

Towards An Ontology-Driven Framework for Generated Game Assets

Bogdan Okreša Đurić, Marko Kir

University of Zagreb Faculty of Organization and Informatics

Artificial Intelligence Laboratory

Pavlinska 2, 42000 Varaždin, Croatia

dokresa@foi.unizg.hr, mkir21@student.foi.hr

Abstract. *Procedural content generation has long used ontologies as an index over a fixed library of hand-made assets. Generative models can now synthesise assets on demand, but their output is difficult to control and keep coherent. We propose a conceptual framework that treats asset realisation as a modular step: an ontology specifies what an asset must be and a realiser produces it, either by retrieving from a library or by generating it and validating the result with a vision-language model. We describe the framework, a local-first instantiation in a procedurally generated detective game, and open questions for empirical study.*

Keywords. ontology, procedural content generation, generative AI, vision-language models, neuro-symbolic, video games

1 Introduction

Building the contents of a game world by hand is a slow and expensive process, and it is often the part of game development that creates the most problems with growing scale. This is the gap that procedural content generation (PCG) has tried to close for the past two decades by writing scripts and software that produce levels, maps, items, quests, and whole worlds rather than authoring them one by one (Togelius et al., 2011). A recurring idea within that effort is to give the generator a model of the domain it is generating for, so that what it produces are not simply randomised variations, but meaningful combinations, e.g. a chest belongs in a room, a key opens a door, a witness has something to hide. Ontologies, in their knowledge-representation context, are a natural way to write such a model down, and they have been used to describe game worlds and to drive their generation (Sacco et al., 2016; Tutenel et al., 2009).

Until recently, however, the ontology in such systems did little more than point at things. A slot in the world, e.g. “a medieval cooking implement on the kitchen table”, was resolved by looking the description up against the descriptions of a curated library of hand-made assets and binding the slot to whichever model

matched best. The ontology was, in effect, a semantic index over a fixed catalogue. The recent rapid development of capable generative models, which can synthesise a three-dimensional object or a texture from a textual description, changes what is possible in the described scenario: the wanted model no longer needs to be sought for and hopefully found, it can be created. The difficulty is that generative models are challenging to steer. Left to themselves, they produce output that is inconsistent in style, ignores constraints of scale or function, and is rarely ready to drop into a playable level without further manual work (H. Lee et al., 2024).

This paper argues that such processes, depending on generative models, can be guided by an ontology, and proposes a conceptual framework in which asset realisation is a modular operation. The ontology specifies what an asset must be, while a realiser turns that specification into a concrete, placeable asset. A realiser may retrieve a pre-made asset from a library, which implements the classical approach as a special case, or it may generate one and then check the result against the specification using a vision-language model (VLM) before accepting it. The retrieval and generative realisers follow the same process, which is what lets one be substituted for the other and what lets the earlier approach be read as the baseline which the framework described in this paper departs from.

The conceptual framework presented here treats asset realisation as a modular step: an ontology specifies what an asset must be, and a realiser produces it, either by retrieving from a library or by generating it and validating the result against the specification with a VLM. The baseline retrieval realiser is the system built for the second author’s master’s thesis (Kir, 2026); the generative realiser is the extension this paper sets out. We deliberately keep the framework independent of any particular generative model, and describe a local-first instantiation, running self-hosted generation and validation, as one instance of it rather than as the contribution itself.

The rest of the paper is organised as follows. Section 2 reviews the related work across PCG, semantic game modelling, generative 3D content, vision-

language validation and neuro-symbolic control, and positions the framework against the closest systems. The framework is presented in Section 3. Section 4 describes use-case examples of retrieval and generation. Section 5 discusses what the framework covers, what remains difficult, and how it should be evaluated. The conclusions are laid out in Section 6.

2 Background and Related Work

The vocabulary this paper relies on comes from the search-based PCG taxonomy of Togelius et al. (2011), which separates what is generated, how it is represented, and how its quality is judged. Two broad families of methods can be recognised therein: constructive and search-based approaches, which build or optimise content directly, and approaches that learn to generate from examples, comprised under the name PCG via machine learning by Summerville et al. (2018). A thorough introduction to the topic of PCG is given by Shaker et al. (2016). Recent survey papers observe the rapid entry of large language models into the generation pipeline (Farrokhi Maleki & Zhao, 2024; Gallotta et al., 2024). Our framework is agnostic about which of the recognised method families plays the role of the realiser. What it provides is a semantic specification around the generator and a validation step after it.

The argument that game worlds should be backed by a semantic model, rather than by ad-hoc data structures, was made earlier by Tutenel et al. (2009), who showed that semantics improve both authoring and runtime consistency. Publications coming from the authors of the same community used Semantic Web ontologies and linked data to drive generation: Sacco et al. (2016) generate game content from ontologies and open knowledge sources, Sacco et al. (2017) give a vocabulary for describing game characters, and Barros et al. (2016) generate adventures from open data. An ontology combining role-playing-game concepts with multi-agent organisation was defined in (Okreša Đurić & Schatten, 2016) and used to drive model-driven gameplay (Schatten et al., 2017). In all of these, however, the ontology informs selection or arrangement of existing content; it does not specify content to be created or generated.

Text-to-3D generation became practical with optimisation-based methods that distil a two-dimensional diffusion prior into a three-dimensional shape, of which DreamFusion is the prime example (Poole et al., 2023). The field has since broadened considerably, as reported in the state-of-the-art report by Fan et al. (2026) and H. Lee et al. (2024). Of particular relevance is recent work explicitly targeting game-ready output, with engine-friendly topology and materials (Lei et al., 2025), since the requirement that a generated asset actually drop into a playable level is the hard part for our purposes. The use of vision-language models to steer 3D generation itself

is now established: Bai et al. (2025), for instance, employ a VLM as a differentiable semantic and spatial foundation inside the generation process. Our use of a VLM can be observed to be coming after generation rather than within it, as a check against the brief rather than a training signal. These models supply the generative realiser, yet none of them carries a semantic specification of its own.

The idea of using a capable model as an evaluator was established in the “LLM-as-a-judge” setting (Zheng et al., 2023) and extended to the multimodal case, where a VLM scores visual content against stated criteria (S. Lee et al., 2024). Most directly relevant to the contents presented in this paper is the use of a VLM inside a generation loop: Singh and Zheng (2023) decompose a prompt into separate assertions, check each against the generated image with visual question answering, and refine until the assertions are satisfied. Our validation stage is of exactly this kind, with the assertions supplied by the ontology rather than by a free-text prompt.

Specifying a space of acceptable artefacts symbolically, i.e. through a definition and a list of properties, and then instantiating them, i.e. producing members of it, is an old idea in PCG; the answer-set-programming approach of Smith and Mateas (2011) is the clearest precedent, generating content that satisfies declared constraints. Combining such symbolic specification with sub-symbolic generation is the subject of neuro-symbolic AI, whose integration patterns are catalogued by Kautz (2022) and surveyed by Marra et al. (2024). Our framework is a neuro-symbolic one in the following sense: a symbolic ontology governs a sub-symbolic generator, and a VLM re-expresses the generator’s output in symbolic terms so that it can be checked.

The content presented in this paper can be recognised as being near two publications in particular, even though they clearly differ. Holodeck (Yang et al., 2024) generates 3D environments from a language description by prompting a large language model for spatial constraints and then retrieving assets from a library of tens of thousands of annotated models. SceneCraft (Hu et al., 2024) synthesises a scene as engine code and, notably, uses a VLM to inspect the rendered result and refine it. Both of the approaches presented in these papers arrange or place assets rather than generating them, and both rely on the implicit semantics of a language model rather than on a formal ontology. The framework proposed here differs on two perspectives at once: the specification is an explicit ontology, and the realiser may generate the asset, with the VLM validating it against that specification.

A more recent line of work does generate assets from a structured specification rather than retrieving them, which brings it closer to the present proposal. PhysForge (Yang et al., 2026) has a vision-language model plan a hierarchical “blueprint” of material, functional, and kinematic constraints, which a physics-grounded

generator then realises into simulation-ready assets. CubePart (Zhu et al., 2026) takes a user-defined parts schema and generates game-engine-ready meshes that respect that structure without manual post-processing. Closer still to the idea proposed herein, the agentic scene generation of Liu et al. (2025) couples asset generation with a vision-language verification step that re-checks the result against a structured spatial context and regenerates when it is inconsistent.

These systems share with our framework the use of a structured specification to steer generation, and in one case a generate-then-verify loop. They differ from it, however, on the points that define our contribution: the specification is authored implicitly by a language model rather than declared as a formal ontology. Furthermore, the VLM acts as a planner or a reward signal rather than as an acceptance gate that admits or regenerates a candidate. Finally, none of them treat retrieval and generation as one interchangeable realiser, which is what lets the classical, library-bound approach be recovered as a special case of the same framework. We therefore claim novelty of the proposed framework for this specific combination, and note that, were a system to couple a formal ontology with generative assets and a validation loop in the same manner, the claim would narrow to its application within a procedurally generated detective game.

Procedurally generated mystery is itself small literature, confined largely to workshop work that generates logically consistent murderous story plots from constraints (Mohr et al., 2018; Stockdale, 2016). We draw on it for the testbed but make no narrative-generation contribution of our own.

3 The Framework

The framework rests on a single abstraction. The ontology declares slots, each of which must be filled by a concrete asset, and for each slot the system derives a realisation brief, i.e. a short structured statement of what the asset must be. A realiser is any component that takes a brief and returns a bound asset. Because of this modular and clearly defined nature of the described process, i.e. brief as an input, a bound asset as an output, different realisers can be substituted freely, and the classical retrieval approach and the generative approach proposed here become two instantiations of one pattern rather than two unrelated systems. The described structure can be seen organised in layers in Fig 1.

The knowledge layer is the ontology itself, and it is the part of the system that knows nothing about any engine, library, or generator. It holds the asset taxonomy, the properties, and attributes of each type, e.g. a lamp may be a light source and, in a particular game, a possible weapon, the spatial and connectivity rules that make a level playable, the constraints on style and scale, and the schema of the mystery. This inde-

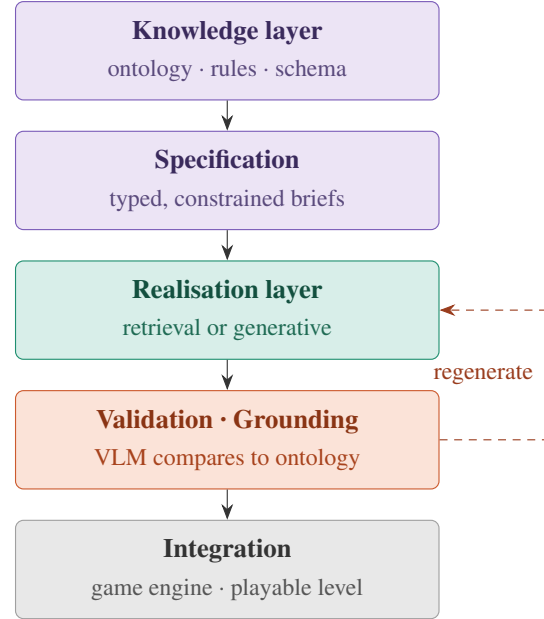


Figure 1. The layered structure of the proposed framework, where a symbolic knowledge layer derives constrained briefs, a swappable realisation layer produces assets, and a validation stage grounds them against the ontology and returns to regenerate on rejection before integration into the engine.

pendence is deliberate, as it lets the layers beneath it change without disturbing the model of the domain.

The specification layer reasons over a specific instance of a world, e.g. the particular mystery and the particular layout that have been generated, and emits the set of slots that need filling, together with a brief for each. The brief assembles the slot’s semantic type, the constraints that apply to it, and its relational context, and it states the criteria under which a candidate asset will be accepted. The brief can be considered as a type of contract, since both the realiser and the validation stage are written and evaluated against it.

The realisation layer is the swappable and most modular component and the central point of the contribution presented here. A retrieval realiser resolves a brief by selecting from a curated library, which is the baseline. On the other hand, a generative realiser resolves it by synthesising a candidate and validating it, which is detailed in Fig. 2.

The validation and grounding stage closes the loop. A generated candidate asset is inspected by a VLM, which is asked, in effect, whether the candidate satisfies the brief: is it the right kind of object, in the right style, at the right scale, plausibly placeable where the brief intends? A candidate that satisfies the brief is bound, whereas one that does not is rejected and regenerated. It is worth stating plainly where the control over the generator is located because it is not inside the generator. The generator is controlled in the two stages that surround it, as shown in Fig. 2: the conditioning

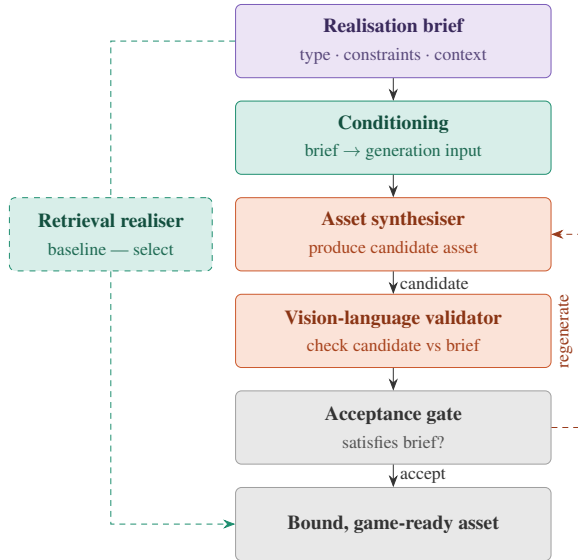


Figure 2. The realisation and grounding loop containing the following steps: 1) the brief is conditioned into a generation input, 2) a synthesiser produces a candidate, 3) a vision-language validator checks it against the brief, 4) the acceptance gate binds the asset or returns to regenerate. The retrieval realiser is the baseline, as it provides the brief-to-asset process by selection.

that turns the brief into the generator’s input, and the acceptance gate that enforces the brief on the output, with the VLM serving as the means of applying a symbolic criterion to a synthesised image.

The integration layer assembles bound assets into the playable level and runs the game. It is the only layer that is engine-specific.

The scope of the described process should be discussed briefly. The same validation stage serves both realisers: in the generative case it is the VLM check just described, and in the retrieval case it resolves to a simple library-fit check. Presenting validation as one stage with two implementations, rather than as a framework feature added to the generative process, is what keeps the baseline and the proposed system inside a single architecture.

To make the loop concrete, consider a single slot from a generated case: a cooking tool resting on the kitchen table, which the mystery has marked as a candidate murder weapon. The specification layer derives the brief shown in Fig 3 from the ontology, drawing the slot’s type, its attributes, the level’s style definition, and the placement constraints from the classes and individuals already present in the world instance.

Conditioning turns this brief into the generator’s input, and the synthesiser returns a first candidate: a cauldron. Rendered from several viewpoints and put to the validator, it is checked against the four acceptance criteria in turn. The validator confirms that the object is a single cooking implement, that its period and style

```

slot:      kitchen/implement_1
type:      CookingImplement
style:     medieval, low-poly, muted palette
attributes: graspable, can_be_weapon
scale:     longest dimension 20-60 cm
place:     upright on KitchenTable, near Hearth
role:      candidate murder weapon
accept if:
  - one cooking implement, nothing else
  - period and art style match the level
  - longest dimension within the set
  - rests upright and stable on a surface
  
```

Figure 3. A realisation brief for one slot, derived from the ontology and passed unchanged to either realiser

match the level, and that it stands upright on a flat surface, but it reads the proportions as roughly knee-height, well above the sixty-centimetre ceiling the brief allows. The candidate is rejected, and the slot is returned for regeneration with the scale constraint tightened. The second candidate, a smaller cast-iron pot, satisfies all four criteria and is bound to the slot. The check that earned its place here is the one a casual glance would have let through: the cauldron looked entirely plausible in isolation, and only the brief’s explicit scale band, applied by the validator, caught that it would have towered over the table it was meant to sit on.

The above example also shows the retrieval baseline and the generative realiser meeting at the same interface: the baseline would resolve this identical brief by selecting the nearest medieval pot from the library, where the generative realiser synthesises one and proves it against the brief before accepting it.

4 Use-Case Examples

The example use-case scenario is based on a small-scale detective game: a single location, a handful of characters, one of them a victim and one is a culprit, a scattering of clues, and a player whose task is to use the clues to deduce who is the culprit. The world, the cast, and the mystery elements are generated from the ontology at the start of the play, so that no two cases are identical. The game is built in a contemporary engine, e.g. Godot, with the ontology and the realisers running in a separate local service that the engine talks to.

4.1 The Baseline Realiser

In the context of the baseline example, which is the system of the second author’s thesis (Kir, 2026), each slot is bound by retrieval from a curated library of assets. The ontology defines the parts of a room as a small grid of floors, walls, objects, and placement points for items and characters, and the generator assembles a playable map by selecting and placing library assets according to the ontology’s rules. Dialogue is authored per role

and assigned to characters according to the parts they have been dealt in the mystery. This is the approach that deals only with retrieving the assets, i.e. it should produce a working game, or at least an in-game world, in which the ontology indexes a fixed catalogue.

4.2 The Generative Realiser

The generative realiser replaces selection with synthesis, as shown in Fig. 2. A brief arriving at the generative realiser is first conditioned into the form the synthesiser expects. This is a deterministic step that translates the ontology's constraints, the asset's type, its style description, its rough dimensions, and its relational context, into the generator's input. The synthesiser produces one or more candidate assets. Each candidate is passed to the vision-language validator, which checks it against the brief's acceptance criteria. The acceptance gate then binds a candidate that passes the validator or, if it does not, returns to synthesis. Accepted assets pass through a step that prepares them for the engine before they are bound into the level.

The instantiation is deliberately local-first: generation and validation run on self-hosted models, with nothing leaving the machine. The motivation is partly practical, since it removes per-call cost and works off-line, and partly methodological, since a self-contained pipeline is far easier for a reviewer or a successor to reproduce than one that depends on a remote service whose behaviour changes without notice. Furthermore, online services often feature various constraints on what may or may not be generated, while this limitation can be mitigated when locally hosted models are used. We stress that the choice of any particular model is an instantiation detail; the framework is stated over the roles needed by the process, not the particular models that fill them.

5 Discussion

The role of the ontology in the finalised process is threefold:

- It makes the generator controllable, by supplying the constraints that are enforced at the conditioning and acceptance layers.
- It makes the pipeline interpretable because a rejection is attributable to a specific unsatisfied criterion rather than to an opaque score.
- It makes the realiser swappable because retrieval and generation differ only in how the brief is approached, not in the role they play in the whole process.

The first of these is the answer to the persistent issue of generative models being uncontrollable. The control mechanism can be found in the proposed framework, but it may be recognised in the ontology-based briefs

and the symbolic context given to the model rather than in the model itself.

The obvious challenge that can be recognised in the proposed framework is generating truly game-ready assets. A synthesised asset that looks correct in a preview may still have unusable topology, the wrong scale, or geometry that the engine cannot collide against, and no amount of semantic validation makes a malformed mesh playable. The acceptance gate mitigates this only to the extent that the VLM can detect it from a rendered view, which is a real limitation and a place where the framework depends on how good a generation model is combined with it. A reasonable fallback, where 3D synthesis proves unreliable, is to generate two-dimensional textures or sprites instead, which current models handle far more dependably. The proposed framework is indifferent to which assets are generated, since both are simply made by different realisers, even though the process itself is unmodified.

The questions worth asking are, for example:

- Does the ontology-conditioned generation produce assets that satisfy their briefs?
- At what cost are those assets produced?
- Does the resulting level remain playable?

These suggest a small set of measures, such as:

- the acceptance rate of generated candidates,
- the number of regenerations needed before acceptance,
- the proportion of brief constraints satisfied,
- a rating of stylistic coherence across a generated level, which a panel of judges or a selected VLM can supply,
- the validity of placements,
- the playability of the finished level.

Several of these measures can be taken with existing instruments rather than devised from scratch. The stylistic and surface quality of generated candidates can be scored with a purpose-built vision-language metric for 3D objects such as Gen3DEval (Maiti et al., 2025), which is designed to align with human judgement without requiring a ground-truth reference, and the scene-level measures can follow the protocol used for comparable systems (Liu et al., 2025), combining standard image-text similarity with human and model ratings of aesthetic quality and functional plausibility.

The natural comparator is the retrieval baseline, against which the generative realiser should be measured for coherence, variety, and authoring effort. A proof-of-concept at modest scale is planned for a future study. Such an example can be used to provide empirical answers to validity and feasibility of the proposed

framework by providing the answers to the above measures, including taking into account the amount of time needed for the whole process to run.

The proposed framework is devised with generalisation in mind. Nothing in the framework is specific to detective games, which are used as examples' genre. The genre is a convenient testbed because its small, well-structured worlds keep the ontology tractable and the playability criteria sharp, but the same brief-and-realiser arrangement applies wherever a game world can be specified semantically and its contents synthesised. Furthermore, the framework is not created with a particular set of models in mind either, i.e. the proposed framework defines the steps of a process and the roles that are needed for its completion, but the specific actors in the process may be interchangeable.

6 Conclusion

We have argued that the ontology in ontology-driven content generation can do more than simply index a catalogue: it can specify assets to be synthesised, and a VLM can check that the synthesis met the specification. Defined as a swappable modular realiser between a symbolic specification and a validation loop, this approach generalises the often-used retrieval method and proposes a way to incorporate generative models into the game assets' development pipeline as a controllable resource. The framework is presented here with the described baseline and generative use-case scenarios, followed by a conceptual study of those specific systems. Further research opportunities are observed in developing an empirical study of a specific instantiation, along with the brief schema that it is based on.

Acknowledgement

The authors acknowledge that Claude Opus 4.8 by Anthropic was used to assist in editing and streamlining sections of this paper.

References

- Bai, W., Li, Y., Luo, W., Chen, W., & Sun, H. (2025). Vision-language models as differentiable semantic and spatial rewards for text-to-3d generation [Preprint]. *arXiv preprint arXiv:2509.15772*.
- Barros, G. A. B., Liapis, A., & Togelius, J. (2016). Playing with data: Procedural generation of adventures from open data. *Proceedings of the First International Joint Conference of DiGRA and FDG*. <https://doi.org/10.26503/dl.v2016i1.801>
- Fan, H., Chen, H., Huang, Z., Liu, Z., & Sheng, L. (2026). 3D Asset Generation: A Survey of Evolution Towards Autoregressive and Agent-Driven Paradigms. *Frontiers of Computer Science*, 20(11), 2011710. <https://doi.org/10.1007/s11704-025-50381-5>
- Farrokhi Maleki, M., & Zhao, R. (2024). Procedural Content Generation in Games: A Survey with Insights on Emerging LLM Integration. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 20(1), 167–178. <https://doi.org/10.1609/aiide.v20i1.31877>
- Gallotta, R., Todd, G., Zammit, M., Earle, S., Liapis, A., Togelius, J., & Yannakakis, G. N. (2024). Large Language Models and Games: A Survey and Roadmap. *IEEE Transactions on Games*, 1–18. <https://doi.org/10.1109/TG.2024.3461510>
- Hu, Z., Iscen, A., Jain, A., Kipf, T., Yue, Y., Ross, D. A., Schmid, C., & Fathi, A. (2024). SceneCraft: An LLM Agent for Synthesizing 3D Scenes as Blender Code. *Proceedings of the 41st International Conference on Machine Learning*, 235, 19252–19282.
- Kautz, H. A. (2022). The Third AI Summer: AAAI Robert S. Engelmore Memorial Lecture. *AI Magazine*, 43(1), 105–125. <https://doi.org/10.1002/aaai.12036>
- Kir, M. (2026). *Developing a Detective Video Game Based on Ontologies and Procedural Content Generation* [Master's thesis, University of Zagreb Faculty of Organization and Informatics] [Forthcoming].
- Lee, H., Savva, M., & Chang, A. X. (2024). Text-to-3D Shape Generation. *Computer Graphics Forum*, 43(2), e15061. <https://doi.org/10.1111/cgf.15061>
- Lee, S., Kim, S., Park, S., Kim, G., & Seo, M. (2024). Prometheus-Vision: Vision-Language Model as a Judge for Fine-Grained Evaluation. In L.-W. Ku, A. Martins, & V. Srikumar (Eds.), *Findings of the Association for Computational Linguistics: ACL 2024* (pp. 11286–11315). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2024.findings-acl.672>
- Lei, B., Li, Y., Liu, X., Yang, S., Xu, L., Huang, J., Tang, R., Weng, H., Liu, J., Xu, J., Zhou, Z., Zhu, Y., Xing, J., Xu, J., Ma, C., Yan, X., Yang, Y., Wang, C., Xu, D., ... Guo, C. (2025). Hunyuan3D studio: End-to-end AI pipeline for game-ready 3d asset generation [Technical Report]. <https://arxiv.org/abs/2509.12815>
- Liu, X., Tai, Y.-W., & Tang, C.-K. (2025). Agentic 3D scene generation with spatially contextualized VLMs [arXiv:2505.20129].
- Maiti, S., Agapito, L., & Kokkinos, F. (2025). Gen3DEval: Using vLLMs for Automatic Evaluation of Generated 3D Objects. *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 18552–18562. <https://doi.org/10.1109/CVPR52734.2025.01729>

- Marra, G., Dumančić, S., Manhaeve, R., & De Raedt, L. (2024). From statistical relational to neurosymbolic artificial intelligence: A survey. *Artificial Intelligence*, 328, 104062. <https://doi.org/10.1016/j.artint.2023.104062>
- Mohr, H., Eger, M., & Martens, C. (2018). Eliminating the Impossible: A Procedurally Generated Murder Mystery. *Joint Proceedings of the AIIDE 2018 Workshops Co-Located with 14th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. Retrieved June 29, 2026, from <https://ceur-ws.org/Vol-2282/>
- Okreša Đurić, B., & Schatten, M. (2016–June 3). Defining Ontology Combining Concepts of Massive Multi-Player Online Role Playing Games and Organization of Large-Scale Multi-Agent Systems. In P. Biljanović (Ed.), *39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* (pp. 1330–1335). IEEE. <https://doi.org/10.1109/MIPRO.2016.7522346>
- Poole, B., Jain, A., Barron, J. T., & Mildenhall, B. (2023). DreamFusion: Text-to-3d using 2d diffusion [arXiv:2209.14988]. *The Eleventh International Conference on Learning Representations (ICLR)*.
- Sacco, O., Liapis, A., & Yannakakis, G. N. (2016). A Holistic Approach for Semantic-Based Game Generation. *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, 1–8. <https://doi.org/10.1109/CIG.2016.7860386>
- Sacco, O., Liapis, A., & Yannakakis, G. N. (2017). Game Character Ontology (GCO): A Vocabulary for Extracting and Describing Game Character Information from Web Content. *Proceedings of the 13th International Conference on Semantic Systems*, 9–16. <https://doi.org/10.1145/3132218.3132233>
- Schatten, M., Okreša Đurić, B., Tomičić, I., & Ivković, N. (2017–June 23). Agents as Bots – An Initial Attempt Towards Model-Driven MMORPG Gameplay. In Y. Demazeau, P. Davidsson, J. Bajo, & Z. Vale (Eds.), *Advances in Practical Applications of Cyber-Physical Multi-Agent Systems: The PAAMS Collection* (pp. 246–258). Springer International Publishing. https://doi.org/10.1007/978-3-319-59930-4_20
- Shaker, N., Togelius, J., & Nelson, M. J. (2016). *Procedural content generation in games*. Springer. <https://doi.org/10.1007/978-3-319-42716-4>
- Singh, J., & Zheng, L. (2023). Divide, Evaluate, and Refine: Evaluating and Improving Text-to-Image Alignment with Iterative VQA Feedback. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, & S. Levine (Eds.), *Advances in neural information processing systems* (pp. 70799–70811, Vol. 36). Curran Associates, Inc.
- Smith, A. M., & Mateas, M. (2011). Answer Set Programming for Procedural Content Generation: A Design Space Approach. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3), 187–200. <https://doi.org/10.1109/TCIAIG.2011.2158545>
- Stockdale, A. (2016). ClueGen: An exploration of procedural storytelling in the format of murder mystery games. *Experimental AI in Games (EXAG) Workshop at AIIDE, AAAI Technical Report WS-16-22*.
- Summerville, A., Snodgrass, S., Guzdial, M., Holmgård, C., Hoover, A. K., Isaksen, A., Nealen, A., & Togelius, J. (2018). Procedural content generation via machine learning (PCGML). *IEEE Transactions on Games*, 10(3), 257–270. <https://doi.org/10.1109/TG.2018.2846639>
- Togelius, J., Yannakakis, G. N., Stanley, K. O., & 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. <https://doi.org/10.1109/TCIAIG.2011.2148116>
- Tutenel, T., Bidarra, R., Smelik, R. M., & de Kraker, K. J. (2009). Using semantics to improve the design of game worlds. *Proceedings of the Fifth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 100–105.
- Yang, Y., Sun, F.-Y., Weihs, L., VanderBilt, E., Herasti, A., Han, W., Wu, J., Haber, N., Krishna, R., Liu, L., Callison-Burch, C., Yatskar, M., Kembhavi, A., & Clark, C. (2024). Holodeck: Language guided generation of 3d embodied AI environments. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 16227–16237. <https://doi.org/10.1109/CVPR52733.2024.01536>
- Yang, Y., Wang, C., Ye, J., Li, Y., Chen, Z., Huang, Z., Mu, Y., Chen, Z., Guo, C., & Liu, X. (2026). PhysForge: Generating physics-grounded 3d assets for interactive virtual world [arXiv:2605.05163]. *Proceedings of the International Conference on Machine Learning (ICML)*.
- Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., & Stoica, I. (2023). Judging LLM-as-a-judge with MT-bench and chatbot arena [arXiv:2306.05685]. *Advances in Neural Information Processing Systems 36 (NeurIPS), Datasets and Benchmarks Track*.
- Zhu, Y., Deng, K., Fauconnier, J.-P., Navarro, I., Li, D., Pun, A., Zhang, Y., Zhuang, P., Sun, X., Agrawala, M., Bhat, K., & Zhou, T. (2026). CubePart: An open-vocabulary part-controllable 3d generator [arXiv:2605.28763]. *ACM SIGGRAPH 2026 Conference Papers*.