Florian Mehm, Stefan Göbel, Ralf Steinmetz: I*ntroducing Component-Based Templates into a Game Authoring Tool*. In: Dimitris Gouscos, Michalis Meimaris: *5th European Conference on Games Based Learning*, p. 395 - 403, Academic Conferences Limited, Reading, UK, October 2011. ISBN 978-1-908272-19-5 CD.

Introducing Component-Based Templates into a Game Authoring Tool

Florian Mehm, Stefan Göbel, Ralf Steinmetz
Multimedia Communications Lab, Technische Universität Darmstadt
Florian.Mehm@KOM.tu-darmstadt.de
Stefan.Goebel@KOM.tu-darmstadt.de
Ralf.Steinmetz@KOM.tu-darmstadt.de

Abstract:

Serious Games, including educational games, can be created based on various business models. Commercial Serious Game developers as well as non-professionals (teachers, employees) tasked with creating a game face significantly smaller budgets available for such games compared to games for entertainment. A use case is epitomized by a teacher investigating how he or she can create a small digital educational game for use in the classroom. The exemplified group of users does not have programming skills or experience in game development.
Authoring tools for games, analogously to authoring tools in the field of e-learning, allow a non-technical expert to compose content (such as images, sounds or videos) into a game by specifying the kind of game to create and then integrating the content. An example is the authoring tool StoryTec previously presented by the authors.
In this paper, an extension to the authoring tool StoryTec is described, which is intended to assist non-programmers and novices by offering specialized authoring functionality for the integration of content into gameplay templates. These templates are based on the software engineering paradigm of component-orientation, resulting in a number of rich templates that are offered to the author during the game creation workflow in order to facilitate the authoring process. They provide authoring capabilities specialized for certain highly dynamic or complex gameplay types, which would be cumbersome or impossible to author using general-purpose authoring functionalities also found in StoryTec. In further steps, the components can be augmented with more information, for example metadata about their usage in order to help authors understand them better or wizards which assist novice authors in correctly and efficiently filling out the gameplay templates.
A first implementation of a gameplay template was chosen based on the analysis of the authoring process of a “city rallye”-type game which involves learning about the target city in order to advance the game. This template was implemented and the authoring effort as well as the complexity were evaluated, yielding in the result that an equivalent game (with the same content) could be authored in a smaller amount of time and with reduced complexity.

Keywords: Component-based Software Engineering, Authoring Tool, Serious Game

1. Introduction

The developers of Serious Games, including the large field of Digital Educational Games (DEGs), are often faced with problems such as limited budgets or highly interdisciplinary work due to the required interaction between game designers, domain experts and game programmers. These problems are commonly more pronounced than in the production of regular entertainment games, since the market for Serious Games is smaller while the requirements for content and educational design are higher in comparison.
We have previously presented the authoring tool StoryTec (Mehm et al., 2009; Mehm, 2010) as a tool which can be used to create Serious Games more efficiently by allowing non-technical experts (especially those without the ability to program using traditional programming languages) to work directly on a game and enter content according to their expertise in the authoring tool.
In this paper, we describe a concept and implementation of the introduction of component-based templates into this game authoring tool. While component-based software engineering (CBSE) is a practice often found in the games industry, we here focus on its usage in an authoring tool, highlighting the different means of author support that become possible by adopting the CBSE approach in an authoring tool. In the following we use the term “gameplay template” for a component which the StoryTec authoring tool integrates at runtime and which encapsulates a certain type of gameplay which can be configured and filled with content by an author. A non-digital analogy are the “Frame Games” described by Stolovitch and Thiagarajan (1980): board games which are designed so that they can be filled with arbitrary educational content (for example by putting it on blank playing cards).

A use case for the concept can be found in a project recently carried out by the authors of this paper in which a virtual city rallye for the city of Darmstadt was developed using StoryTec. The gameplay is centered around a thriller story in which the player has to move around town (visualized by videos), collect knowledge and answer questions. The finale of the game is a "word puzzle" in which the player has to arrange letters in the right order to form the code word (see Figure 1). This mini-game was originally built using multimedia primitives available to the author in StoryTec, which required a large amount of images of letters and some programming using the visual programming approach used in StoryTec. In sections 4 and 5 we show how the concept is applied to this use case and how it was implemented, resulting in easier and faster authoring of this game.

Figure 1: The simple use case chosen for this paper: The gameplay template in question models the gameplay of a "word puzzle": by clicking on each letter, the user can cycle through different letters in order to enter the correct codeword.

2. Related Work

Component-Based Software Engineering (CBSE) (see Szyperski, 1997) is a common practice in software development and is based on the idea of splitting up a system into maximally independent components which are developed to carry out clearly defined tasks. These components have several positive features than can be utilized:

- **Well-defined interfaces**: Components commonly feature a strong separation between the definition of their interface and the implementation of this interface inside the component
- **Self-describing**: By using a standardized format to describe their interfaces, components describe their functionality themselves
- **Separation of concerns**: Apart from communication with other components via defined interfaces, components are able to complete all their tasks on their own
- **Substitutability**: Since interfaces for components are separated from the actual implementation inside the component, they are substitutable for another component implementing the same interface. In game development, this can be used to produce different versions for different platforms, allowing portability
- **Tool support**: Using tools developed for this purpose, users without a programming background can combine components by manipulating graphical representations of the components, for example by connecting the output of one component with the input of another component

In the development of games and especially of game engines, CBSE has been used to some extent. Research results of applying CBSE to the development of games include the work of Folmer (2007), in which a general reference architecture for game engines based on components is described.
Specifically for the development of educational games, Maciuszek et al. (2010) show how an intelligent tutoring system centred around component-orientation is used in educational games. Similarly, Bisognin et al. (2010) demonstrated a learning environment which integrates components which encapsulate parts of the game. In conjunction with this system, the EDoS authoring system presented by Tran et al. (2010) targets this game architecture.
An authoring tool designed to be running on a very minimal mobile system (Nintendo DS) is the e-Training DS system (Tornero et al., 2010). This tool incorporates the idea of component-orientation in an authoring tool in a similar way as presented here.

3. StoryTec

In this section, we give a short and concise overview of the authoring process in StoryTec as far as it relates to the content of this paper. For more details, see (Mehm, 2009, 2010) as well as the website of the authoring tool[1].

Figure 2: The user interface of StoryTec with the main editor windows visible (from bottom left in clock-wise order): Story Editor, Stage Editor, Objects Browser and Properties Editor.

An author in StoryTec usually starts by defining the overall structure of the game as well as the game's story by entering it in an abstract fashion in the Story Editor window of StoryTec (see Figure 2). In this editor, the whole game is visualized by boxes representing scenes (parts of the game) as well as transitions between scenes visualized by arrows connecting the scenes.

When this structure has been laid down, the author can continue with filling the game with content by choosing a scene to work on and switching to the Stage Editor window, which is a WYSIWYG-editor showing the objects currently placed inside the scene. Objects can be dragged from the Object Repository window to the scene and then be configured using the Property Editor window. Objects include multimedia primitives such as images, sounds, videos, characters or label texts. Objects added in the Stage Editor are also indicated by icons which are placed in their respective scene boxes.

In a final step, the interaction between the game and the player has to be defined. For this, StoryTec utilizes a visual programming approach found in the ActionSet Editor window, in which the reaction of the system to an event generated by the player is defined. This again uses boxes to represent actions such as playing a video, having a character perform a speech act or changing the scene. The boxes are aligned in a flow diagram-like structure allowing branches based on variables such as the score of the player or the number of previous attempts at a certain task.

This general workflow is currently being changed towards specialized workflows for different experts. For example, a content provider would be unable to change the overall structure and instead be confined to just adding the content they have available.

[1] http://www.storytec.de

4. Concept

In the following, a concept for the integration of component-based game templates into the authoring tool StoryTec is described. Each component captures a certain kind of gameplay to be found in the final game. The individual aspects the components provide are shown in Figure 3 and will be described in the following sections.

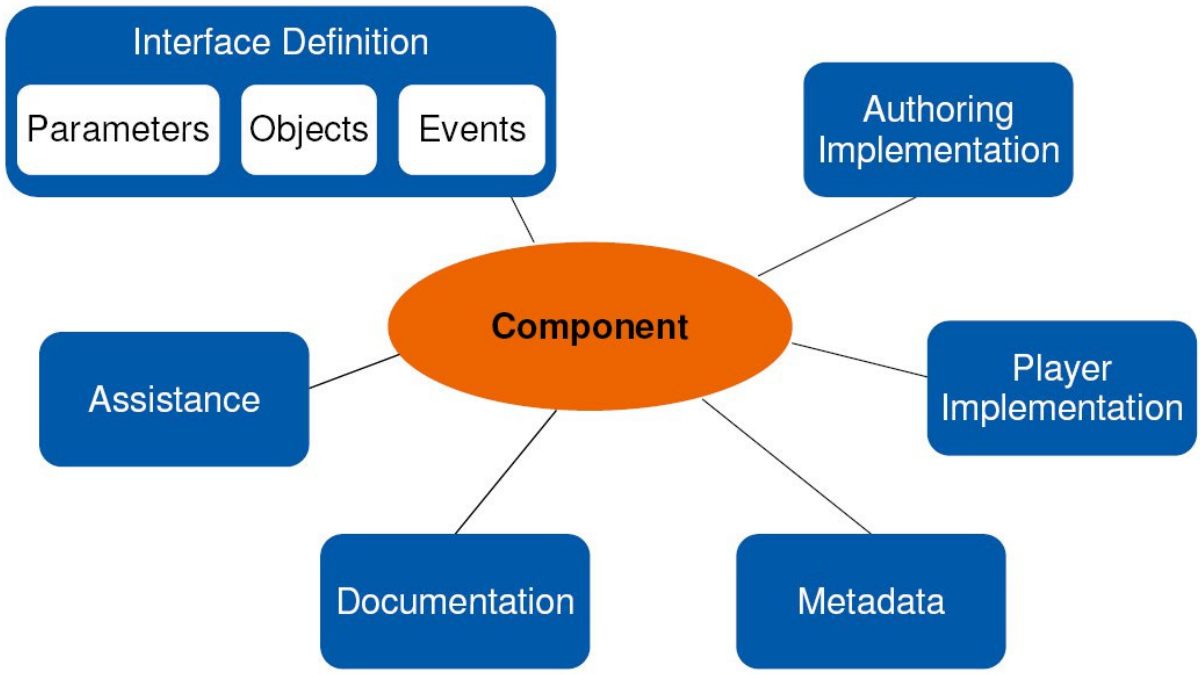

Figure 3: An overview of component-based gameplay templates as described in section 4. A component consists of an implementation (for the authoring tool and for each player application/platform it will run in) and is described by metadata including an interface definition and documentation. Finally, the authoring implementation can also provide specific assistance to the author e.g. in the form of software wizards.

4.1 Interface Definition

Each component is equipped with a description of the available ways to configure it. The first major part of this definition are parameters (in the use case, parameters might include the correct code word and the difficulty of the task). The second are Objects in the sense used in StoryTec, denoting content such as images as well as logical structures such as variables. For the use case of the word puzzle game, content could be an optional background image. Finally, the interface definition includes the events generated by the component, such as events generated when a task is successfully or unsuccessfully finished by the user. Authors can use these events to program the reactions of the overall game, e.g. increase a score or go to a different scene in the game.
This interface definition is beneficial in several areas. Programmers implementing the component are given a specific set of requirements by having to implement the interface definition exactly as defined. In the authoring tool, the interface definition can be used to generate documentation or wizards automatically. For example, if the component features a slot for a background image, the authoring tool could ask a novice user in a wizard to specify a background image.

4.2 Implementation

Each component has to be implemented for a specific player application running on a target platform. This implementation is provided with the configuration (parameter values and objects) specified by the author in the authoring tool and consists of the actual implementation of the gameplay. Implementations for different target platforms can provide different input mechanisms or make other adjustments to adapt to different platform specifics (e.g. mobile devices with small screens and touch input vs. deskop PCs with large displays and mouse/keyboard input). Since all implementations are driven by the same configuration as specified in the authoring tool, this promotes portability, since the

re-usable templates have to be ported once to a new target platform in order to play the same games without adapting the input files.

4.3 Authoring Implementation

Apart from the implementations of the components to be used in the player application, programmers can provide a specialized implementation for the authoring tool, which is used in the Stage Editor of StoryTec to provide specialized WYSIWYG authoring for the specific gameplay modelled in the gameplay template. Using the example of the use case, the Stage Editor in StoryTec could be replaced by a view of the component in which the user can enter the correct codeword in a WYSIWYG-fashion and immediately test the game. Without this authoring implementation which provides the user interface for this, the user could only edit the parameters of the gameplay template using the Properties Editor of StoryTec, without the possibility to immediately see changes.

4.4 Metadata

Apart from the interface definition described in section 4.1, further metadata of the component and associated gameplay can be included. As an example, information about the appropriateness for a certain type of player (→ player modelling, see Göbel et al. (2010)) can be added, so that authors can see what kind of player will appreciate the gameplay provided in the component and choose it accordingly when creating an adaptive game in StoryTec.

4.5 Documentation

Documentation of the gameplay template can be automatically created or be added by the programmer/provider of the component manually. Automatically generated documentation can be built by the system dynamically by using the information provided in the interface definition (cf. section 4.1). For example, when an author requests help for a specific template, the system could generate an overview page listing optional and mandatory parameters or assets.
Manually added documentation can increase the quality of the documentation by providing textual as well as multimedia presentations of the gameplay template. For example, the documentation could include a small tutorial video how the component is used and also a preview video to give authors an impression what kind of gameplay is realized in the gameplay template.

4.6 Assistance

Especially for novice authors starting out with using StoryTec for game authoring, a gameplay template provider can include assistance mechanisms for these users in the authoring implementation (cf. section 4.3). The main assistance mechanism is the provision of software wizards, i.e. dialogue windows which guide the user through the correct usage of the component. In the “word puzzle” use case, a programmer could provide a wizard asking the user for the correct code word on one page, then for the difficulty on the next page, followed by the optional step of including a background image. This is shown in detail in section 5. The settings made in the wizard are then reflected in the settings found in StoryTec, so that advanced authors can then later fine-tune the settings. Expert users can disable the wizards if they feel that they can make the changes more efficiently themselves.

5. Prototype

In the following, a prototypical implementation of the concept described in the last section is shown, demonstrating how the concept is related to implementation and usability issues.
Gameplay templates are shown in StoryTec along with the general-purpose objects such as images, videos or variables built into StoryTec. However, these templates are not hard coded in StoryTec, but are rather loaded dynamically from a repository of templates which also includes the associated data such as the documentation or preview videos. Using this mechanism, the core StoryTec program does not have to be adapted each time a new gameplay template is added or an old one is removed. Additionally, the cohesion between the authoring tool and the templates is reduced, since template programmers only have to provide the implementation of their component and do not have access to the remaining parts of StoryTec.

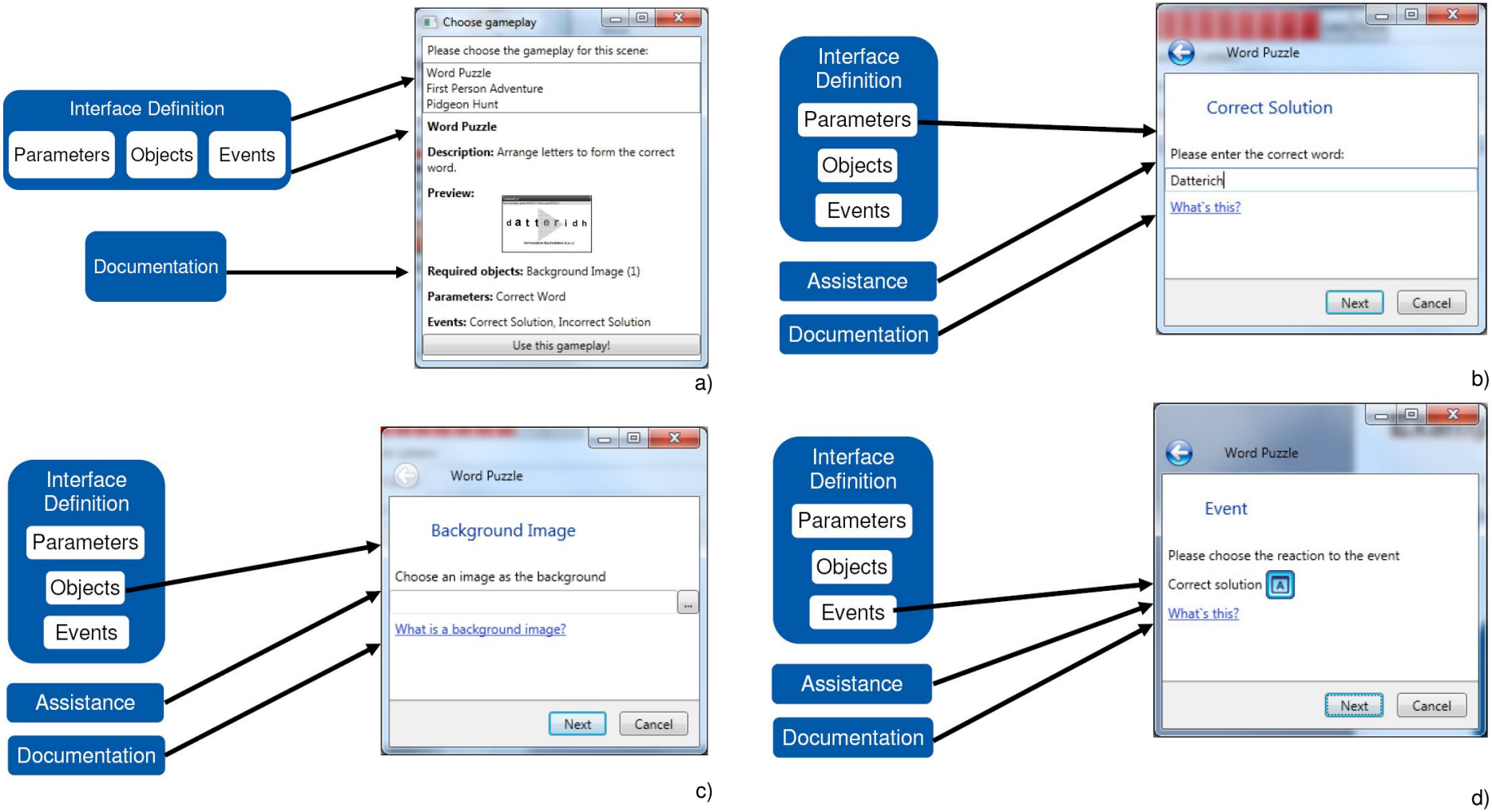

Figure 4: a) The initial wizard for selecting the gameplay template. **b)** A wizard page for the correct solution of the game. **c)** Choosing the background image for the scene. **d)** A wizard page for setting up the game's reaction to player inputs.

5.1 Authoring Tool Implementation

First we show the implementation in the authoring tool, specifically a set of wizard pages which guide the user through the choice and configuration of the gameplay template specified in the use case.
The first step towards this process is shown in Figure 4 a) where the novice user is shown a set of available gameplay templates. For each template, the automatically generated and the manually provided documentation is shown in a summary page, including a preview video of a reference implementation authors can view in order to get a feeling of how the component can be used. Additionally, the required and optional assets as well as parameters and events are listed.
Once an author has chosen one of the gameplay templates available to them, a second wizard assists in filling out the relevant information and with integrating content. Figure 4 b) shows the wizard page asking the user to enter the correct codeword. This wizard page can be automatically generated by the system by parsing the interface definition of the component and providing an input field for each parameter found. Additionally, the authoring implementation provider can deposit additional documentation such as the description of the purpose of individual parameters such as the code word.
Figure 4 c) is another instance of a wizard page that can be automatically generated from the interface definition and be enriched with documentation provided by the authoring implementation provider/programmer. This screen asks the user for the (optional) background image to be shown behind the game during gameplay.
Finally, the author has to configure the reaction of the game system to the events generated while the player is playing the game. This is shown in Figure 4 d). The visual programming approach of StoryTec is centred around the ActionSet Editor window (cf. Mehm (2010)). This editor is symbolized by a button with a capital “A” inside it, which can be seen in the figure. Using this editor the author can specify what will happen in the game once the player has entered the correct codeword. Among the possible actions are adding points to the score of the player, transitioning to another scene with a different gameplay template, having a character comment on the performance of the player or any other actions supported by StoryTec.

5.2 Player implementation

For demonstrating the separation of content and gameplay as well as portability as stipulated above, different implementations of the gameplay component found in the use case have been created. Apart from a 2D implementation as seen in Figure 1, an alternative representation has been built using the Unity3D engine[2] (see Figure 5). This implementation maps the characters found in the original implementation to sides of polyhedra which can be turned by using the mouse. While the representation is quite different from the 2D representation, the input parameters are identical and the game is controlled using the same overall mechanism.

[2] http://www.unity3d.com

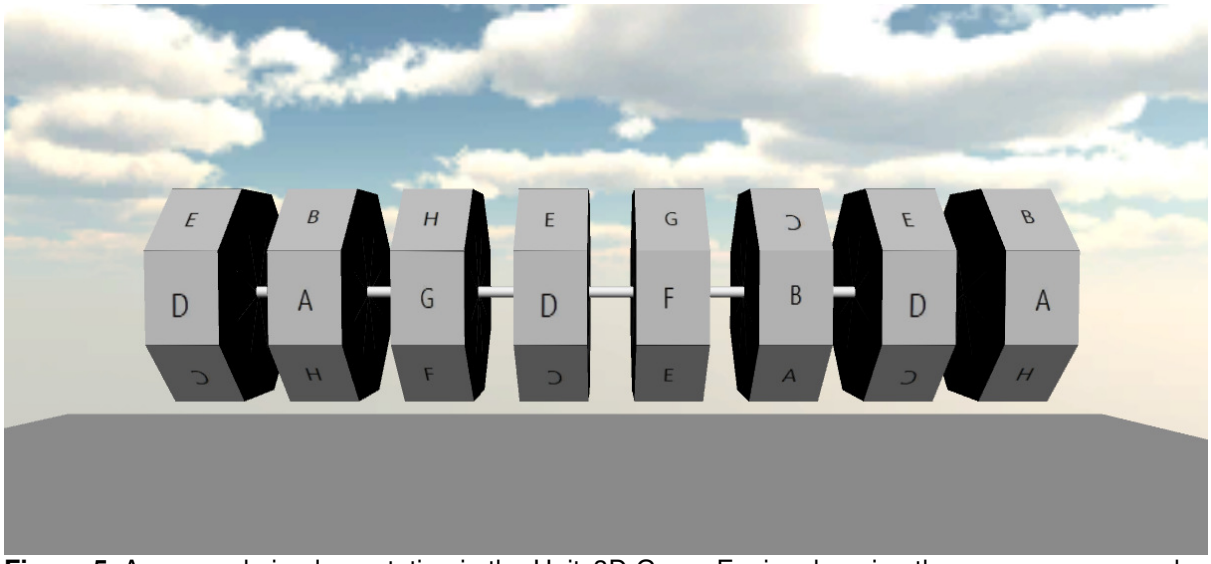

Figure 5: An example implementation in the Unity3D Game Engine, keeping the same core gameplay and parameters as the implementation shown in Figure 1 in place but replacing the visualization by an alternative presentation in 3D.

6. Conclusion

We describe a concept and prototypical implementation of the integration of component-based gameplay templates into the game authoring tool StoryTec. Each template captures the essence of a certain gameplay and is realized as a software component. During the authoring process, the aspects of these components are utilized by the authoring tool, for example by providing documentation or wizards for novice users. The two major results especially relevant for the production of Serious Games are the level of assistance authors can be provided with using this approach (see section 5) as well as the increased performance of an author.

No formal comparison studies have been carried out yet, however, several authors (including the original author of the city rallye game) were given the old version of StoryTec where the "word puzzle" sequence had to be constructed manually by the use of multimedia primitives (specifying images for each of the letters and then specifying in which order the images were exchanged when clicked on by the user). These authors were then given the newer version which included the gameplay template described here. It was found that the authors were able to complete the authoring task given to them in a considerably smaller amount of time. Additionally, all authors pointed out that the specific authoring implementation with the WYSIYG-approach made the task much clearer and easier to work on.

The work described herein is currently being merged with other work being undertaken in StoryTec, including the role-specific authoring mentioned in section 3 as well as the semi-automatic generation of authoring workflows based on the concepts introduced in this paper.

References

Bisognin, L., Carron, T. and Marty, J. (2010) "Learning Games Factory: Construction of Learning Games Using a Component-Based Approach", *Proceedings of the 4th European Conference on Games-Based Learning (ECGBL 2010)*, pp. 19-30, Academic Publishing, Reading, UK

Folmer, E. (1997) "Component based Game Development - A solution to escalating costs and expanding deadlines?", *Component-Based Software Engineering*, 2007, pp. 66-73

Göbel, S., Wendel, V., Ritter, C. and Steinmetz, R. (2010) "Personalized, Adaptive Digital Educational Games using Narrative Game-based Learning Objects", *Edutainment'10 Proceedings of the Entertainment for education, and 5th international conference on E-learning and games,* Springer Verlag Berlin/Heidelberg

Maciuszek, D., Ruddeck, G. and Martens. A. (2010) "Component-based development of educational games: The case of the user interface", *Proceedings of the 4th European Conference on Games-Based Learning (ECGBL 2010)*, pp. 208–217, Academic Publishing, Reading, UK

Mehm, F., Göbel, S., Radke, S. and Steinmetz, R. (2009) "Authoring Environment for Story-based Digital Educational Games", *Proceedings of the 1st International Open Workshop on Intelligent Personalization and Adaptation in Digital Educational Games*, pp. 113-124.

Mehm, F. (2010) “Authoring Serious Games”, *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, pp. 271-273, ACM, June 2010. ISBN 978 - 1 - 60558 - 937 - 4
Stolovitch, H.D., Thiagarajan, S. (1980) *Frame Games,* Educational Technology Publications, Englewood Cliffs, N.J.
Szyperski, C. (1997) *Component Software: Beyond Object-Oriented Programming*. ISBN: 0-201-17888-5. ACM-Press, Addison-Wesley
Tornero, R., Torrente, J., Moreno-Ger, P., Fernandez-Manjón, B. (2010) “e-Training DS: An Authoring Tool for Integrating Portable Computer Science Games in e-Learning” *Advances in Web-Based Learning – ICWL 2010,* pp. 259-268, Springer Berlin / Heidelberg
Tran, C., George, S. and Marfisi-Schottman, I. (2010) “EDoS: An Authoring Environment for Serious Games Design Based on Three Models”, *Proceedings of the 4th European Conference on Games-Based Learning (ECGBL 2010)*, pp. 393-402, Academic Publishing, Reading, UK