Florian Mehm, Christian Reuter, Stefan Göbel, Ralf Steinmetz: Future Trends in Game Authoring Tools. In: Marc Herrlich, Rainer Malaka, Maic Masuch: *Entertainment Computing - ICEC 2012*, no. LNCS 7522, p. 536-541, Springer, September 2012. ISBN 978-3-642-33541-9.

Future Trends in Game Authoring Tools

Florian Mehm, Christian Reuter, Stefan Göbel, Ralf Steinmetz

Technische Universität Darmstadt, Multimedia Communications Lab (KOM)
Rundeturmstrasse 10
64283 Darmstadt, Germany
{florian.mehm, christian.reuter, stefan.goebel,
ralf.steinmetz}@kom.tu-darmstadt.de

ABSTRACT. Authoring Tools for digital games are used to create games from scratch, integrate content and game mechanics easily and can assist in a multitude of ways in the production chain of a game. For example, they can allow non-programmers to work on the game logic by means of domain-specific or visual programming languages, increase the collaboration between team members by integrating computer-supported collaborative work techniques, assist in catching errors in the game by model checking and offer publishing to multiple platforms by saving games in an intermediate format which can be run on various systems. This already interesting and viable approach can be extended in a number of ways which we exemplify in this position paper to indicate possible future directions for game authoring tools.

1 INTRODUCTION

The concept of providing an authoring tool that streamlines the workflow in a certain area (may it be multimedia [1], e-Learning [2] or more recently games [3, 4]) has a long history in computing. It is successful due to several factors: it allows non-technical users to work on projects that would otherwise be out of their reach (due to lack of expertise, especially concerning programming languages); it can bring structure into unstructured domains (such as game development) and it can speed up development by streamlining and automating common tasks. Authoring tools also reduce coordination effort between different groups involved in the game production process such as writers and artists. They separate themselves from level editors in most commercially available games as they support more development tasks up to the creation of a complete game.

In the field of game authoring tools, several examples can be seen, each focusing on a specific approach to authoring. The e-Adventure authoring tool is specialized for the creation of educational adventure games [3]. The SHAI Scenario Editor described by Est et al. [5] allows users to define educational scenarios using high-level logic visualized in diagrams that is converted to low-level logic internally by the system. The Alice [6, 7] tool uses a visual programming language to allow users to program their own game and learn about programming concepts simultaneously. The authors of this paper have provided the authoring tool StoryTec [4] targeted at the creation of

Serious Games, combining several positive properties also found in the projects mentioned here, such as a visual programming approach and high-level logic for authors.

In the following, we focus on possible areas of innovation where authoring tools can be extended and thereby improved in the future. This includes the extension into the content generation process as well as domain specific languages and other novel authoring concepts that can make authoring tools more expressive without increasing their complexity for the user. The concept of model checking can assist users by detecting errors automatically. We also discuss multiplayer games, an area largely ignored by current authoring tools, and describe ideas on how the authoring process itself can be enhanced by using game concepts.

2 Areas of Innovation

2.1 Procedural Content Generation

While authoring tools have been demonstrated to be able to assist (non-technical) users in adding content to games, the next possible step is to assist users in creating content by the provision of procedural content generation tools.

Work in the field of procedural content generation for games has intensified in the last years [8]. Examples for this are systems creating infinite levels for platform games in the style of Nintendo's Super Mario Bros [9]. These systems aim at generating as many aspects of a game as possible (including the game's narrative[10] or rules [11] in some cases).

The vision of Procedural Content Generation is to allow users to enter a semantic description and receive a fitting assets, may it be a virtual character, 3D object or piece of music. Currently, research into authoring tools and procedural content generation is still largely separated, but the combination of both could help the work of authors tremendously by speeding it up and reducing the number of necessary experts in the production of a game significantly. Other approaches could reach similar results by using methods for semantic content retrieval, giving access to content already created for other applications. Among the possible challenges of this approach is the way in which the semantic description for the object under creation is to be given (as parameters in a restricted domain, as natural language, etc.) and if and how authors can manipulate the object after it has been created.

2.2 Domain-Specific Languages & Visual Languages

While domain-specific and visual programming languages are already used in current authoring tools, their leverage can be extended by increasing their expressiveness and applicability to game development. Examples of visual programming approaches can be found in the previously mentioned e-Adventure, Alice and StoryTec systems. A text-based domain-specific language for games is the Inform 7 language for the creation of interactive fiction [12]. Especially the latter language shows the elegance possible with this approach: reading the source files of a game created with it makes im-

mediate sense to a person able to understand English, while being parseable by the computer at the same time.
Especially model-driven and component-based software engineering approaches fit together well with authoring tools, as they abstract from implementation details and offer users only the level of semantics and detail they want to work with. BinSubaih and Maddock [13] describe a system that separates a game's core logic from the actual implementation in a game engine. The EGGG language [14] is designed to describe games on a very high level of abstraction, realizing them using a set of re-usable software components.
Potential difficulties with Domain-Specific Languages in game authoring tools result from the definition of these languages themselves, including the scope as well as the granularity of the language constructs, possibly resulting in too restrictive languages on the one hand and too unwieldy languages for practical use on the other.

2.3 Novel Authoring Concepts

Departing from the paradigms of standard e-Learning or multimedia authoring tools such as timelines or 2D layouts, new authoring concepts can be embraced. A strong example is the block-based editing paradigm successfully used by the independent game Minecraft[1] [15], which has spawned whole worlds of user-created 3D models without the need for 3D modeling know-how.

By switching to such simple authoring concepts, novice authors and authors without technical expertise (commonly the main target users of authoring tools) have a very low threshold of learning compared to traditional 3D authoring tools, which require careful manipulation of the mouse and keyboard and complicated concepts such as cameras, lights and 3D meshes made of vertices and polygons.

2.4 Model Checking

Apart from streamlining and structuring content and game logic, authoring tools can support users in more ways by offering automatic checks for syntactical and especially semantic errors. An example can be found in the model checking capabilities described in the context of the e-Adventure authoring tool [16], which can inform users of potential problems in the game. Future versions of such mechanisms could employ methods of artificial intelligence such as machine learning in order to detect semantic problems in authored games and make the author aware of them, potentially even providing automatic correction advice. For example, an educational game authoring tool might alert the user whenever a certain learner type has been neglected in the game or whenever a dead end (a section of the game where no more progress can be made due to conflicting or missing game logic) has been created.

[1] Available from http://minecraft.net

2.5 Multiplayer Authoring

By the increasing number of (massively) multiplayer and social games, it can be seen that multiplayer games are becoming a larger and larger aspect of the game industry. Authoring tools can support this sector by making multiplayer mechanics (synchronization of players, collaboration, etc.) evident and manageable to authors.

This field of authoring has been largely neglected so far. In recent work, the authoring tool StoryTec has been extended for the authoring of multiplayer games, yielding a first prototype in this field[17]. It was then used to create an adventure game being playable for two players. The early state of research in this field shows that especially the puzzle design has to be supported in the authoring tool by offering mechanisms to synchronize or separate players. As an example from the associated demo, one player is set in the authoring tool to be restricted to one part of the game world due to being pinned down by a fallen-down tree, while the other player has to move around and find a way to free him or her. However, the first player is the only one in possession of a map, forcing the two players to collaborate. This interaction has been set up in the authoring tool by a set of constraints and conditions, showing a first version of this area of authoring tool improvement.

The aspect of adaptivity (based for example on player or learner modeling) found in current authoring tools such as StoryTec or e-Adventure will increase in complexity but also in potential reward in terms of added enjoyment and effectiveness by the addition of multiple players. Whereas in the single player case the game only has to be adapted for the needs of a single player, in the multiplayer case, the adaptation has to happen to each single player as well as for the whole group at the same time. A first approach including automatic and manual adaptation by a Game Master is shown by Wendel et al. [18].

2.6 In-Game Editing, Gamification and User-Generated Content

By dropping the classical authoring tool interface and allowing editing directly inside the game, users can easily work on the game while receiving immediate feedback. This concept has been touched upon by platforms such as Jumala[2] or games like Little Big Planet [19], as well as the already mentioned Minecraft. At the same time, this can be leveraged to incentivize user-generated content by means of gamification, giving players in-game rewards like achievements or items for building new content and extending games.

Again, this step lowers the threshold for novice authors tremendously by offering the authoring capabilities directly in the game environment. Usual authoring tools for games rarely have a true WYSIWYG (What you see is what you get) interface; instead, they rely on abstractions in order to maintain the overview for the author. With in-game editing, authors can immediately see the effects of their changes and try them out in the exact same context as their players. The closest authoring tools have gotten

[2] Available at http://jumala.com

to this are systems like CryTek's Sandbox editor[3] or the Unity game editor[4], which allow starting the game right inside the editor interface for testing purposes.

Problems that are commonly associated with User-generated content have to be addressed in the context of authoring tools as well, namely those of copyright infringements, quality control and management and retrieval of a possibly very large body of generated content [20] .

3 Conclusion

In the previous sections, we have outlined a set of areas where future authoring tools can go beyond the current state of the art, opening up authoring tools to new groups of users and improving the strong aspects of authoring tools further. These areas should not be seen in isolation. Instead, the combinations of areas can prove worthwhile and add new strengths of their own while mitigating the drawbacks of individual approaches. For example, in-game editing and procedural content generation in combination yield vast worlds that authors can change on the fly, without the need to create the basic layout of the world first. A large part of making multiplayer authoring feasible will probably lie in model checking, since the addition of more than one player in the game increases the space of possible actions and combinations of variables in the game exponentially, making it a hard problem for humans to keep all logical and semantic constraints in view at the same time.

At the same time, critical questions also appear due to these innovations. For example, procedural content generation could be seen as hindering the free creativity inherent in games. Automatic retrieval for content aggregation or user-generated content might give rise to copyright issues. It is also questionable whether the content produced by these methods yields enough quality compared to assets created by professional artists. We therefore see the further evaluation of these ideas as future work.

References

1. Bulterman, D.C.A., Hardman, L.: Structured multimedia authoring. ACM Transactions on Multimedia Computing Communications and Applications. 1, 89-109 (2005).
2. Cristea, A.: Authoring of adaptive educational Hypermedia. Seventh IEEE International Conference on Advanced Learning Technologies ICALT 2007. 943-944 (2007).
3. Torrente, J., Blanco, Á.D., Cañizal, G., Moreno-ger, P., Fernandéz-Manjón, B.: <e-Adventure3D>: An Open Source Authoring Environment for 3D Adventure Games in Education. ACE '08 Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology. ACM, New York (2008).
4. Mehm, F.: Authoring Serious Games. In: Pisan, Y. (ed.) Proceedings of the Fifth International Conference on the Foundations of Digital Games - FDG '10. pp. 271-273. ACM (2010).

[3] See http://mycryengine.com/
[4] See http://www.unity3d.com

5. Est, C.V., Poelman, R., Bidarra, R.: High-Level Scenario Editing for Serious Games. In: Richard, P. and Braz, J. (eds.) GRAPP 2011 - Proceedings of the International Conference on Computer Graphics Theory and Applications. pp. 339-346. SciTePress (2010).
6. Villaverde, K., Jeffery, C., Pivkina, I.: Cheshire: Towards an Alice Based Game Development Tool. International Conference on Computer Games, Multimedia & Allied Technology. pp. 321-328 (2009).
7. Kelleher, C.: Motivating programming: using storytelling to make computer programming attractive to middle school girls, (2006).
8. Hendrikx, M., Meijer, S., Velden, J.V., Iosup, A.: Procedural Content Generation for Games : A Survey. In Press: ACM Transactions on Multimedia Computing, Communications and Applications. 1, 1-24 (2012).
9. Shaker, N., Togelius, J., Yannakakis, G.N., Weber, B., Shimizu, T., Hashiyama, T., Sorenson, N., Pasquier, P., Mawhorter, P., Takahashi, G., Smith, G., Member, S., Baumgarten, R.: The 2010 Mario AI Championship : Level Generation Track. IEEE Transactions on Computational Intelligence and AI in Games. 3, 1-16 (2011).
10. Cavazza, M., Lugrin, J.-L., Pizzi, D., Charles, F.: Madame bovary on the holodeck: immersive interactive storytelling. Proceedings of the 15th international conference on Multimedia MULTIMEDIA 07. pp. 651-660. ACM (2007).
11. Smith, A.M., Mateas, M.: Variations Forever : Flexibly Generating Rulesets from a Sculptable Design Space of Mini-Games. IEEE Conference on Computational Intelligence and Games (CIG). pp. 273-280. Ieee (2010).
12. Reed, A.: Creating Interactive Fiction with Inform 7. Course Technology PTR (2010).
13. BinSubaih, A., Maddock, S.: Game portability using a service-oriented approach. International Journal of Computer Games Technology. 2008, (2008).
14. Orwant, J.: EGGG: Automated programming for game generation. IBM Systems Journal. 39, 782-794 (2000).
15. Duncan, S.C.: Minecraft, Beyond Construction and Survival. Well Played v1. 9-22 (2011).
16. Moreno-Ger, P., Fuentes-Fernández, R., Sierra-Rodríguez, J.-L., Fernández-Manjón, B.: Model-checking for adventure videogames. Information and Software Technology. 51, 564-580 (2009).
17. Reuter, C., Wendel, V., Göbel, S., Steinmetz, R.: Multiplayer Adventures for Collaborative Learning With Serious Games. Accepted at 6th European Conference on Games Based Learning (2012).
18. Wendel, V., Hertin, F., Göbel, S., Steinmetz, R.: Collaborative Learning by means of Multiplayer Serious Games. In: Luo, X., Spaniol, M., Wang, L., Li, Q., Neidl, W., and Zhang, W. (eds.) Proceedings of ICWL 2010. pp. 289-298. Springer (2010).
19. Westecott, E.: Crafting Play: Little Big Planet. Loading. 5, 90-100 (2011).
20. OECD: Participative Web and User-Created Content. OECD Publishing, Paris (2007).