Script Cards: A Visual Programming Language for Games Authoring by Young People

Conference Paper · January 2006

DOI: 10.1109/VLHCC.2006.42 · Source: DBLP

3 authors:

Kate Howland
University of Sussex
45 PUBLICATIONS **505** CITATIONS
SEE PROFILE

Judith Good
University of Amsterdam
133 PUBLICATIONS **2,542** CITATIONS
SEE PROFILE

Judy Robertson
Heriot-Watt University
28 PUBLICATIONS **966** CITATIONS
SEE PROFILE

Script Cards: A Visual Programming Language for Games Authoring by Young People

Katherine Howland
Department of Informatics
University of Sussex
Brighton, BN1 9QH, UK
K.L.Howland@sussex.ac.uk

Judith Good
Department of Informatics
University of Sussex
Brighton, BN1 9QH, UK
J.Good@sussex.ac.uk

Judy Robertson
School of Computing and Mathematical Sciences
Glasgow Caledonian University
Judy.Robertson@gcal.ac.uk

Abstract

This paper describes Script Cards, a visual programming language which enables young people to script story events in a 3D role-playing game (Neverwinter Nights). Script Cards is designed to eliminate the need to learn a complex scripting language when creating interactive stories. An initial evaluation of Script Cards suggests that young people found it easy to use.

1. Introduction

Since the inception of visual programming languages (VPLs), a subset have been designed specifically for young people (see Show and Tell [1] for an early example). More recently work has focussed on end user programming languages which allow children to create sophisticated microworlds and simulations. Alice [2], one key example, aims to teach programming skills to novices in an environment which allows users to create 3D virtual worlds and govern the behaviour of objects they place in the world.

In addition to simulations and microworlds, games are another area in which children are increasingly becoming end user programmers. Some commercial games now ship with toolkits which allow users to create their own games. Neverwinter Nights (NWN), one such example, is a popular 3D role-playing game. NWN is produced by Bioware and ships with the Aurora Toolset, a powerful set of features which allows users to create their own 3D games.

Over the past three years, we have been using NWN to investigate the benefits of interactive game creation for children's narrative skills [3]. Many of the skills required to create a successful game, such as devising a compelling plot, creating characters with depth, and writing dialogue for the characters, are skills that are shared by more traditional forms of narrative.

The toolset's graphical interface allows users to create simple interactive stories. However, to add more complex interactions, such as conditional events, users must master NWScript (a scripting language based on the C programming language). Not surprisingly, the required technical skills are too difficult for our workshop participants to master in the timescale.

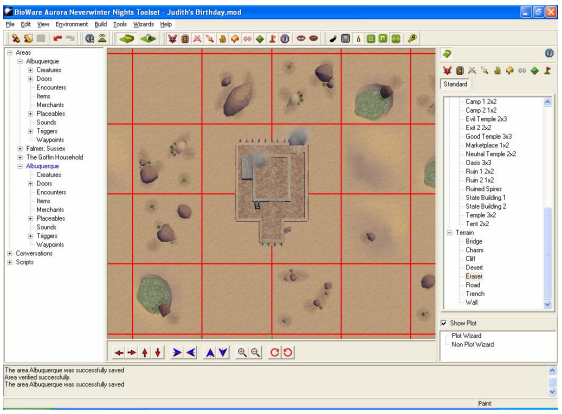

Figure 1: The Neverwinter Nights Aurora Toolkit

Additionally, whilst teaching children to program is a laudable goal in its own right, it detracts from our primary goal of fostering children's narrative skills. Creating a complex plot event through scripting requires children to step away from a narrative mode of thought to a detailed, low-level mode which focuses on variables and control structures. For this reason, we decided to create a visual language, Script Cards, which would allow children to specify complex plot events using a combination of graphics and natural language.

In the sections below we outline the design process, describe the environment itself, and summarise results

Visual Languages and Human-Centric Computing (VL-HCC'06)
0-7695-2586-5/06 $20.00 © 2006 IEEE

from a preliminary evaluation. We conclude with thoughts on further work.

2. Design Process

The design process for Script Cards differed from that of more general purposes VPLs because the language was developed specifically to provide a solution in the area of creation of interactive narratives. Many of the ideas which have proved very successful in recent work on microworld and simulation based VPLs could not be made use of due to the specific constraints of this project. A learner-centred design approach, based on a framework described in [5], was adopted to ensure that the software provided the required functionality and was usable by the target age group.

Appropriate coverage for this version of Script Cards was determined though analysis of children's games from previous game creation workshops (the initial implementation was not intended to provide full coverage; the aim was to build a functional language which could be evaluated to give a proof of concept). It was decided that the system should cover the most commonly used plot events which require scripting.

Usability requirements for the system were established based on data collected in a series of dedicated requirements gathering workshops with target users, and analysis of data collected at previous game creation workshops. The idea that the syntax required for scripting is the main barrier which our target users face was supported in interviews with participants at previous game creation workshops, and is in keeping with the findings of the Alice team. The second version of Alice eliminated the need for users to enter syntax and instead used a drag-and-drop based system, as syntax was singled out as a particular problem for novice users [4].

In accordance with our view that scripting is discordant with the way target users think about interactive story creation it was important for us to find out more about the way in which users do conceptualise this process. Participant requests for help at workshops provided useful information here. We observed that users tend to identify objects by pointing at them on screen, and describe in simple natural language what they wanted to happen .

This indicated that the method of object identification should be primarily graphical, with natural language descriptions used to create story events. It was also decided that story event construction should be scaffolded to ensure that only well-formed story events can be created.

We designed a system in which users select and insert graphical event components into predetermined structures to eliminate syntax errors without limiting creativity. A low-fidelity prototype was created and tested by target users. The testing gave positive results and indicated that users found the interface to be fairly intuitive. It emerged that the scaffolding in the initial design was too restrictive in some cases as there were some differences in the order in which users preferred to create events from component commands. The system design was altered to allow for these variations. A full description and explanation of the design process is given in [6].

3. The Script Cards Environment

3.1. Interface Level

The metaphor of a card game was used for the Script Cards interface. It was thought that young people would be familiar with the idea of dragging cards to different positions on the screen from popular computer-based card games such as Solitaire. The concept of cards only 'fitting' in certain slots depending on their category should also be familiar, and was used to provide scaffolding in the event creation process.

The key event building blocks, script cards, can be of three types: '*action card*', '*when card*' or '*if card*' representing an action statement, when statement or if statement respectively (see Figure 2 for an example of an action card).

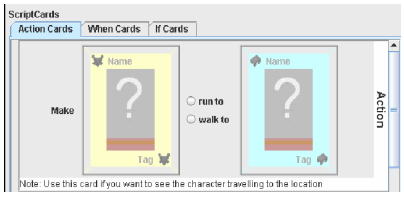

Figure 2: An Action Card

Each card represents its statement with a simple natural language description, and has blank slots where a graphical object needs to be inserted to complete the statement. A variety of script cards were designed to meet the established system coverage. For each card the user fills the blank slots with a slot-filler card representing the appropriate object. The slot-filler cards represent game objects with an object portrait, name and tag (textual identifier). Script Cards uses six categories of slot-filler cards, corresponding to the different categories in the NWN Aurora toolkit (creatures, items, doors, placeable objects, waypoints

and stores). Cards for the first three categories are shown in Figure 3.

Figure 3: Slot Filler Cards

In the Script Cards interface (shown in Figure 4), script cards are arranged in a vertical toolbar on the left, and slot-filler cards in a vertical toolbar on the right. The script composer pane is located in the centre of the screen. Once a card has been placed on the composer pane, options appear on an intermediary panel which give the user the choice to select the type of card they would like to add next. Only options which lead to a well-formed story event are displayed.

There is a simple process for adding a story event to a module, with an option to 'Add event to module' appearing once a valid story event has been created.

3.2. System Level

Script Cards is implemented in Java and, in order to achieve direct interaction between the software and the NWN module, the system uses a Java package which facilitates writing data to NWN modules. The 'nwnintf' package (which can be downloaded from http://www.cs.ualberta.ca/~games/scripting/nwn/resources.html) was created by the University of Alberta Artificial Intelligence Scripting Group during the development of their ScriptEase software (a menu based textual language for scripting NWN events). The package, used in conjunction with a stand alone compiler, allows the attachment of scripts to NWN modules.

Java drag and drop support was used to scaffold the event construction process. Once the first card has been placed, the system can begin to make deductions about the structure of the story event under construction. For example if a *when card* has been placed there will be no option to add another *when card*, because a story event can only have one trigger for its occurrence. This configuration allows users to construct story events in their preferred order, whilst providing scaffolding to ensure that nonsensical story events cannot be created.

To add slot-filler cards the user simply drags the appropriate card and places it in the appropriate slot on a script card. The drag and drop transfer handler allows any slot-filler card to be 'dragged', but only allows cards of the correct category to be 'dropped'. The Java Advanced Imaging package was used to build recognisable portrait style pictures on slot-filler cards for every possible game object.

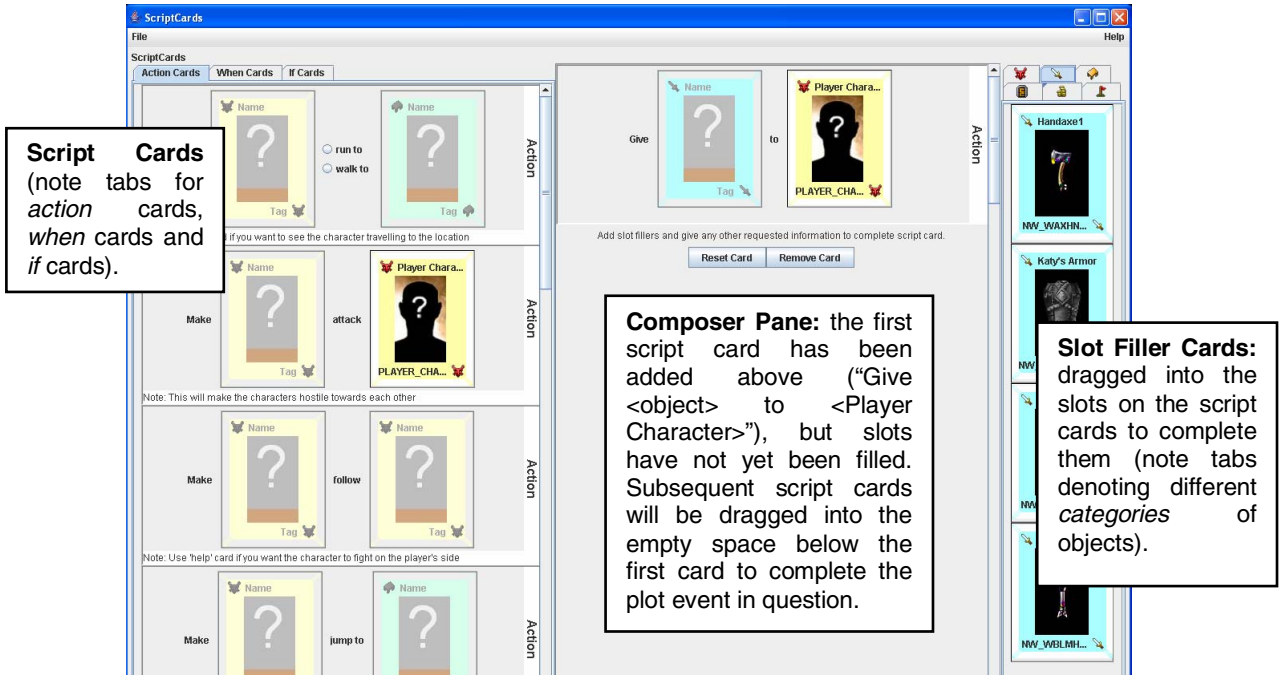

Figure 4: The Script Cards Interface

4. Evaluation

An initial evaluation of Script Cards was carried out on the last day of a five day game making workshop using the NWN Aurora toolkit A comparison of the participants' use of Script Cards with their use of NWScript (with NWScript acting effectively as an experimental control condition) was not feasible. Of the ten workshop participants only one had managed to do any scripting himself. Additionally, the initial version of Script Cards does not provide complete coverage of NWScript. Therefore, we focussed the evaluation on the extent to which participants could learn to use Script Cards to construct story events within a short session. Participants were each set three tasks which required them to create a story event using Script Cards based on a natural language description of an event.

The results were very positive with a high rate of correct task completion. Participants were interviewed after undertaking the tasks and all participants stated that Script Cards was easy to use. One participant noted that, "it makes sense… you can just read it like this, and it's easy to know what will happen". All participants felt that Script Cards was a definite improvement over NWScript. One noted that there is no need for "special words" with Script Cards as "you just use simple sentences". Two participants stated that Script Cards seemed to make scripting faster as well as easier, and one went on to say that "you make less mistakes, and if there is a mistake you can fix it a lot easier". He elaborated "[with NWScript] you have to put in exact words, all the grammar, you make one mistake and it's all a waste of time. With [Script Cards] it does all that for you, you just need to choose what it is and where it goes".

Where difficulties did occur it appeared to be due to oversights rather than fundamental misunderstandings of Script Cards. However, the task and interview data did suggest some interesting areas for improvement of the Script Cards interface. There seems to be a case of what Green [7] terms "premature commitment" at work in the interface which caused difficulty for some users. A full account of the evaluation is given in [6].

5. Conclusions

Script Cards is a VPL for scripting plot events when using the NWN Aurora toolset. It was designed to allow young people to use a combination of graphics and natural language to express events in a way which is as close as possible to the way they describe these events to others. An evaluation of Script Cards suggests that the prototype system achieved its aim, and was easy to understand and use by our target users.

One of the main strengths of Script Cards lies in the fact that young people can express their story events in their preferred order (an option not available in most conventional programming languages). Additionally, it strikes a balance between allowing young people this freedom in expressing their plot events, and providing sufficient scaffolding so that they do not create uninterpretable scripts.

Development of Script Cards is ongoing. We are currently extending the scope of the system to cover a wider range of possible plot events. In addition, we are investigating solutions to the issue of premature commitment in the hope that this will make Script Cards even more easy and motivating to use.

6. References

[1] Kimura, T.D., Choi, J.W. and Mack, J.M. (1990). *Show and tell: A visual programming language*. In E. P. Glinert, (Ed.) *Visual Programming Environments: Paradigms and Systems*, pp. 397-404.

[2]Conway, M., Audia, S., Burnette, T., Cosgrove, D. and Christiansen, K. (2000) *Alice: lessons learned from building a 3D system for novices.* Proceedings of CHI 2000, pp. 486-493.

[3] Robertson, J. and Good, J. (2006). Children's narrative development through game authoring. *TechTrends* 49: pp. 43-59.

[4] Kelleher, C., Cosgrove, D., Culyba, D., Forlines, C., Pratt, J.and Pausch, R. (2002). *Alice2: Programming without Syntax Errors*. User Interface Software and Technology. Paris, France.

[5] Good, J. and Robertson, J. (In Press). "CASS: A Framework for Learner Centred Design" *International Journal of Artificial Intelligence in Education.*

[6] Howland, K. (2005). *A Visual Programming Language for Scripting Events in Neverwinter Nights*. Unpublished Master's Dissertation, University of Sussex.

[7] Green, T. R. G. (1989). *Cognitive dimensions of notations*. In A. Sutcliffe and L. Macaulay (Eds.) *People and Computers V*. Cambridge, UK: Cambridge University Press, pp 443-460.