Interactive Self-Assembling Agent Ensembles

Samuel Truman and Sebastian von Mammen

Games Engineering, Julius-Maximilians University, Würzburg, Germany
{samuel.truman,sebastian.von.mammen}@uni-wuerzburg.de

Abstract. In this paper, we bridge the gap between procedural content generation (PCG) and user-generated content (UGC) by proposing and demonstrating an interactive agent-based model of self-assembling ensembles that can be directed though user input. We motivate these efforts by considering the opportunities technology provides to pursue game designs based on according game design frameworks. We present three different use cases of the proposed model that emphasize its potential to (1) self-assemble into predefined 3D graphical assets, (2) define new structures in the context of virtual environments by self-assembling layers on the surfaces of arbitrary 3D objects, and (3) allow novel structures to self-assemble only considering the model's configuration and no external dependencies. To address the performance restrictions in computer games, we realized the prototypical model implementation by means of an efficient entity component system (ECS). We conclude the paper with an outlook on future steps to further explore novel interactive, dynamic PCG mechanics and to ensure their efficiency.

Keywords: Procedural Content Generation · User-Generated Content · Game Mechanics · Agent-based Models · Self-Assembly

1 INTRODUCTION

As an extension to the broadly applied foundation of the MDA framework (mechanics, dynamics, aesthetics) on game design [12], the DPE framework (design, play and experience) emphasizes the technological basis of game design as it determines the potential design space in terms of user experience, gameplay, storytelling, and learning in a pedagogical sense, if applicable [22]. The technological perspective and its tight relationship with game design is especially obvious in the context of PCG, where various elements of a game immediately emerge from algorithmic instructions. In particular, PCG has been applied to small *bits* of a game such as graphical assets, behavioral descriptions or local effects, their combination into *spaces*, most prominently defined through maps, *systems* that put bits into relation (also spatially), *scenarios* that introduce gameplay challenges, *designs* that combine various elements into playable experiences, and *content* which drives the experiences [4]. More recent advances of the MDA framework and its successors, e.g. the DDE framework (design, dynamics, experience) [18] shift the focus towards the possible and emerging relationships

between the player subject and the antagonist that considers the entirety of designed aspects the player is exposed to. The essence of the DDE framework is, accordingly, the need to iteratively consider all the different, tightly interwoven design aspects to arrive at a wholistic design of an interactive experience. The resulting increase in the frequency of playtests culminates in the idea of PCG that adapts game design to specific player models in advance or during play, see e.g. [23,14,8,10].

Different from such *adaptive* PCG approaches, in this paper we consider another direction that *dynamic* PCG can take[1]. In particular, we demonstrate how ensembles [5] of agents [2] can implement rather versatile interfaces to inform dynamic PCG processes. In the following section, we refer to several concepts that motivated and approaches that we built an exemplary model on, which we detail in Section 3. We showcase its capabilities and discuss its limitations. We conclude this paper with a short summary and considerations for future work.

2 RELATED WORK

PCG refers to algorithmically generated content [13], whereas UGC refers to content that platform users create, a.o. players of virtual worlds [7]. Such binary distinction quickly fades away when looking at crafting systems which provide mechanics for players to proactively contribute to the generation of game-related assets [3]. In [6], a 5-type taxonomy for crafting systems is introduced: type 1 implies that certain resources are traded for items (or upgrades of items) which could be reduced to the use of currency. Type 2 requires the player to uncover and follow a specific recipe to yield certain items. Type 3 crafting systems allow the player to explore the space of possible (pre-defined) recipes by themselves, whereas type 4 systems allow the player to make several choices which result in custom items, e.g. by combining different materials. Finally, type 5 are considered "true" crafting systems in the sense that minute details that the player determines interact with each other in a complex manner. Type 5 systems challenge the player in acquiring the knowledge how these interactions play out and in crafting solutions that meet some given requirements. As type 5 crafting systems generate outcomes based on computing the interplay of some inputs, they also qualify as procedural systems, bridging the gap between UGC and PCG, and potentially elevating "the crafting system to being the primary game mechanic" [6].

Arbitrary generative models can serve as a basis for systems at the boundary between UGC and PCG. Individually designed components might, for instance, *self-assemble* "into patterns or structures without human intervention" [21]. Simple models of self-assembly can be implemented by modeling virtual agents [2] that attract each other and stick together (*static* self-assembly), yielding higher-level artifacts. These artifacts may exhibit properties that *emerge* from the in-

[1] We refer to dynamic PCG to particularly highlight the need and capability to procedurally generate on-the-fly to yield new solutions dynamically as, for instance, pursued in [24,20,19].

terplay of the self-assembled *ensemble* of lower-level agents, i.e. that none of the lower-level agents exhibit by themselves [5]. In this paper, we present an interactive generative model that implements according self-assembling agent ensembles. We do not focus on the specific requirements of the user interface which arise when instructing large numbers of self-organizing agents [16] but rather on a description of the model and its implementation concept to achieve real-time performance.

One of the authors previously presented works in which agent ensembles were evolved and guided to grow three-dimensional structures based on tracing the agents' trajectories, whereas some agents proliferated to yield branching structures [9] and others concerted their flight to generate braids [17]. These approaches featured *reactive* agents [15] that preform actions based on their perceived environment and internal states. In particular, they extended the original "boids" flocking model based on neighboring agents' states [11] in terms of the aforementioned structural trace and behavioral augmentations. Reactive agents have also been used to interactively design large biomolecular models [1].

3 MODEL

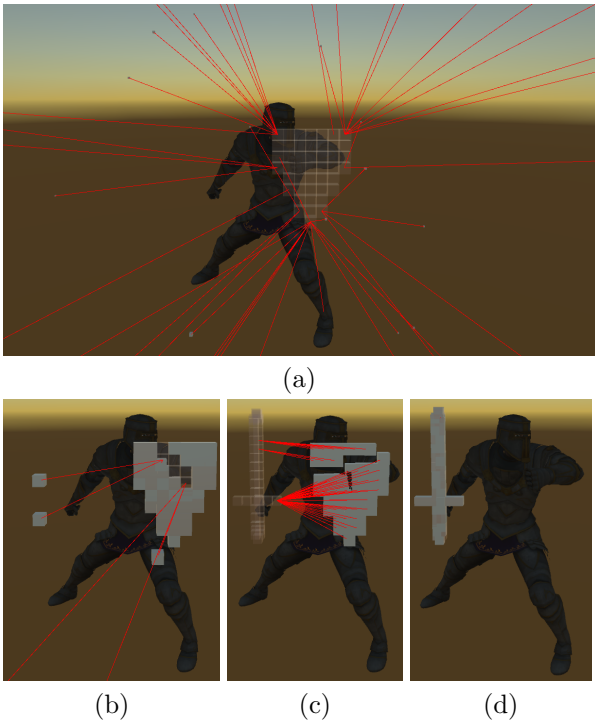


Fig. 1. (a) Agent ensembles that are distributed across the scene (b) assemble into a shield, and (c) transition into (d) a sword.

In the preceding sections, we elaborated about PCG in the context of game design and motivated agent-based PCG mechanics directly made available to the player. To further this concept in the context of computer games, we implemented a simple agent-based model that can implement rules of self-assembly but can also be guided by the player subject or the antagonist (following the terminology of the DDE framework [18]). We realized this by agents (represented as grey cubes) that are stationary and exhibit ports (semi-transparent grey cubes) for other mobile agents to lock on. In the mobile state, an agent does not exhibit ports but seeks

them out within its vicinity to proactively come into port. Agents will move to their target position in a straight line and are allowed to pass through each other. Currently, all agents share the same appearance, dimensions and speed. While several agents may move toward the same port, agents encountering an occupied port either stop moving or choose an alternative destination. Spatial conflicts are resolved by automatic offsets to the side.

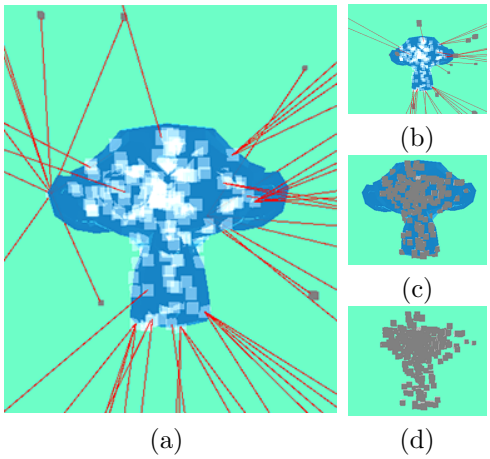
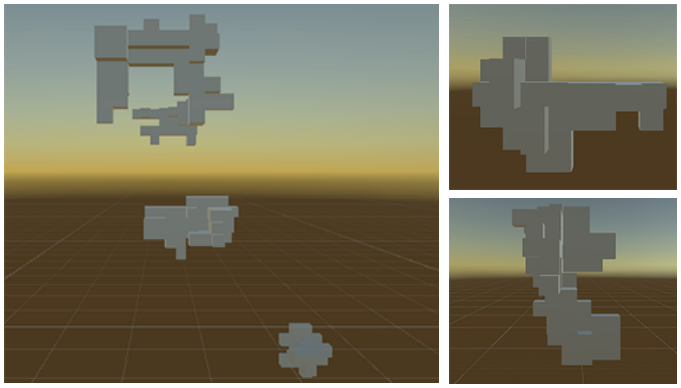


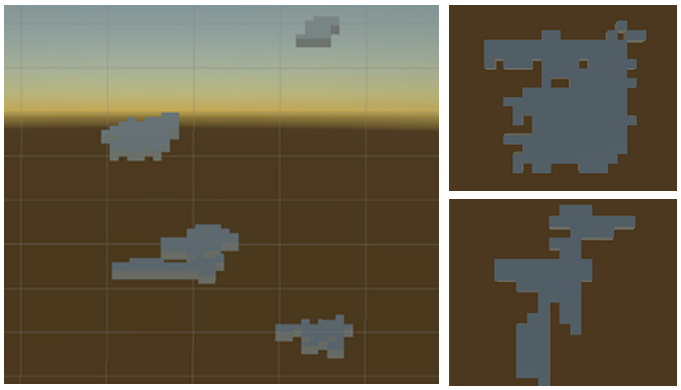
Fig. 2. The agents lock onto the closest ports that were randomly generated on the surface.

We implemented three concrete model instances, each promoting a different degree of user interaction, to demonstrate some of the capabilities of our model: Use case 1 shows how the agents can assemble into different assets over time. Use case 2 shows how the agents can dynamically occupy a surface. Use case 3 highlights the agents' capacity to self-assemble without the need for a shape that provides context. At the beginning of each assembly process, the available agents are randomly distributed within a well-defined box that is centered at the origin of the scene. The initial distribution of agents can have a major impact on the outcome as their placement determines their locally perceived information. Figure 1 shows use case 1, i.e. how the agents first assemble into a warrior's shield and transition into his sword. The shapes' port information is precomputed, whereas the agents' destinies are computed during runtime, once the player triggers the assembly of one or the other shape.

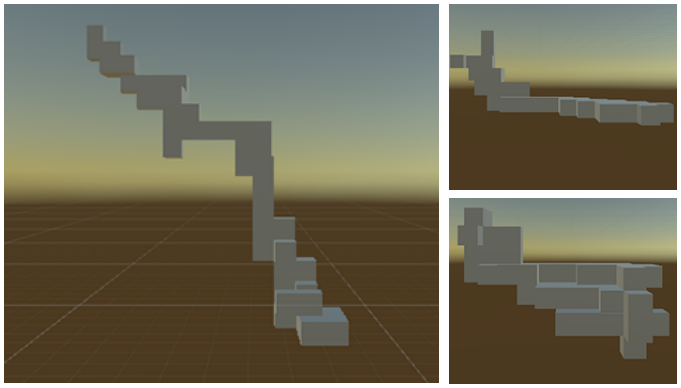
Use case 2 is demonstrated in Figure 2. It shows how an assembly instruction is dynamically applied to the environment without pre-computation: The user designates an object which triggers the random generation of ports on its surface and the agents' embarkment. We consider the relative surface area of each triangle when calculating the probability of its occupation. However, complex meshes with double-sided faces or inner faces might skew the odds. Figure 3 displays outcomes of use case 3, i.e. self-assembly without contextual port definitions as in use cases 1 and 2. Rather, the agents stick together and grow into clusters by themselves. With a given probability, each agent determines whether to remain mobile or switch into the stationary state at the beginning of the simulation. The mobile agents dock onto free ports based on proximity, turn static and open up new ports themselves. The three examples in Figure 3 have been obtained by different rule sets that specify port availability.



(a)



(b)



(c)

Fig. 3. (a) Agent ensembles can dock onto free ports without restrictions. (b) Port generation by stationary agents is restricted to two dimensions. (c) Agent ensembles can only dock to the most recently generated ports.

In the first example (Figure 3(a)), stationary agents open up ports on each vacant side and, as a consequence, mobile agents are not limited to grow the cluster into any direction. In the second example (Figure 3(b)), only ports in the xy-plane are generated by the stationary agents which limits the growth to flat platforms. In the third example (Figure 3(c)), mobile agents can only dock onto ports opened by those agents that docked onto a cluster most recently. All the ports previously opened by agents that had docked on before, but one random one, are removed.

Efficiency is a great challenge when considering interactive PCG in the context of games considering the multifaceted intertwined processes that make for a proper, immersive experience. Therefore, we designed our implementation accordingly and built it on top of Unity's Data Oriented Tech Stack (DOTS). It combines an entity component system (ECS), a job system, and a performant compiler especially to promote large numbers of interacting entities. Implementing the presented agents as entities, each one is composed of a set of uniquely identifiable data components, which can be iterated at high speed, especially due to a systematic avoidance of cache misses.

4 SUMMARY AND FUTURE WORK

We motivated the use of interactive, dynamic PCG as a play mechanic based on broad perspectives of game design. Based on examples of agent ensembles that have previously been presented in interactive generative and simulative contexts, we proposed a model of agent ensembles that self-assemble based on player input. We demonstrated three according use cases where concrete model instances lead to self-assembly of different structures constrained by docking rules (use case 3), to agent coverage of arbitrary surfaces to highlight the flexibility of the model in a given virtual environment context (use case 2), and to player-triggered dynamic transformation of the ensemble into different predefined assets. We addressed the high performance requirements of interactive, dynamic PCG by realizing our model by means of an efficient ECS system.

In order to push interactive PCG in games, further interactions and especially their impact on play need to be explored. To this end, small whitebox prototypes could unearth some new directions where such game elements could lead. Models, technological realization and game mechanics should be systematically embedded into game design frameworks such as MDA, DPE or DDE to better explore their impact and the space for design opportunities they unfold. With the first steps of "true" design capabilities of interactive PCG—analogous to the quote about "true" crafting systems in Section 2—larger cascades and cycles of interlocking mechanics should be considered at a systematic, abstract level as well. For instance, based on the model instances presented in this paper, one could investigate the impact of dynamic assets on ecosystems of online multiplayer platforms that might correlate to actual processing and storage capacities. Or, as another example, one could investigate the physical impact of

user-created assets in the context of competitive fighting, racing or sports games, etc.

As mentioned above, the performance requirements are also high, both in terms of real time capabilities at small scopes of interactive, dynamic PCG or, if those can be ensured, for scaling up the scope, e.g. the numbers of agents and their capabilities. To this end, various development challenges can be overcome. For instance, our implementation could have harnessed the power of the utilized ECS framework more rigorously by identifying and setting up more jobs that can be run in parallel or by using ECS during the initialization step or when novel agents are spawned during runtime. Well-established acceleration algorithms such as spatial data structures could further reduce the number calculations to determine interactions. The performance impact of structural changes, i.e. changing an entity's archetype by adding or removing components, should be evaluated. In case structural changes are too costly, boolean flags might be used.

References

1. Davison, T., Samavati, F., Jacob, C.: Lifebrush: Painting interactive agent-based simulations. In: 2018 International Conference on Cyberworlds (CW). pp. 17–24. IEEE (2018)
2. Denzinger, J., Kordt, M.: Evolutionary on-line learning of cooperative behavior with situation-action-pairs. In: ICMAS. pp. 103–110 (2000)
3. Francis, B.: 7 crafting systems game designers should study. Online at `https://www.gamasutra.com` (July 2015)
4. Hendrikx, M., Meijer, S., Van Der Velden, J., Iosup, A.: Procedural content generation for games: A survey. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) **9**(1), 1–22 (2013)
5. Jun, H., Liu, Z., Reed, G.M., Sanders, J.W.: Ensemble engineering and emergence. In: Software-Intensive Systems and New Computing Paradigms, pp. 162–178. Springer (2008)
6. King, A.: 5 approaches to crafting systems in games (and where to use them). Online at `https://gamedevelopment.tutsplus.com` (January 2015)
7. Lastowka, G.: User-generated content and virtual worlds. Vand. J. Ent. & Tech. L. **10**, 893 (2007)
8. Lopes, R., Hilf, K., Jayapalan, L., Bidarra, R.: Mobile adaptive procedural content generation. In: Proceedings of the fourth workshop on Procedural Content Generation in Games (PCG 2013), Chania, Crete, Greece (2013)
9. von Mammen, S., Jacob, C.: Genetic swarm grammar programming: Ecological breeding like a gardener. In: Srinivasan, D., Wang, L. (eds.) CEC 2007, IEEE Congress on Evolutionary Computation. pp. 851–858. IEEE Press, Singapore (2007)
10. Oliveira, S., Magalhães, L.: Adaptive content generation for games. In: 2017 24º Encontro Português de Computação Gráfica e Interação (EPCGI). pp. 1–8. IEEE (2017)
11. Reynolds, C.W.: Flocks, herds, and schools: A distributed behavioral model. Computer Graphics **21**(4), 25–34 (1987)

12. Robin Hunicke, Marc LeBlanc, R.Z.: Mda: A formal approach to game design and game research. In: Proceedings of the AAAI-04 Workshop on Challenges in Game AI. pp. 1–5 (2004)
13. Shaker, N., Togelius, J., Nelson, M.J.: Procedural Content Generation In Games. Springer (2014)
14. Shaker, N., Yannakakis, G.N., Togelius, J.: Towards player-driven procedural content generation. In: Proceedings of the 9th conference on Computing Frontiers. pp. 237–240 (2012)
15. Stuart, R., Peter, N.: Artificial Intelligence: A Modern Approach. Prentice Hall, Upper Saddle River, NJ, 4th edn. (2020)
16. von Mammen, S.: Self-organisation in games, games on self-organisation. In: Games and Virtual Worlds for Serious Applications (VS-Games), 2016 8th International Conference on. pp. 1–8. IEEE (2016)
17. Wagner, D., Hofmann, C., Hamann, H., von Mammen, S.: Design and exploration of braiding swarms in vr. In: Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology. p. 13. Gothenborg, Sweden (November 2017)
18. Walk, W., Görlich, D., Barrett, M.: Design, dynamics, experience (dde): an advancement of the mda framework for game design. In: Game Dynamics, pp. 27–45. Springer (2017)
19. Washburn, M., Khosmood, F.: Dynamic procedural music generation from npc attributes. In: International Conference on the Foundations of Digital Games. pp. 1–4 (2020)
20. Wheat, D., Masek, M., Lam, C.P., Hingston, P.: Dynamic difficulty adjustment in 2d platformers through agent-based procedural level generation. In: 2015 IEEE International Conference on Systems, Man, and Cybernetics. pp. 2778–2785. IEEE (2015)
21. Whitesides, G.M., Grzybowski, B.: Self-assembly at all scales. Science **295**(5564), 2418–2421 (2002)
22. Winn, B.M.: The design, play, and experience framework. In: Handbook of research on effective electronic gaming in education, pp. 1010–1024. IGI Global (2009)
23. Yannakakis, G.N., Togelius, J.: Experience-driven procedural content generation. IEEE Transactions on Affective Computing **2**(3), 147–161 (2011)
24. Zook, A., Riedl, M.: A temporal data-driven player model for dynamic difficulty adjustment. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. vol. 8 (2012)