is credible given the large effect size, but its magnitude needs confirmation in a larger, more diverse sample. Subgroup analyses (by technical affinity and age) split this already-small sample further and lacked correction for multiple comparisons, so they are hypothesis-generating rather than confirmatory.

Internal validity is also limited. The within-subject design controls individual differences but risks carryover and learning effects; counterbalancing reduces but cannot eliminate these, especially since both conditions used the same target scene, potentially advantaging whichever came second. A novelty effect is also possible, since the AI-assisted workflow's benefits may partly reflect its being new rather than inherently more usable.

External and construct validity are similarly constrained: the study covered one authoring task, one scene type, and one comparison condition, so transfer to educators' real courses is unclear. Outcomes were limited to perceived usability (SUS) and a custom, non-standardised layout-quality measure, not downstream design quality or learning outcomes. Qualitative feedback also flagged prompt interpretation and iterative correction as current weak points.

6. Conclusion and Future Work

This paper presented an AI-assisted scene-synthesis workflow for the 3D educational game authoring tool *GameTULearn*. The workflow translates natural-language scene descriptions into editable 3D room layouts through a staged pipeline that combines function selection, asset retrieval, structured placement, and Unity-side execution. Its purpose is to make scene creation easier to start and easier to manage; not to remove the user from the authoring process.

The user study shows that this approach improves perceived usability compared with a traditional drag-and-drop workflow. At the same time, the qualitative results show that users do not want full automation. They want support that saves effort while leaving room for inspection and correction. Hybrid human-AI authoring is therefore not a compromise. In this setting, it is the most suitable operating model.

Future work should improve prompt interpretation and iterative correction, make AI decisions more transparent, and evaluate the workflow on a broader range of scene-authoring tasks. It would also be useful to study how well the approach scales to more complex educational scenarios and richer object sets. Most importantly, future work should move beyond usability and examine educational value, include expert evaluations of the pedagogical quality of generated scenes against learning-design criteria, studies in which educators author games for their own teaching, and classroom studies that investigate effects on learning design outcomes and learner performance, building on established approaches for evaluating digital educational games (Petri and Gresse von Wangenheim 2017; Baigi et al. 2022). Additionally, automatically integrating the generated scene items into the envisioned story of the author – in the case of GameTULearn, via the node editor – could provide value. A further direction is the integration of learning analytics and automated feedback, which could help authors assess and improve the educational quality of their games during authoring (Chaudy, Connolly and Hainey 2014; Cavalcanti et al. 2021; Banihashem et al. 2022; Ruipérez-Valiente et al. 2021). These are the next steps if AI-assisted scene synthesis is to become a dependable part of everyday educational game authoring.

Ethics Declaration: The research referred to did not require ethical clearance.

AI Declaration: During the preparation of this work, the authors used a Large Language Model to improve the language and readability. The original concepts, technical descriptions of the GameTULearn ecosystem, and all empirical data were developed solely by the authors. After using the tool, the authors reviewed, edited, and human-verified all generated output to ensure accuracy and alignment with the technical contributions of the paper. The authors take full responsibility for the final content of the manuscript.

References

- Aguiña-Kang, R. et al. (2024) 'Open-Universe Indoor Scene Generation using LLM Program Synthesis and Uncurated Object Databases'. arXiv. Available at: <https://doi.org/10.48550/arXiv.2403.09675>.
- Baigi, S.F.M. et al. (2022) 'Evaluation Tools for Digital Educational Games: A Systematic Review', *ACTA MEDICA IRANICA*. Available at: <https://doi.org/10.18502/acta.v60i8.10835>.
- Banihashem, S.K. et al. (2022) 'A systematic review of the role of learning analytics in enhancing feedback practices in higher education', *Educational Research Review*, 37, p. 100489. Available at: <https://doi.org/10.1016/j.edurev.2022.100489>.
- Cavalcanti, A.P. et al. (2021) 'Automatic feedback in online learning environments: A systematic literature review', *Computers and Education: Artificial Intelligence*, 2, p. 100027. Available at: <https://doi.org/10.1016/j.caeai.2021.100027>.
- Çelen, A. et al. (2025) 'I-Design: Personalized LLM Interior Designer', in A. Del Bue et al. (eds) *Computer Vision – ECCV 2024 Workshops*. Cham: Springer Nature Switzerland, pp. 217–234. Available at: https://doi.org/10.1007/978-3-031-92387-6_17.
- Chaudy, Y., Connolly, T. and Hainey, T. (2014) 'Learning analytics in serious games: European Conference in the Applications of Enabling Technologies 2014'. Available at: <http://www.ed20work.eu/sites/default/files/ECAETConferenceProgrammeFinal.pdf> (Accessed: 28 April 2026).
- Feng, W. et al. (2023) 'LayoutGPT: Compositional visual planning and generation with large language models', in A. Oh et al. (eds) *Advances in neural information processing systems*. Curran Associates, Inc., pp. 18225–18250. Available at: https://proceedings.neurips.cc/paper_files/paper/2023/file/3a7f9e485845dac27423375c934cb4db-Paper-Conference.pdf.
- Fu, R. et al. (2024) 'AnyHome: Open-Vocabulary Generation of Structured and Textured 3D Homes'. arXiv. Available at: <https://doi.org/10.48550/arXiv.2312.06644>.
- Horn, F., Brandt Da Costa, S.R. and Göbel, S. (2025) 'Bias In, Bias Out? Risks and Opportunities of Generative AI in Educational Game Authoring'. *Mensch und Computer 2025 - Workshopband*, Gesellschaft für Informatik e.V., p. 10.18420/muc2025. Available at: <https://doi.org/10.18420/muc2025-mci-ws17-148>.
- Horn, F. and Göbel, S. (2024) 'AI as a Co-creator: A Survey on AI Support for Educational Game Authoring Tools', in J.L. Plass and X. Ochoa (eds) *Serious Games*. Cham: Springer Nature Switzerland, pp. 3–18. Available at: https://doi.org/10.1007/978-3-031-74138-8_1.
- Horn, F., Vogt, S. and Göbel, S.P. (2023) 'GameTULearn: An Interactive Educational Game Authoring Tool for 3D Environments', in M. Haahr, A. Rojas-Salazar, and S. Göbel (eds) *Serious Games*. Cham: Springer Nature Switzerland, pp. 384–390. Available at: https://doi.org/10.1007/978-3-031-44751-8_32.
- Hu, Z. u. a. (2024) 'SceneCraft: An LLM Agent for Synthesizing 3D Scenes as Blender Code'. *Forty-first International Conference on Machine Learning*. Available at: <https://proceedings.mlr.press/v235/hu24g.html>.
- Lin, C. and Mu, Y. (2024) 'InstructScene: Instruction-Driven 3D Indoor Scene Synthesis with Semantic Graph Prior'. arXiv. Available at: <https://doi.org/10.48550/arXiv.2402.04717>.
- Mehm, F. (2013) 'Authoring of Adaptive Single-Player Educational Games'. Available at: <https://doi.org/10.26083/tuprints-00003721>.

- Mehm, F., Göbel, S. and Steinmetz, R. (2013) 'An Authoring Tool for Educational Adventure Games: Concept, Game Models and Authoring Processes', *International Journal of Game-Based Learning (IJGBL)*, 3(1), pp. 63–79. Available at: <https://doi.org/10.4018/ijgbl.2013010105>.
- Petri, G. and Gresse von Wangenheim, C. (2017) 'How games for computing education are evaluated? A systematic literature review', *Computers & Education*, 107, pp. 68–90. Available at: <https://doi.org/10.1016/j.compedu.2017.01.004>.
- Plass, J.L., Homer, B.D. and Kinzer, C.K. (2015) 'Foundations of Game-Based Learning', *Educational Psychologist*, 50(4), pp. 258–283. Available at: <https://doi.org/10.1080/00461520.2015.1122533>.
- Ruipérez-Valiente, J.A. *et al.* (2021) 'Ideating and Developing a Visualization Dashboard to Support Teachers Using Educational Games in the Classroom', *IEEE Access*, 9, pp. 83467–83481. Available at: <https://doi.org/10.1109/ACCESS.2021.3086703>.
- Sousa, C., Neves, P. and Luz, F. (2023) 'Barriers and Hindrances to the Effective Use of Games in Education: Systematic Literature Review and Intervention Strategies', *European Conference on Games Based Learning*, 17(1), pp. 611–620. Available at: <https://doi.org/10.34190/ecgbl.17.1.1472>.
- Wang, C. *et al.* (2026) 'Chat2Layout: Interactive 3D Furniture Layout With a Multimodal LLM', *IEEE Transactions on Visualization and Computer Graphics*, 32(2), pp. 2243–2259. Available at: <https://doi.org/10.1109/TVCG.2025.3626731>.
- Yang, Yue *et al.* (2024) 'Holodeck: Language Guided Generation of 3D Embodied AI Environments', *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16277–16287. Available at: <https://doi.org/10.1109/CVPR52733.2024.01536>.
- Yang, Yixuan *et al.* (2024) *LLplace: The 3D Indoor Scene Layout Generation and Editing via Large Language Model*, *arXiv.org*. Available at: <https://arxiv.org/abs/2406.03866v1> (Accessed: 28 April 2026).