Tools for Developing Design Patterns for Mathematical Computer Games

Yishay Mor, Niall Winters
London Knowledge Lab,
Institute of Education,
University of London,
23-29 Emerald Street,
London WC1N 3QS.
yishaym@gmail.com,
n.winters@ioe.ac.uk

David Pratt
Institute of Education,
University of London.
20 Bedford Way,
London WC1H 0AL
d.pratt@ioe.ac.uk

Staffan Björk
Computer Science and Engineering
Chalmers University of Technology and Göteborg University
SE-412 96 Göteborg, Sweden
staffan.bjork@chalmers.se

Abstract

The last few years have witnessed a growing recognition of the educational potential of computer games. However, it is generally agreed that the process of designing and deploying computer games for learning generally is a difficult task. As an approach to support this interdisciplinary activity, we promote the use of design patterns that describe specific ways of designing or adapting games as well as how to deploy these into learning environments. This paper provides a detailed description of the methodology involved in identifying over 90 design patterns for designing and deploying computer games for learning mathematics.

In addition, a suite of methodological tools that was developed to facilitate the process is described. We discuss how these tools were developed in parallel with the language, as our understanding of the process in which we were engaged evolved. Furthermore, we describe how they supported our collaborative development.

Keywords
Serious games, design patterns, learning, games, mathematics

Introduction

The last few years have witnessed a growing recognition of the educational potential of computer games. However, it is generally agreed that the process of designing and deploying computer games for learning generally is a difficult task. The Kaleidoscope project *Learning patterns for the design and deployment of mathematical games* aimed to investigate this problem from the premise that designing and deploying games for mathematical learning requires design knowledge from diverse domains of expertise including mathematics, games development, software engineering, learning and teaching. In setting out to capture the process of designing and using mathematical games, we recognised at the outset the complex challenges we faced due to the ambitious nature of the enterprise. These was documented in the first outcome of the project, a review of the literature (Mor, Winters, Cerulli, and Björk, 2006; Alexopoulou, et al, 2006), which highlighted both the educational potential of games but also the complex challenges which impede the realization of this potential. We identified a range of design approaches to educational research in the field of technology enhanced mathematical education, and that an inherently interdisciplinary process was needed. Although simply designing a compelling game may be a challenge in itself, we argued that the issues pertaining to communicating design knowledge across a broad set of domains are dominated since failing in one discipline makes successes in other disciplines irrelevant. This interdisciplinary approach required a method to bridge these disciplines and make it possible for them to explicitly describe their knowledge in a fashion understandable by researchers and practitioners in other disciplines. The basis for the chosen methodology was design patterns, a methodological tool developed specifically to express tacit knowledge that can only partly be formalized and do so in a way that is accessible to a non-specialist.

This paper describes our methodological process used to develop a collection of over 90 design patterns. Our focus was on eliciting patterns based on real-world experience. This is important because the patterns are meant to be applied in real-world situations and must therefore be based on everyday practices. Furthermore, we support an interdisciplinary approach to the development of computer games for learning, and so communication and expertise sharing amongst participants is critical. We view design patterns as an enabler of this process and detail three distinct steps in their development: development of typologies, development of case studies, development and iteration of patterns. This iterative cycle is aimed at focusing participants on the relationship between design and deployment by reflecting on their experience. Each step is supported by a suite of online tools (http://lp.noe-kaleidoscope.org) including a typology tool, case studies repository, pattern editor and the pattern browser.

Background

Computer (and video) games are emerging as a fundamental activity in children’s lives. From their commercial origins in the 1970’s the computer game industry has grown rapidly in the last decades. In 2004, the US market alone was worth $9.9 billion (NPD Group, 2005), and in the UK in 2002 the market was worth approximately £2 billion (ELSPA, 2005). As such, games are playing a more central role in peoples’ lives than ever before and as a result games are becoming a topic of serious research interest, sometimes referred to as ludology. Given the size of the game industry and the interest from research communities it may appear somewhat surprising that there has been raised an explicit concern among professional game designers that a developed design discipline for computer games is lacking, and specifically that a language to discuss gameplay is needed (Spector 1999; Costikyan 2005).

Several potential solutions to this problem have been proposed by the industry itself. Writing to a designer audience, Church (1999) introduced formal abstract design tools (FADTs) as a way to reach a shared design vocabulary. Falstein (2002) suggested a more formalized method with the 400 Rules Project, which has collected proven game design rules and techniques, stating these as instructions. Both FADTs and the 400 rules project have been promoted within the game industry but have suffered from not having resources allocated to them to make them in into collections of significant size to be practically applicable.

Several books dealing with game design have been written by professional game designers (Schuessler and Jackson, 1981; Crawford 1982; Rollings and Adams 2003, Koster 2004). Although used both within the industry and education these books are based upon personal experiences and anecdotes without a framing in academic disciplines such as design or pedagogy. One of the few examples of game design textbooks written by educators is Game Design Workshop (Fullerton, Swain, and Hoffman, 2004) which advocated methods typically found within interaction design (Sharp, Rogers, and Preece, 2002; Shneiderman and Plaisant, 2004), e.g. iterative design cycles, end-user testing, and low fidelity prototyping. Looking at academic publications, the field's first peer-reviewed journal was only launched in 2001 (Aarseth, 2001) and a collection of books (Salen and Zimmerman, 2003; Juul, 2005) and anthologies (Salen and Zimmerman, 2006; Wolf and Perron, 2003) show the extent of research produced. However, little of this has been focused towards the design of games but rather towards game studies, studies of players, or studies of games in relation to other media. As a reaction to this, voices within the discipline of ludology have stressed the importance of "making games for learning instead of playing games for learning" (Kafai, 2006) and thereby indicate the importance of understanding what games are so that one can create games with specific design goals. These design goals have to co-exist with characteristics that have been identified as being essential for activities to be pleasurable (Csikszentmihalyi, 1991), which in a game context translated to providing good game-play (Järvinen, Heliö, and Mäyrä, 2002).

Designing Languages and Design Patterns

Looking at design in general, theorists such as Jones (1992) have differentiated between traditional and modern design, pointing to several fundamental differences that require designers to rethink their methods. Specially, Jones states modern design demands more consideration due to the increased complexity in technology available as well as better understanding of the diversity and interrelationships within the society in which a product will be used. The increased demands will typically enlarge the circle of people involved in the actual design process; either other designers or people with vested interests in the design results (e.g. financers, distributors, producers, retailers, end-users, service providers, government officials, interest organizations).

Although traditional methods of externalizing the design process, such as sketching, can allow designers to divide work time on a design to several different periods, these do not support explicit externalization of the process of designing but rather provide snapshots of specific potential designs. This may not be a problem within a well-developed design community if they have a well-developed *design language* where the implicit assumptions are embedded in the work process. Rheinfrank and Evenson (1996) describe design languages as consisting of collections of elements, principles of organization, and qualifying situations and state three distinct purposes of a design language: embed meaning into artefacts, allow artefacts to express meaning to people, and allow artefacts to be assimilated into peoples' lives. However, modern design may require involvement with people not fluent in a particular design language, e.g. designers from various specialities or stakeholders with no design background. Although these people may understand the particular design language, as any user of a product must have, they may not be able to *express* themselves in the language.

Design patterns were developed as a form of design language within architecture (Alexander, Ishikawa, Silverstein, Jacobson, Fiksdahl-King, and Angel, 1977). This was done with the explicit aim of externalizing knowledge to allow accumulation and generalization of solutions and to allow all members of a community or design group to participate in discussion relating to the design. As such, design patterns can be seen as a methodological tool to support the creation of an explicit design language. In general, a design pattern is defined as a high-level specification for a method of solving a problem by design. Its particular strength is in highlighting recurring techniques and solutions to design problems that are found again and again in real-world application development. Design patterns enable this process of knowledge discovery by specifying the particulars of a problem, and how the designated design instruments can address them. Classically, design patterns have been proposed in a format that consists of the following components (Alexander, David, Martinez, and Corner, 1985):

- An introductory paragraph, which sets the context for the pattern
- A concise problem statement
- The body of the problem—it describes the empirical background of the pattern, the evidence for its validity, the range of different ways the pattern can be manifested
- The solution that describes the relationships required to solve the stated problem, in the stated context. It is preferable to state the solution in the form of an instruction. A diagram may be included here.
- A relationship between this pattern and others.

The idea of design patterns has been shown to other disciplines. Once introduced design patterns have quickly become widespread within the domains of computer science to describe aspects of object-oriented programming (Gamma, Helm, Johnson and Vlissides, 2001). Within human-computer interaction, they have been used to describe problem domains (Ericksson, 2000; Borchers, 2001). The idea of using design patterns for describing game design has been introduced by members of the game industry (Kreimeier, 2002) and an extensive collection has been developed for gameplay design patterns (Björk, 2004).

Games and Learning Mathematics

It has been noted by Vygotsky (1978) and Bruner (1976) that there appears to be great learning potential for the use of games. Thus, the value of play for learning is no novelty. Over the last decade, the value of electronic games for learning has been recognized by academics and designers. Gee (2004) provides a classification system for games and their learning impact. Many experienced mathematics educators use games in their daily practice. Such games foster both individual and group learning, and provoke children to brave mathematical challenges which are far more advanced than their school curriculum subjects. In 2002, the British Educational Communications and Technology Agency undertook a small-scale pilot study (BECTA, 2002) investigating the use of six computer games in a school setting. In summary, they found some promising potential for future work by researchers, teachers and games developers based on their initial, tentative findings that games can support student's ICT skills, increase their motivation, encourage collaborative working and have positive side effects such as increased library use.

However, the design and deployment of games is not a trivial task. Kirriemuir and McFarlane (2003) undertook a literature review of games and learning for the National Endowment for Science, Technology and the Arts in the UK. An overview of the main developments in research into gaming and the educational relevance of video games illustrated that, although the use of 'mainstream' games in schools is rare, parents, carers and teachers increasingly recognise the potential of games to support valuable skills development, such as "strategic thinking, planning, communication, the application of numbers, negotiating skills, group decision-making and data-handling". Significantly they also highlight the fact that educational games often fail to realise players' expectations because the games are often too simplistic or repetitive with respect to commercial computer and video games, and are often poorly designed, with little support for active learning to achieve understanding.

The above emphasizes that designing games for education is difficult since it requires the integration of state-of-the-art knowledge about software engineering, gameplay design, with the latest knowledge about pedagogy, the former the preserve of engineers and designers and the latter the preserve of academics and pedagogues. One project exploring the area,, Electronics Games for Education in Math and Science, investigated the design and use of computer games in enhancing mathematics education specifically for students aged 9-14 (Klawe, 1998). In particular, it examined prototyping educational computer games and prioritised focused quantitative and qualitative studies to evaluate the effectiveness of various design and use options. The project findings suggest that computer games can be highly effective in increasing children's learning and enjoyment of mathematics when children actively "think about and value the mathematics embedded in the computer game [with] three factors to be particularly important in focusing students' attention on the mathematics: teacher attitudes, supporting activities and collaborative play". Without these factors almost no mathematical learning results from playing the game, stressing the centrality of the facilitator and pedagogic principles (Jonker & Galen, 2004).

Process and Initial Tools

Design patterns were central to our study since they promised to provide a mechanism which could embrace both design and deployment by capturing the essential recurring themes that underlie the iterative process from initial concept to pragmatic use in the classroom. Our aspiration was that, by identifying such patterns, design knowledge would become (i) explicit, enabling fragmented stakeholders a means of appreciating the whole endeavour, and (ii) cumulative, allowing future work to build on the actions of earlier developers and users. Focusing on computer games for learning mathematics, we realized that we were not expanding on already existing collections of design patterns for gameplay or programming but developing a new collection. This was not only scoped to include the design of computer games for learning mathematics but also, the challenges of deployment. Thus, the pattern language reflects simultaneously on the production of artefacts and the context of their use.

Due to the distributed nature of the team, it was necessary to develop a suite of on-line tools to facilitate the process of collaborative language development. These tools would, we hoped, allow the team to collaborate in reflecting on design process in order to share initial thoughts about the design patterns and to critique and modify other inputs. The use of workshops was critical in disseminating the emerging patterns and extending the breadth and depth of the language. The workshops also helped to identify relationships between the patterns, which began to form a hierarchical structure as we came to understand which patterns were in fact instances of others, and which patters espoused a higher level of abstraction.

Typologies

After identifying relevant knowledge domains, the next effort was aimed at producing a set of typologies that could provide a structured lexicon derived from these domains; in effect they act as a resource in the form of a content-based relational map, for classifying the different aspects of design knowledge required in the process of the design, development and application of mathematical games for learning.

For our purposes the typologies used were: *mathematical content*, *learning and instruction*, *educational context*, *games*, *interaction design* and *software design*. The typologies were developed to reflect the synergistic collaboration between the design and deployment strands of the project, through addressing critical aspects of the process identified by experts in both strands.

We initiated this process through a brainstorming session conducted during a project meeting. This provided the initial outline and candidate typologies. These were whittled down and following this session, domain experts published an initial draft of their respective typology online using an earlier version of the typology tool. These drafts were scrutinized by the other project members. Using an online discussion mechanism, we queried each other for clarifications and illuminated possible gaps and overlaps. This led to the typologies being iterated through a synergistic collaboration between all partners. The rationale for this approach was our belief that for the typologies to be a productive tool, they needed to be refined through productive use. This resulted in their current form. However, we note here that the typologies are not fixed. Using the online tool they are available for constant refinement by the wider community.

The *typology tool* allows community members to browse, review and edit the various typologies. A typology is a complex monolith which captures the knowledge of a single specialist (or specialist group). Hence it is convenient to edit it off-line and upload versions as they mature. This is achieved by using a mind map editor (the open-source program *FreeMind*). Once a typology is uploaded, it can be viewed either as a map image or as an html tree. The definitions of the terms are displayed alongside the tree view, and can be edited online. Each typology map is accompanied by a discussion forum, where other members of the community can comment, suggest changes or ask for clarifications. When a new version is uploaded, the previous versions are retained for reference. In order to facilitate cross-referencing between typologies as well as typology-driven context descriptions in the case studies and patterns, we provide a wiki-style method of linking to typology terms.

Case Studies

Once the typologies were developed, we used them to help participants work with and through their case studies. The purpose of the case study development is multifold: i) to provide concrete examples of practice within disciplines; ii) to map practices and content detailed in the case study to the set typologies iii) for the team to identify linking points between disciplines and iv) to provide the starting point for pattern development.

Overall, we received 24 case studies. These can be broken down in two main categories: i) those that resulted from the insights gained from deploying technology enhanced learning tools in classrooms and ii) those that reflected the design practices evident in their development. Each case was relatively short and was designed to be presented to others (either project team member or to those who took part in our workshops) and to act as a starting point for discussion around design knowledge. Each case was broken down into: a) the context it describes, b) the relationship to the typologies, c) the aims of the particular case, d) development/deployment details and e) outcomes. It was around these aspects of each case that we expected to find common starting points for the development of the initial set of ‘seed’ patterns.

The *case study repository* allowed case studies to be created, edited and indexed online using a simple template and a visual editor. Contributors created a new case study by providing a name and a short summary. They were then directed to an editable online document based on the case study template. This template prompted them to provide the context, aims, details, outcomes and references. The details section was a free-form narrative which formed the main bulk of the case study. Contributors were encouraged to include graphical materials, such as screen shots and diagrams.

Patterns and Pattern-Related Tools

The tools described above provided a basis for the task which attracted most of our efforts, developing of a language of design patterns for computer games aimed at mathematical learning. This language was developed in a highly iterative collaborative process, supported by bespoke tools which co-evolved with the refinement of our own practices. A reoccurring practice of work emerged from this process, which was formalized and then replicated and modulated in the course of several public workshops. This process also identified the need of additional tools that were developed.

Our design pattern for developing pattern languages begins with the identification of a design problem, the context in which it arises, and a viable method of solution. This triad is captured as a ‘seed’ pattern and then iteratively elaborated, refined and weaved into the fabric of the language.

The original template of design patterns by Alexander et al. to support the participatory design in architecture was modified to suit the context of mathematical games to be used in learning environments. Each pattern is introduced through a Problem/Intent in a vein similar to the Alexanderian model but to support the various domains each pattern is described in how it applies to mathematical content, learning and instruction, educational context, games, interface and interaction, and software design.

To encourage a volume of pattern candidates and interdisciplinary discussion the open source motto ‘release early, release often’ was adopted as the main paradigm regarding pattern creation. Nevertheless, the patterns needed to go through several iterations of refinement before they are suitable for general consumption. To this end, we defined four phases each pattern needed to pass through: ‘seed’, ‘alpha’, ‘beta’ and ‘release’.

Patterns often emerged in the course of a discussion between experts, e.g. at a workshop, or in the process of refining another pattern. When this happened, they were immediately noted in a minimal form, typically a name and a sentence or two of description. Such a pattern is defined was a seed: it was legible only to the person or group who entered it.

Since a seed pattern was little more than a note the author had taken for herself, it was obvious that its author needed to elaborate it before it could be shared with others. The alpha phase was aimed at this purpose. In this phase, the pattern’s original author filled in the pattern components according to the template: she described the problem addressed or the intent it served, identified the context in which it was applicable, added the details of its structure and situated it in the hierarchy of the pattern language.

When a pattern was reviewed by other experts, they related it to their experience and to other patterns of which they were aware. As a consequence, they would ask for clarifications or offer refinements. They would identify lateral links with other patterns and see the pattern in other case studies, which the original author was not familiar with. This collaborative process aimed to promote patterns to a mature, coherent and robust state. Such patterns would be classified as being in beta state.

Once a pattern reached beta state, it had the potential to be useful to the broader community, i.e. anyone from outside the development team who might be interested in the questions to which this pattern might refer. At this phase, the pattern was exposed publicly and slowly refined based on public feedback. Thus, a pattern might have remained in this phase for several months before it was finally upgraded to release state.

The rest of this section describes the tools developed to support this practice and their modes of use.

Pattern Template

One of the key objectives of developing a pattern language is the standardization of design knowledge. As a result, the task of defining a *Pattern Template* is common to many pattern language initiatives and was adopted by us. Although the template is not a tool in the sense of being a concrete artefact or application (but see the pattern editor below), it is a *methodological* tool since it enabled us to work structurally, dividing and synchronizing work.

Alexander (1977) defines a pattern as a “three part rule, which expresses a relation between a certain context, a problem, and a solution”. Each community elaborates this structure in a slightly different form. Our template includes the following sections:

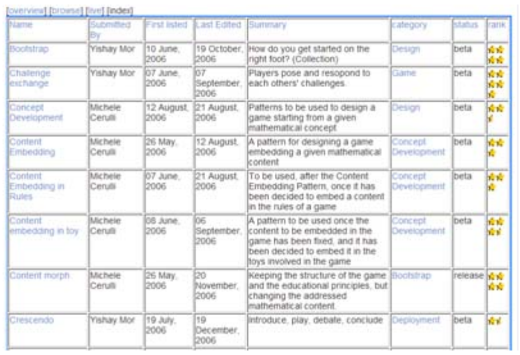
- Name: 3-4 words, catchy and descriptive.
- Meta-date (author, entry date, last edit date, category, and status)
- Short summary: one or two lines.
- The problem / intend: what is the problem or need that this pattern addresses.
- Context: where is it applicable?
- The pattern: ‘cookbook’ description of a possible method of addressing the specified problem in the specified context.
- Related patterns: vertical and lateral links to other patterns and categories of patterns.
- Examples: links to concrete case studies where this pattern is manifested.
- References: note of academic or other resources pertaining to this pattern.

Choosing a good name is important because it makes the core idea of the pattern explicit in a compact and easy-to-remember manner for users. This name is the pattern’s identifier in verbal and online design discussions, as well as the key for searching or browsing the pattern language site. The short summary allows visitors to quickly evaluate their interest in a pattern. It also plays an important role in the pattern refinement dynamics described in the next section: when a pattern emerges in the course of discussion, its initial record will include only its name and summary. Further details will be added by the pattern’s main author and then refined through internal and public discussions. The context includes a narrative of the circumstances in which the pattern was identified, and a mapping to the various typologies, as described above. In the case of simple patterns, the solution body itself may take the form of a sequence of numbered steps for users to follow to operationalise it. For more complex patterns, diagrams and detailed descriptions will be added as well as implementation notes, links to sub-patterns etc. The relationships between patterns are listed along two axes: elaboration, and association. Elaboration defines a hierarchical inheritance relationship similar to class structure in object oriented languages. Association is manifested through ‘leads to’ and ‘follows’ links, signifying temporal, content, or structural lateral links (e.g. ‘after using this pattern, that one may be useful’ or ‘that pattern suggests a different approach to a similar problem’). Mapping these links important as it defines the networked structure of the pattern language.

Our templates are all ‘soft templates’: they offer a structure, but do not impose it. The contributor has full artistic license to describe her case study in any way she finds most fitting. Typically, when a pattern is identified, only the name, meta-data and summary are entered. In the transition from ‘seed’ to ‘alpha’ the pattern author fills in all the slots of the template. Thus, as more patterns shift from seed to alpha and then to beta status, we have moved from a loosely defined patterns set, to a structured and mapped pattern language. The pattern language is organized hierarchically. The top layers of this hierarchy are abstract categories of patterns, while the lower ranks are concrete patterns and sub-patterns.

The Pattern Editor

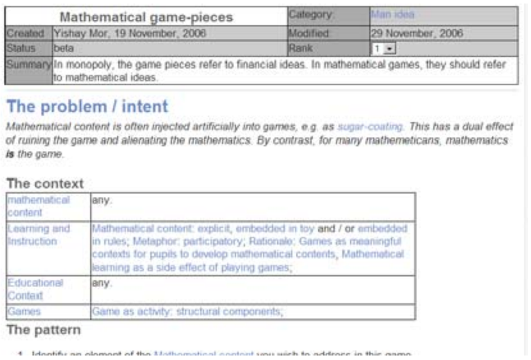
The core tool for developing the pattern collection was a *Pattern Editor*. This tool supported the transition from the identification to the development stage of the methodology, marked by populating the pattern repository with seed patterns derived from the case studies. This is achieved by using the online system to provide a name, short summary and category. Once this is done, the user is directed to the webpage generated for the new pattern. The meta-data for this pattern is automatically generated and listed in the page header. As with the typologies and case studies, each pattern is accompanied by a discussion forum to facilitate its refinement by the community. Pattern editing and publication is a two-phase process: an author can edit and save a draft of the pattern without affecting the publicly viewable version. Once the author hits the 'publish' button, the current version is archived and the draft is published.



[overview] [browse] [live] [index]

Name	Submitted By	First listed	Last Edited	Summary	category	status	rank
Bootstrap	Yishay Mor	10 June, 2006	19 October, 2006	How do you get started on the right foot? (Collection)	Design	beta	★★★★
Challenge exchange	Yishay Mor	07 June, 2006	07 September, 2006	Players pose and resopond to each others' challenges.	Game	beta	★★★★★
Concept Development	Michele Cerulli	12 August, 2006	21 August, 2006	Patterns to be used to design a game starting from a given mathematical concept	Design	beta	★★½
Content Embedding	Michele Cerulli	26 May, 2006	12 August, 2006	A pattern for designing a game embedding a given mathematical content	Concept Development	beta	★★★
Content Embedding in Rules	Michele Cerulli	07 June, 2006	21 August, 2006	To be used, after the Content Embedding Pattern, once it has been decided to embed a content in the rules of a game	Concept Development	beta	★★★
Content embedding in toy	Michele Cerulli	08 June, 2006	06 September, 2006	A pattern to be used once the content to be embedded in the game has been fixed, and it has been decided to embed it in the toys involved in the game	Concept Development	beta	★★★½
Content morph	Michele Cerulli	26 May, 2006	20 November, 2006	Keeping the structure of the game and the educational principles, but changing the addressed mathematical content.	Bootstrap	release	★★★★
Crescendo	Yishay Mor	19 July, 2006	19 December, 2006	introduce, play, debate, conclude	Deployment	beta	★½

Figure 1: Index view in the Pattern Browser.



Mathematical game-pieces		Category:	Man idea
Created	Yishay Mor, 19 November, 2006	Modified	29 November, 2006
Status	beta	Rank	1
Summary	In monopoly, the game pieces refer to financial ideas. In mathematical games, they should refer to mathematical ideas.		

The problem / intent

Mathematical content is often injected artificially into games, e.g. as sugar-coating. This has a dual effect of ruining the game and alienating the mathematics. By contrast, for many mathemeticans, mathematics ***is*** *the game.*

The context

mathematical content	any.
Learning and Instruction	Mathematical content: explicit, embedded in toy and / or embedded in rules; Metaphor: participatory; Rationale: Games as meaningful contexts for pupils to develop mathematical contents, Mathematical learning as a side effect of playing games;
Educational Context	any.
Games	Game as activity: structural components;

The pattern

1. Identify an element of the Mathematical content you wish to address in this game

Figure 2: Part of an individual pattern.

The Pattern Browser

Once a pattern has been added to the collection, it can be found, edited, and woven into the fabric of the language through the *Pattern Browser*. The pattern browser is the central tool in our system. It provides several modes for viewing the patterns, as well as entry points to tools for creating new patterns and structuring the language. When a new pattern is entered into the database, it is automatically listed in the table view, which can be sorted by various keys. As part of the refinement stage, the pattern author needs to map its relations to other patterns. She does so by first assigning it to a category through the pattern editor.

Looking beyond the single pattern, the structure of the language as a whole can be edited as a FreeMind mindmap. Community members reviewing the language start from the overview mode, which displays the main branches of the pattern hierarchy as an image. They then switch to browse mode and use a tree-based navigation tool to hone in on the patterns they wish to discuss. The structure of the language as a whole can also be discussed in a forum adjacent to the browser. Alternatively, reviewers can use live mode, a java applet to browse the map, combining the functionality of both overview and browse mode.

To eliminate the risk of 'orphaned' patterns which are not linked into the language structure, the Index (table) mode lists all patterns in the database, even those in preliminary state which have not yet been integrated into the structure. Index mode displays each pattern's name, author, creation and last modification date, summary description, category, status and ranking. The table can also be sorted by each of these columns.

Trail Browser

Paradoxically, often as more expert knowledge is embedded in a pattern languages it may become less accessible to novices. As a pattern language grows richer and more intricate it becomes the private language of the community which created it. Novices do not know where to start and how to penetrate it, because the structure of the language does not expose the path along which it evolved. In an attempt to address this issue, we have added a tool called *Trail Browser*. A trail is an informal illustrative account of how patterns were derived or how they might be used. It is not presented as hard data or detailed analysis, but rather as an aid for the casual reader.

Trails are built using the following procedure: Using the pattern map, choose the top-level category you are addressing. These are divided into methodology, design, development, deployment and evaluation. With the category, search through the patterns to find one that you feel best suits your problem. It is also useful to read the online discussions, as they provide a background to the process of how the pattern was developed. If the pattern is suitable, move onto the ‘Leads to’ pattern(s) and determine how they can help you with a more focused part of your game development or deployment. If the pattern is not suitable, move to the ‘Follows’ pattern(s) to investigate your problem at a more abstract level.

Patterns of Design and Deployment

Overall, we have collected 91 original patterns, about half of which are in Beta or Release state. The split between the three categories is fairly balanced, with a slight bias towards the Structure category. In addition, 26 patterns taken from Gamma, Helm, Johnson and Vlissides (2001) were inserted and modified for the context. Due to space restrictions, no examples of identified and developed patterns are given. Interested readers can find the whole collection at http://lp.noe-kaleidoscope.org/ together with the tools.

The identified collection of patterns were structured in accordance with three main headings. These are:

- **Methodology** which describes general frameworks deployed when designing, developing and deploying games in educational contexts in general.
- **Process** which describes specific techniques relating to the processes of designing, developing, deploying and evaluating games for mathematical education. Most of these techniques are useful regardless of the methodology deployed, but some are derived from a specific framework.
- **Structure** which describes techniques deployed to enable the design of particular game elements. These techniques are derived directly from the authors' prior experiences in game development and are documented within the case studies.

The *Methodology* collection provides theoretical grounding and general frameworks for game design, development and deployment in educational contexts. We focus on three common practices for producing games in educational research, and point to several Related Knowledge Collections. Each practice described is interpreted with respect to the specific context under investigation that of games in mathematics education.

We are aware that the collection excludes methodologies (or lack of such) which are common in many commercial and research development settings. Our choice reflects a conscious preference for learner-centred approaches. Arguably, an uncompromising choice of methodology may restrict designers' creativity and flexibility. While we are aware of this risk, we believe it is important to be informed about the different paradigms, even if only to avoid their characteristic pitfalls.

We distinguish between methodologies and techniques. A choice of methodology defines the whole development process, from conception to deployment. Methodologies are typically exclusive, value-driven and theory-heavy. The choice of methodology is often implicitly derived from the choice of educational Metaphor and Grand Theory. Techniques, on the other hand, are methods of solving pointwise problems in the design process, such as overcoming a particular communication barrier. They are more widely applicable, in that they do not require the user to commit to a specific philosophy. In game like terms, a methodology is a map for reaching your prize, while the techniques are a set of keys for the doors you might need to open on the way. Some such 'keys' are presented in the form of Process patterns.

The *Process* collection of patterns describes specific techniques used in the processes of designing, developing, deploying and evaluating educational games in mathematics. It addresses the main processes involved in the 'life-cycle' of an educational game for learning mathematics. Thus, it describes techniques deployed to not only derive the Design of a game but also, its Development which encompasses its Deployment and Evaluation.

These patterns in the *Structure* collection highlight potentially powerful elements of Game design, Interaction design and Microworld design. They aim to expose and address specific recurring problems in these realms. Such patterns are closer to the common interpretation of *design patterns* in software engineering and game design and analysis. Some of them are extensions of known patterns in these domains. Within this collection, the following sub-collections can be found:

Discussion

During our work with developing the tools and the pattern collection several issues have become apparent to us that were beyond the scope of the project to address. Below we discuss the most salient of these issues.

Number of Patterns

We talk about patterns elaborating and being elaborated by other patterns within that web of relations. In this sense the further one drills down into the specific contexts of application of the pattern the more patterns one can find. Contrarily, the more one abstracts leaving behind the contexts of application the fewer patterns one finds. Indeed, we now see the enterprise as one of mapping that structure of inter-relationships between patterns of design and deployment. There is no question that the project has begun to construct and communicate that map through its website. In a sense, the task continues to be the invention of further constructs, that is to say higher level patterns, which reduce the number of patterns when the designer or the user wishes to access the map at the highest levels of abstraction.

Types of Patterns

The learning patterns we developed attempt to strike a balance between problem solving and being feature specific. Some patterns address the process of game development and in doing so emerged from problems we were trying to overcome. As such, they can be viewed as problem solving patterns. Others are directly concerned with particular game features and interaction issues and are considered feature specific. However, in describing the patterns we use the generic term ‘problem’ to encompass both perspectives. For example, to address a particular problem the designer may add a specific feature.

Relations to other Pattern Collection

Already in the beginning of the process, we identified that several other pattern collection have impact on designing mathematical games. Specifically, the knowledge described in gameplay design patterns and programming patterns were seen as intrinsic to any process of developing computer-based games, whatever the purpose of the specific game being created. During the process of creating the collection notes were made whenever specific links to other design pattern collections were found but no structured approach was taken to formalize the interrelationships between different pattern collections.

It is also worth mentioning that our collection was developed for a subset of games. Work within another subset would most likely produce a collection with some overlap but many differences.

The use of Tools

Although the patterns identified and described are relevant for those designing and deploying such games, we believe that another result is of more general value. This result springs from the fact that many design pattern collections are developed to bridge the knowledge domains of diverse disciplines to facilitate interdisciplinary collaboration through a common design language. However, the development of the pattern collection itself requires interdisciplinary collaboration and thereby a common design language. Thus, even though design patterns are parts of design languages, they are not the tools that can bootstrap a process to develop a design language.

The tools developed in the Kaleidoscope project are examples of more fundamental tools that support creating initial descriptions of the knowledge domains (the typology tool and the case studies repository), developing design patterns (the pattern template and the pattern editor), and structuring and presenting the collection (the pattern browser and the trail browser). Although created and implemented for development of a specific design pattern collection, we do believe that such tools can help facilitate the development of other design patterns and the developed tools can be modified or expanded to do so.

Conclusion

In the space of one year, the *Learning patterns for the design and deployment of mathematical games* project has proved the feasibility of identifying patterns that embrace both design and deployment. This has been achieved through the development of a methodology which brought researchers face-to-face with designers and face-to-face with classroom practitioners.

We have along the way found that the emerging map of design patterns is understood in differing ways by designers and practitioners. There is prima facie evidence that the former appreciate the strongly hierarchical relationships between the patterns and tend to be comfortable with a top-down reading of the map. Equally, practitioners seem to access the map through concrete instances and value the case studies and trails. We cannot know at this stage whether this distinction is fairly widespread but we note it with interest and expect that there will be a need to invent further ways in which the map can be read.

The emergence of a democratically accessible map of design patterns opens up the possibility of stakeholders working together in an intimate relationship, using the design patterns as a mutual language. Such a *connected design* approach should enable each community to appreciate the constraints, opportunities and aspirations of the other, promising the emergence of a new family of mathematical games which are educationally rich for the classroom and yet entertaining for the home.

References

Aarseth, E. (2001). Computer Game Studies, Year One. Game Studies, *The International Journal of Computer Game Research*. 1.1. Available for download from (last accessed 29.04.2007) http://www.gamestudies.org/0101/editorial.html.

Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I. & Angel, S. (1977). *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, 1977.

Alexander, C., Davis, H., Martinez, J. & Corner, D. (1985). *The Production of Houses*. New York, NY: Oxford University Press.

Alexopoulou, E., Bennerstedt, U., Childs, M., Jonker, V., Kynigos, C., Pratt, D., and Wijers, M. (2006): *Literature review on the use of games in mathematical learning, Part II: Deployment*. Report of the Learning Patterns for the Design and Deployment of Mathematical Games project.

Becta (2002). *Report on Computer Games in Education*. Available for download from (last accessed 29.04.2007) http://partners.becta.org.uk/index.php?section=rh&rid=11207.

Björk, S. & Holopainen, J. (2004) *Patterns in Game Design*. Charles River Media, 2004

Borchers, J. (2001). *A Pattern Approach to Interaction Design*. Wiley; 1st edition, 2001.

Bruner, J., Jolly, A., & Sylva, K. (1976). Play -- its role in development and evolution. New York: Basic Books.

Church, D. (1999). *Formal Abstract Design Tools*. Online article available at www.gamasutra.com.

Costikyan, G. (2002). *I Have No Words & I Must Design*. In Mäyrä, F. (2002) Conference Proceedings of Computer Games and Digital Cultures, pp. 9-33, Tampere University Press.

Crawford, C. (1982). *The Art of Computer Game Design*.

Csikszentmihalyi, M. (1990). *Flow: The psychology of optimal Experience*. New York Harper Collins Publishers

ELSPA (2005). *Public Announcement from The Entertainment and Leisure Software Publishers Association (ELSPA).* Available for download from (last accessed 29.04.2007) www.totalvideogames.com/news/ELSPA_Confirms_2005_As_Record_Year_9082_0_0.htm.

Erickson, T. (2000) *Lingua Francas for Design: Sacred Places and Pattern Languages*. In Proceedings of DIS 2000 (Brooklyn, NY, August 17-19, 2000). New York: ACM Press, 2000, pp 357-368.

Falstein, N. (2002). Better By Design: The 400 Project. *Game Developer magazine*, Vol. 9, Issue 3, March 2002, p. 26.

Fullerton, T., Swain, C. & Hoffman, S. (2004). *Game Design Workshop: Designing, Prototyping, and Playtesting Games*. CMP Books, 2004.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (2001). *Design Patterns – Elements of Reusable Object-Oriented Software*. Addison-Wesley, 2001.

Jones, J.C. (1992). *Design methods*., John Wiley & Sons, 2nd Edition, 1992.

Jonker, V. & Galen, F. (2004). Kidskount. Mathematics Games for Realistic Mathematics Education in Primary School. *10th International Conference on Mathematics Education* (ICME), Kopenhagen, Denmark.

Juul, J. (2005). *Half-Real: Video Games between Real Rules and Fictional Worlds*, The MIT Press.

Järvinen, A., Heliö, S. and Mäyrä, F. (2002). Communication and Community in Digital Entertainment Services: Prestudy research report, *University of Tampere Hypermedia Laboratory Report*. http://tampub.uta.fi/tup/951-44-5432-4.pdf, Last accessed 30.04.2007.

Kafai, Y. (2006). *Playing and Making Games for Learning: Instructionist and Constructionist perspectives for game studies, Games and Culture*, 1, 36-40.

Kirriemuir, J. & McFarlane, A. (2003). Use of Computer and Video Games in the Classroom, *Presentation to The Digital games Research Conference*, Utrecht University, The Netherlands, 4-6 November 2003.

Klawe, M. M. (1998). *When Does the Use of Computer Games and Other Interactive Multimedia Software Help Students Learn Mathematics?* Available for download from (last accessed 23.02.2006) http://www.cs.ubc.ca/nest/egems/reports/authors.html.

Koster. R. (2004). *Theory of Fun for Game Design*. Paraglyph, 2004.

Kreimeier, B. (2002). *The Case For Game Design Patterns*. Online publication available from http://www.gamasutra.com/features/20020313/kreimeier_03.htm.

Mor, Y., Winters, N., Cerulli, M. and Björk, S (2006): *Literature review on the use of games in mathematical learning, Part I: Design*. Report of the Learning Patterns for the Design and Deployment of Mathematical Games project.

NPD Group (2005). *NPD Group Reports on Video Game Sales and Best Selling Video Game Titles*. Available for download from (last accessed 30.04.2007) http://retailindustry.about.com/od/seg_toys/a/bl_npd012703.htm.

Rheinfrank, J. & Evenson, S. (1996). Design languages. In Winograd, T. (ed.) (1996). *Bringing design to software*, pp. 63-85, ACM Press, 1996.

Rollings, A. & Adams, E. (2003). *Andrew Rollings and Ernest Adams on Game Design*. New Riders Games; 1st edition, 2003.

Salen, K. & Zimmerman, E. (2003). *Rules of Play: Game Design Fundamentals*, Cambridge, Mass.: MIT Press.

Salen, K. & Zimmerman, E. (2006). (Eds.) The *Game Design Reader: A rules of play anthology*. Cambridge, Mass.: MIT Press.

Schuessler, N. & Jackson, S. (1981). *Game Design – Volume 1: Theory and Practice*. SJ Games. 1981.

Sharp, H., Rogers, Y. & Preece, J. (2007). Interaction Design: Beyond Human-Computer Interaction. Wiley, 2007.

Shneiderman, B. & Plaisant, C. (2004). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison Wesley; 4 edition, 2004.

Spector, W. (1999). *Remodeling RPGs for the New Millennium*. Online article available at http://www.gamasutra.com/features/game_design/19990115/remodeling_01.htm.

Vygotsky, L.S. (1978). Mind and society: The development of higher mental processes. Cambridge, MA: Harvard University Press.

Wolf, M. & Perron, B. (2003). (Eds.) *The Video Game Theory Reader*, Oxford: Routledge.