Designing Games with Patterns

Daniel Cermak-Sassenrath

Abstract

This article discusses pattern design as a methodology in interaction design. The paper sketches out Alexander's idea of design patterns and describe our experiences with using patterns for designing old-school action games in the classroom context. The process included finding patterns, making a language, using it for creating several game designs and realising one of these designs collaboratively. The article grew out of a University course titled *A Pattern Approach to Action Game Design*, which was offered as an elective in the Creative Technologies program at Auckland University of Technology, New Zealand, in 2011. We present the concept of the course, our pattern language and the game we made. While the language is arguably more like a patchy pattern collection, the various game designs quite loose and the realised game unfinished, the process was challenging and intense, and offered students a new perspective on design. In the spirit of design patterns, we only did what the task at hand required. We attempted to connect theory and practice in a natural, direct way as we presented, discussed and used everything we did in order to continue our journey. Our course was aimed at a process that was constantly in flux through collaboration by people who interact and share a common pattern language, use, test, revise and refine it while moving on.

General Terms: design, experimentation, languages, theory

Keywords: pattern design, pattern language, design methodology, game design, action games, teaching

Introduction

Media, artifacts and processes of any complexity are structured by patterns. This observation was the starting point for our course on design patterns in interaction design. In the course we explored how patterns can be used generatively to inform creative processes for the design of interactive systems. We were using Christopher Alexander's concept of design patterns (Alexander, 1979; Alexander et al., 1977; Alexander et al., 1975). Each Alexandrian pattern consists of a triplet describing a specific situation, a problem and a proposed solution. Patterns are made to function in a certain structure, called a language, which informs and orders their application.

While Alexander created his pattern language for the domain of architecture, we aimed at computer action games. The focus was on designing game play, not on coding. Alexandrian design patterns have been applied to a range of different domains. They also have already been applied to interaction and game design (cf. Simpson, 1998; Church, 1999; Borchers, 2001; Kreimeier, 2002; Björk et al, 2003; Kreimeier, 2003; Lemay, 2007; Nystrom, 2010). We could have used (part of) an existing language for computer game design (for example, Björk & Holopainen, 2005) but we preferred to attempt to cover the whole process of identifying patterns, formulating a language and using it ourselves. This was done to create a more engaging and challenging situation to learn, to facilitate understanding of the theory, and also to strengthen the feeling of identification with what we did with a sense of discovery. While risking making a pattern collection instead of a complete language and not reaching a very high grade of abstraction or depth, we valued the process more than the product.

We analysed classic 8-bit and 16-bit computer games, mostly C64 and Amiga games, as a large number of these is readily available through emulators (*VICE* for C64 on Mac, *WinUAE* and*WinFellow* for Amiga on PC). It appears that the patterns are identical in old and new games, but the patterns are easier to spot in games that are technically limited to the essentials (i.e. interaction), than in the latest quite elaborate and complex games. We limited the scope of our approach to certain types of action games, and we chose to include jump 'n' run (e.g. *Mario*,*Great Giana Sisters*), shoot 'em up (*Xenon 2*) and maze games (*Gauntlet*), but excluded sports (*Kick Off*, *Projectile*, *Speedball*), race (*Super Cars*), karate (*IK+*) and sniper games (*Cabal*).

In this article, we describe the experiences in anecdotal form from a lecturer's point of view; a formal evaluation of the results has not been attempted. Verbal and written student feedback was collected and is presented in the discussion section.

The BCT Programme

The course *A Pattern Approach to Action Game Design* was offered as an elective to year 2 and 3 students of the Bachelor of Creative Technologies (BCT) programme at Auckland University of Technology, New Zealand, in semester two, 2011. The programme was only established in 2008 and is located within the Interdisciplinary Unit which was formed to develop new experimental alliances, research collaborations and student-centred learning experiences across the overlapping disciplines of Art & Design, Computing & Mathematical Sciences, Communication, and Engineering. The research-led BCT is seen as a key component of this interdisciplinary project.

While the programme seeks to shift the traditional focus from teaching-by-transmission to a more socialised notion of learning through collaboration, the day to day experience of our 'collaboratory' – an experimental studio environment for nurturing collaboration and creativity – has also shown us that such learning spaces will foster complex, subtle, dynamic, risky and opportunistic relationships; between academics and students as agents in a network of competing methodologies, knowledge domains, technological developments, skills and applications.

Within this new learning context, institutional power relationships and individual agencies of teacher, learner, practitioner or peer are frequently inverted, or may become fluid and provocative enough to challenge traditional pedagogical expectations and institutional structures of authority. Given the current international interest in both 'creativity' and 'collaboration' in numerous educational policy documents and graduate profiles, it is appropriate and timely that the programme initiates both critical and playful engagement with these themes – not least by challenging institutional praxis.

The Idea of Design Patterns

In this section the entities and the structure of patterns, the processes of their creation and application, as well as their interactions and relationships as described by Alexander are discussed briefly. A rough understanding of the concept of design patterns will then enable us to project the idea unto the development of digital games. The process of applying patterns is not a selective or exclusive process but all mentioned facets function in unison, and played a role in our experiment.

Following Alexander, patterns are seen here as rules of thumb.[1] For instance, when making a barn, build it 'in the shape of a rectangle, 30–55 feet wide, 40–250 feet long, the length at least 3x feet, where x is the number of cows the barn has to hold'.[2] "Include a wide double door for the hay wagon". "Divide the inside of the barn into three parallel aisles: two cow milking aisles down the outer sides, and a central hay-storage aisle" etc. A Gothic cathedral is made from a "nave *flanked* by aisles which run parallel to it", a transept, which "is at *right angles* to the nave and aisles", an ambulatory, columns, vaults, etc., all of which are in certain relations to each other and to the whole. While there are myriads of variants and there is constant flux in all systems, they retain a certain kind of invariant "character, a 'thing,' a 'structure,' which remains the same". Although "every church is different" *something* "remains the same, from church to church" and this is what "we call 'church'". "[T]he 'organism' [of a living system such as a town, neighborhood or building] is not so much an object, but the character of the invariants behind the flux". This invariant structure of

[1] There is a range of notions which are similar and related to Alexander's use of the word pattern, and which are commonly used in different contexts; for instance, motif, meme, code, theme, topic, subject, model, block, and part.
[2] All quotes in this section are taken from Alexander (1979).

entities and relations between them is the area in which design patterns work. They control and (trans-)form it. In effect, what we call a church is a selection of patterns in specific relationships with each other. A pattern language for a certain type of building consists of elements and relations between these elements, and “is really nothing more than a precise way of describing someone’s experience of building”, an explicit way of notation for design principles.

Patterns attempt to express invariant concepts which apply to specific problems in certain situations. Each pattern is thus formulated as a ‘three-part rule’ “*which establishes a relationship between a context, a system of forces which arises in that context, and a configuration which allows these forces to resolve themselves in that context*”.

Patterns are basic, deep, potent, simple, ordinary and easy to understand. Every pattern “is so concrete, so clearly expressed as a rule, and as a thing, that anyone can make one, or conceive one, in the buildings where he lives, or in a building which is going to be created”. It takes only little time to “*design a building in this way*. […] The speed is the essence. It takes time to learn the language. But it takes no more than a few hours or days to design a house”. “The depth, and spirituality, of the [idea of design patterns] is not made less by the fact that [these rules] can be expressed, nor that [they are] so simple. What matters, simply, is that [these rules are] extremely deep, extremely powerful.” The power to create “lies […] in the simple mastery of the steps in the process, and in the definition of these steps”. But when selections of these simple elements are combined and integrated, they “generat[e] an entirely unpredictable system of new and unforeseen relationships” and complex systems.

We are always dealing with a system of patterns; “patterns are not isolated” but “interdependent, at many levels”. Patterns interact with each other, in a system of relationships. “Each [pattern] is incomplete, and needs the context of the others, to make sense.” Large patterns give small patterns a place and put them in a certain relation to the whole, and the small patterns realise, facilitate and support the large patterns. Every pattern in a system “becomes whole in its own terms, because it is adapted to the larger wholes which it is part of, and because it is adapted to the smaller wholes which are a part of it”.

The realisation of patterns depends on the specific context. Patterns are relations of relations in many variations; they are not just physical parts, stackable objects or building blocks which are repeated identically. Patterns need to be fitted to specific settings for best results. A system which is alive or anything beautiful cannot be made “merely by combining fixed components” or “*by adding preformed parts*”. It can only be generated by a process in which “each part is modified by its position in the whole”, and is “different every time [it] occur[s]”: “Each bench, each windowsill, each tile, needs to be made by a person, or a process, in tune with the subtle minute forces there, making it a little different at each point along its length and different from all the others”. While each realisation is uniquely tailored to a specific situation, “[t]he patterns repeat themselves because, under a given set of circumstances, there are always certain fields of relationships which are most nearly well adapted to the forces which exist”. “The similarity of parts occurs because the forces which create the parts are always more or less the same. But the slight roughness and unevenness among these similarities, come from the fact that forces are never exactly the same." The focus on situated activity and the appropriation of specific settings and adaption to certain situations and circumstances, and the unity of action and space appear to point to parallels in phenomenology (e.g. Suchman, 1987).

The process of adaption to specific local circumstances also favours a holistic approach of planning and making, giving up the division of mind and matter. It attempts to connect reasoning and acting in a natural way. Planning informs the making and the practice feeds back into the planning. "*[…] the details of a building cannot be made alive when they are drawn at a drawing board.* […] The person who draws a working drawing cannot draw each window, or each brick, differently, because he has no basis for knowing the subtle differences which will be required. These only become clear when the actual building process is already under way." A design can be created on location, as close as we can get to the actual situation and to the shared, collectively experience: "To make the building live, its patterns must be generated on the site, so that each one takes its own shape according to its context".

When creating a design using a pattern language, every single pattern is to be made as intense as possible. "*There is no reason to be timid.*" This process is not about compromise, i.e. weakening two patterns and adding them both half-heartedly, but about creating one strong pattern at a time and relating it to all the other patterns that are already in place in the system, to "go all the way with it". Multiple patterns in one place take not away from each other but complement, enrich and balance each other.

The act of putting a pattern into a system is an act of integration, not addition; it is a fluid process, in which each pattern has the power to "transform […] the whole design created by the previous patterns". Patterns "are not parts, which can be added – but relationships, which get imposed upon the previous ones, in order to make more detail, more structure, and more substance". This is obvious in game design; a single feature transforms the whole game. As buildings need to be in balance with the activities which happen in them, games need to be in balance too, that is, no action of the player, feature or situation in a game should imbalance the whole system, e.g. a dominant winning strategy. Games, as buildings, need stability, i.e. rhythms or cycles of activity, that are contained, encompassed and expressed inside the system. The parts the design is composed of "overlap and interlock to such an extent that the oneness of all things becomes more marked". The whole design is transformed with each new pattern which is introduced, and, in turn, each new pattern is also transformed by the patterns and the structure which is already is place. "Each [building] process (given by a pattern) takes the configuration which has been produced by the previous processes, and adapts itself to them. No matter where the columns are, the process of weaving a vault can form the vault *according to*the position of the columns. No matter where the edges of a window are, the process of making a window forms the window and its panes *according to* the size and shape of the window frame." The observation that every act is to be seen in relation to what has already happened connects to Heidegger's notions that acting comes first, and of being thrown into the world, and not being able to step back. We are part of what we perceive and standing in the middle of it: "*Dasein* exists in the world and 'is engaged before it is reflective" (Coyne, 1998; Crogan, 2011). Living is always already acting and taking part.

The concept of design patterns sees design as a process in which the whole precedes the parts. The design stays whole during the entire process, while it is being differentiated, or rather, while the patterns in it differentiate themselves. While keeping the system whole in this process, "structure is injected into the whole by operating on the whole and crinkling it, not by adding little parts to one another". Everything is connected for the whole time. "In the process of differentiation, the whole gives birth to its parts: the parts appear as folds in a cloth of three dimensional space which is gradually crinkled. The form of the whole, and the parts, come into being simultaneously." The design starts and develops as a single entity. "There are no gaps between the parts, because each gap is just as much a part itself. And there are no clear divisions between levels in the structure, because, to some extent, each part reaches down, and is continuous and integral with smaller units of structure, which, once again, cannot be lifted out, because their boundaries overlap, and are continuous with larger units."

The process of using design patterns for building has been discussed for making new buildings so far. But "*there is a second, complementary process which produces the same results, but works piecemeal, instead*. When a place grows, and things are added to it, gradually, being shaped as they get added, to help form larger patterns […] the gaps are filled, the small things that are wrong are gradually corrected, and finally, the whole is so smooth and relaxed, that it will seem as though it had been there forever". It is the same process at work as before, when making something new, but "stretched out in time".

A pattern language is to be "*morphologically and functionally complete"* for a specific task (e.g. building, blues music) (Borchers, 2001). "It is morphologically complete, when the patterns together form a complete structure, filled out in all its details, with no gaps. And it is functionally complete when the system of patterns has that peculiar self-consistency in which the patterns, as a system, generate only those forces which they themselves resolve – so that the system as a whole, can live, without the action of self-destroying inner conflicts." In creating a design, you can let go of your control over it and "let the pattern[s] do the work. […] If the pattern[s] make […] sense, you do not need to control the design" from the outside. Then "the language, with very little help, is able to do almost all the work, and […] the building shapes itself". In this process of design nothing is to be added "except just what the patterns demand". This brings out the "natural, necessary order of a thing". We "must start with nothing in [our] mind[s]", and be "comfortable with the void, […] confident that the laws of nature, formulated as patterns, […] will together create all that is required". If we already have all the answers before we start our work we cannot listen to what the design asks for. If a design is completely "filled with the will of its maker […] there is no room for its own nature". A system which follows internal rules only is free from contradictions that weaken it; it can be pure and strong and true because it is at peace with itself, "in tune with its own inner forces". The specific realisation and results of such a process "must be unpredictable, so that the individual acts of building can be free to fit themselves to all the local forces that they meet".

A town or a neighborhood is always in flux, constantly changing, not a finished or frozen product, just as an interactive system. Even more, "[t]here *is* no product […]: the building and the town, which live, are that incessant flux, which, guided by its language, constantly creates itself" [emphasis added]. Such systems are alive because they are tested and refined in use.

People can make, adapt, apply and share their pattern languages for the building and construction tasks they face. Anybody "*with a pattern language can design any part of the environment"* and is entitled to do so, as "*it is essential that the people do shape their surroundings for themselves"* because they as users know best: "[W]indows must be shaped by people who are looking out". Large systems such as towns are made up of "*millions upon millions of these tiny acts, each one in the hands of the person who knows it best, best able to adapt it to the local circumstances".* This applies to large systems and small: "Each detail has meaning. Each detail is understood. Each detail is based on some person's experience, and gets shaped right, because it is slowly thought out, and deeply felt."

A pattern language arguably provides a group a people with a means of effectively communicating, "*almost as if they had a single mind"* to collaboratively make a whole, single and integrated structure because, "with a [pattern] language, the assumptions are almost completely explicit from the start". Patterns invite discussion, because they "are not fragile – they are as solid that they can be talked about, expressed quite clearly", challenged and questioned. A pattern like the 'Entrance Transition' (Alexander et al., 1977) "can be shared, precisely because it is open to debate, and tentative. Indeed, it is the very fact that it is open to debate, that makes it ready to be shared". But using patterns is not a mechanical process, a guarantee of anything or a magic bullet for success. "Pattern languages are the source of beauty and of ugliness. They are the source of all creative power: nothing is made without a pattern language in the maker's mind; and what that thing becomes, its depth, or its banality, come also from the pattern language in the builder's mind." The patterns are only as capable as the people who use them.

To test a system or pattern, Alexander argues for querying and trusting people's feelings as humans and users, and not for asking experts' opinions or blindly following fashions. Everybody involved in the process of design "can decide for himself whether [a pattern] is true, and when, and when not, to include it in his world". To judge a pattern he suggests to go to a town, building or place, where the pattern in question is implemented, "and [to] see how [we] feel there", to ask why we like something or not, and to try to identify and isolate the core of this experience. This will accurately tell us all we need to know about the pattern. "If a pattern does make [us] feel good, there is a very good chance that it is a good pattern. If a pattern does not help [us] to feel good, there is very little chance that it is a good pattern." This is not asking for our opinions or tastes but purely for feelings. "These feelings which are in touch with reality are sometimes very hard to reach. […] It is not hard because the feeling is not there, or because the feeling is unreliable. It is hard, because it takes an enormous and unusual amount of attention, to pay attention for long enough to find out which [pattern] does actually feel better." Alexander claims a very high rate of agreement in the cases he did this experiment with several people and the 'Window Place' pattern.

Using patterns is not teaching us anything new. But "*they only remind us of what we know already*", in our hearts, "old feelings", which we have forgotten and cannot access. A pattern "*language, and the processes which stem from it, merely release the fundamental order which is native to us".* It helps us "to come more into touch with the simple reality of things, and thereby become egoless and free" and "to be [ourselves]", to "act as nature does". In any design work people discover over and over again the same underlying, fixed and invariant principles, "based on fundamental realities, which everyone already knows, in his innermost self". Pattern languages express these principles explicitly and allow us to release our energy to simply do what we know is true and what needs to be done, "*what emerges from ourselves"*, without being held back by images, trends, fashions or expert opinions. Using patterns is not a goal in itself, and patterns are not cooking recipes that can or should be followed to the letter; they are concepts that need to be applied in the spirit in which they were conceived. When we have rediscovered the process which lets us get in touch with our ordinary, deep and "innermost feelings", the use of pattern languages has reached its end.

The Concept of the Course

The concept of the course was centred on the idea of combining theory and practice in a natural way. The practical work was to be carried by theory, and the discussion of theory to be informed by practical experiences. Exercises were not be artificial or detached from the design process, but everything we did was presented and discussed, and fed into the next step of the process. All participants worked collaboratively on a range of tasks differing in scope, difficulty and priority. This provided ways of engaging everybody, giving all participants ample opportunity to identify with the process and to make it their own project. Everybody could discover what he/she could contribute, try out new things, learn and take risks. The participants were aware that a course like this could quite easily go wrong; it was conceived as an exploration into the unknown, and this added a sense of discovery, surprise and thrill.

While the teaching time was formally divided into lectures, tutorials and lab sessions, in practice, the distinctions were fluent. In the lectures, theory, examples and our experiences were discussed, and students presented the results of the exercises.

Figure 1. Pong screen shots

The collective work in the lab included playing classic games with emulators, finding patterns, creating our own game designs, coding and testing. We started by implementing our own *Pong*and *Tron* versions (Figure 1–Figure 3) to get going. Usually, we started to work together on an exercise right after the lecture. This provided an immediate positive hands-on experience. It made people feel part of the process and helped to reduce both the distances between the topics of the lecture and the own work, and between the participants. Everybody experienced that they can do something, create and be part of the process. It also helped to reduce the amount of time before people actually started to engage. The practical work was to trigger the need for theory, create questions, and offer experiences that could be discussed in the lecture. This made the theory appear real and give the practice background and value, and also provided it with reasons and goals. We favoured group work over individual work, not only to encourage collaboration and motivation, but also to encourage learning from one another and the opportunity to challenge each other's ideas.

Figure 2. Tron title screen and in-game screen shot

There were ten exercises, nine of which had to be completed to pass the course. Some of the exercises were to be done individually, the rest in 3-person teams. Part of most exercises was a presentation in class and a hand-in (i.e. a pdf or source code). Usually, the topics of the lectures trailed the exercises by one week to enable students to first make their own experiences to which they then could relate when discussing theory in the lecture. The course relied heavily on students participation, so expectations in this department were high.

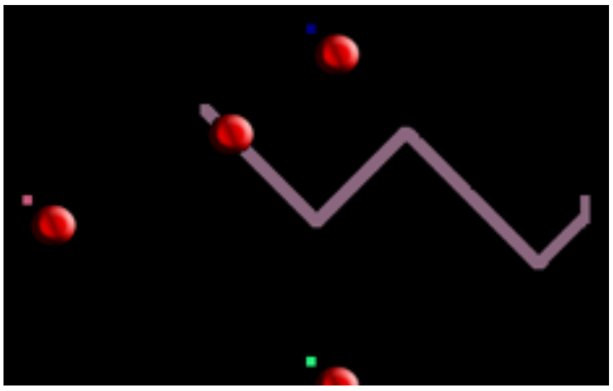


Figure 3. Diagonal Tron

An Action Game Design Pattern Language

In our course, we created a pattern language for action games, i.e. jump 'n' run, shoot 'em up and maze games, excluding sports, race, karate and sniper games. Our patterns roughly follow the format of Alexanderian patterns. Each one has a name, a certain context or situation, a short description, and is connected to larger patterns above and smaller patterns below. In many cases examples of an occurrence of a particular pattern in a game are given. All participants worked collaboratively on this language. They wrote, edited, moved, revised and deleted patterns. While a number of patterns were identified and described, the result is more like a collection than a complete language. The overall number of these patterns still needs to be reduced, the hierarchy needs to be revised, and patterns need to be linked and related to each other.

The titles of our patterns are *In-Game Objectives, Morality, Something To Do For The Player, Action Consequences, Reward For Risk, Character, Special Abilities, Weapons, Armor, Enemies, Civilians, Limited Lives, Health Bar, Manage Character, Power-Ups, Items To Collect, Shop, A Setting For The Game, Level Themes, Secrets, Hidden Chambers, Invisible Goodies, Shortcuts, Cheats, Trap, Competition Between Players, Quick Movement, Linear Flowing Gameplay, Everchanging Environments*, and *Game Gets Harder*. Because the pattern collection is too large to be included here completely, three example patterns of different abstraction are described below.

Reward for Risk

One of the most common patterns in action games is *Reward for Risk*. It differs from most other patterns in that it is an abstract pattern – it describes a style of gameplay rather than an actual object in the game.

Why is *Reward for Risk* such a useful pattern to implement in action games? Because it creates a psychological hook for the player. The human brain is wired so that if we successfully complete something risky, we get rewarded with a short burst of positive endorphins, along with an immense feeling of relief and satisfaction. Very quickly, the player gets addicted to this short emotional high, and is willing to invest significant time in a game to experience it. While this pattern is characteristic of the gambling genre, it is also an essential pattern for action games as it keeps the player engaged with the game. Examples of the *Reward for Risk* pattern include having to risk your life against a difficult boss to beat a level, being able to cross a dangerous lava pit with the potential reward of an extra life, and fighting more challenging monsters to get better loot. The risk is something that has to be balanced carefully if the reward is critical to the main gameplay; if a game is too hard to complete then players will quit. If a game is too easy then players will get bored. However, if the reward is something that the player does not necessarily need to finish the game (as in Figure 4), then the risk can be as high as the game's designer chooses.

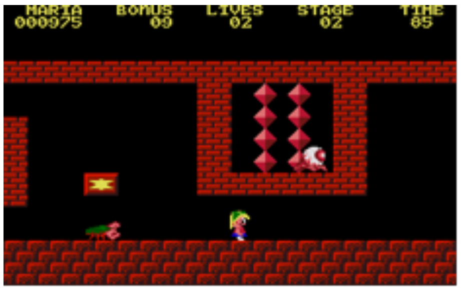


Figure 4. Great Giana Sisters: Releasing the monster and trying to collect the diamonds for a high score?[3]

[3] www.oldgamesclub.com/snap/amiga/greatgianasisters.png (7/6/2012)

Shop

In a game with many different enemies and/or levels, it might be interesting to offer the player the possibility to choose his own weapons. Players can then buy and sell weapons and other equipment, and if the prices vary between shops, they can even trade with them. Shops might vary in offer and price. Shops can be located anywhere in a level, but most commonly between levels or at the halfway point. Most shops in games of this type contain a very basic interface. There is usually an easy-to-navigate scrollable item menu – either filling the screen, or over a graphic depicting a shop counter. Sometimes, however, a shop will only appear as an options screen or dialog box after a particular action has been completed, asking whether or not you want to buy or upgrade something. Upgrades to weapons, armour or vehicles are usually available, and better enhancements cost more. Other items that can often be bought from shops include ammunition, damage boosters, and health items. Items from the shop are usually paid for with items collected in levels, or using an in-game currency that collectables or score can be exchanged for. Therefore, a *Shop* is incorporated into the game to add an element of strategy to the action game, and give the player control over his abilities and equipment. Offers and price can be varied between shops to enable trade and are usually placed at the end of levels or at the half-way point.

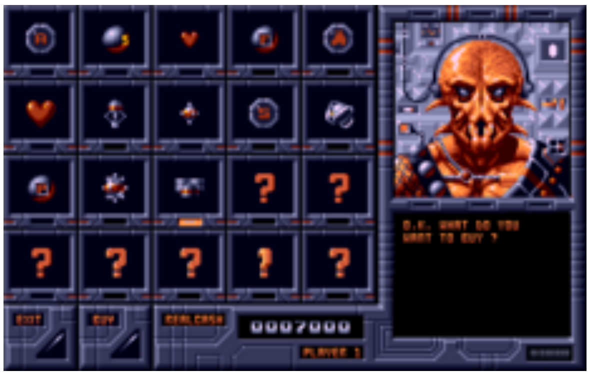

O.K. WHAT DO YOU WANT TO BUY ?

Figure 5. The shop in *Xenon 2* [4]

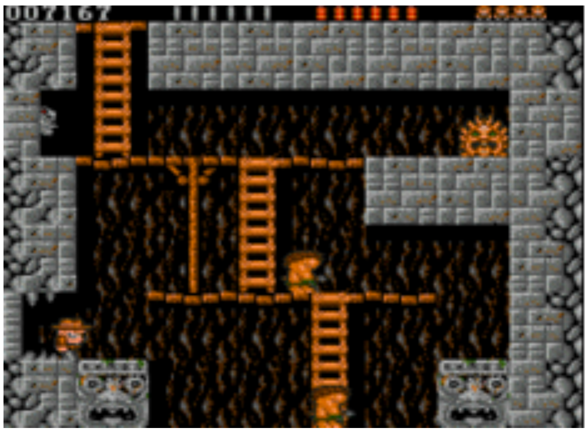


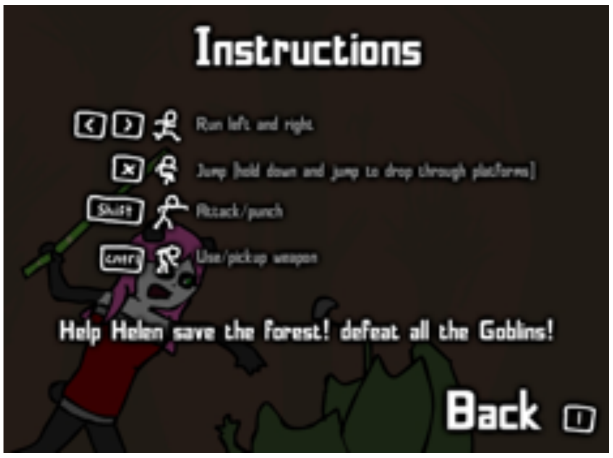
Figure 6. A very cleverly placed trap, left on the upper level in *xrick*, a clone of *Rick Dangerous* [5]

[4] Image slightly modified from kingofgng.com/media/20120205_xenon2_2.png (7/6/2012)

Trap

In addition to enemies, traps can be dangerous to players. They can be easy to see or hidden. There are many different types of traps. Most traps are part of the level and cannot be defeated or destroyed, simply avoided. Traps are incorporated in games to add a sense of discovery. Players will then carefully observe every detail of the level design. Traps should be visible (as in Rick Dangerous, Figure 6), and not only be found by trial-and-error (as in Lost Vikings). Traps can also be dangerous for enemies, therefore enabling the player to use them to his advantage, adding a twist to the game beyond shooting at everything that moves. There should be a reason for the trap, and a payoff for defeating it, such as a bonus.

Figure 7. Super Bush! Title screen


Instructions

< > Run left and right

x Jump [hold down and jump to drop through platforms]

Shift Attack/punch

enter Use/pickup weapon

Help Helen save the forest! defeat all the Goblins!

Back 1

Figure 8. Super Bush! Controls screen

Super Bush! Chronicles

The game we made in the course is a single-player jump ‘n’ run game (see Figures 7–11) controlled with the keyboard, and incorporating original sound and graphics. It was implemented using *C++* and the *SDL* library.[6]

---

[5] linuxreviews.org/games/classics/xrick.png (7/6/2012)

[6] A PC download of the first level is available at dace.de.

A panda bear is defending its jungle against fierce goblins who want to build a town at this location to support their gambling needs. The game includes a number of patterns from the devised pattern language: *Something To Do For The Player, Action Consequences, In-Game Objectives, Reward for Risk* (score, goodies), *Character* (panda bear), *Morality* (helping a good cause, defending the forest, liberating caged jungle animals), *Special Abilities* (jumping very high and bamboo stick kendo, eat weapon upgrades (bamboo stick) to boost health), *Opposing Force*(goblins with axes, chainsaws or guns, boss goblins in construction vehicles, i.e. bulldozers),*Limited Lives* (three), *Health Bar* (for player and goblins), *Manage Character, Items To Collect* (nuts as currency), *Shop* (buy armour and weapon upgrades for previously collected nuts), *Competition Between Players* (through saved highscores), *Game Gets Harder* (increasing number of enemies and traps), *Trap, A Setting For The Game* (conflict over jungle resources) and *Level Themes* (five levels with slightly different themes).

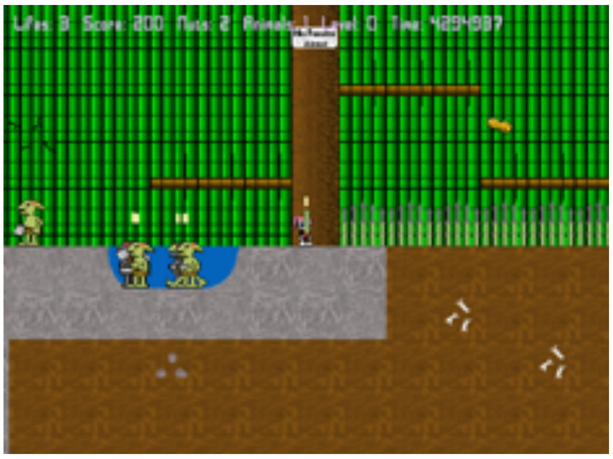


Figure 9. In-game screen shot

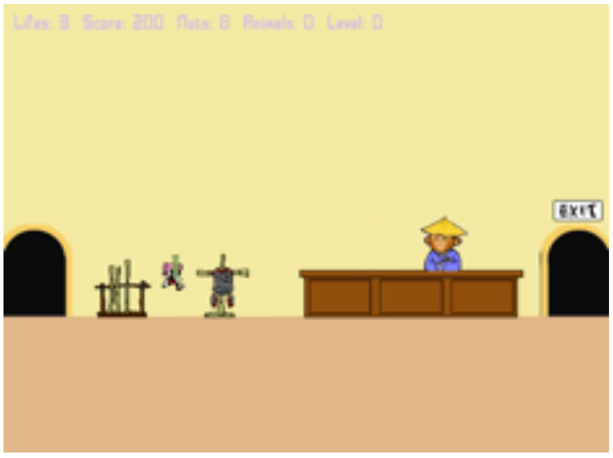


Figure 10. Weapon and armour shop between levels

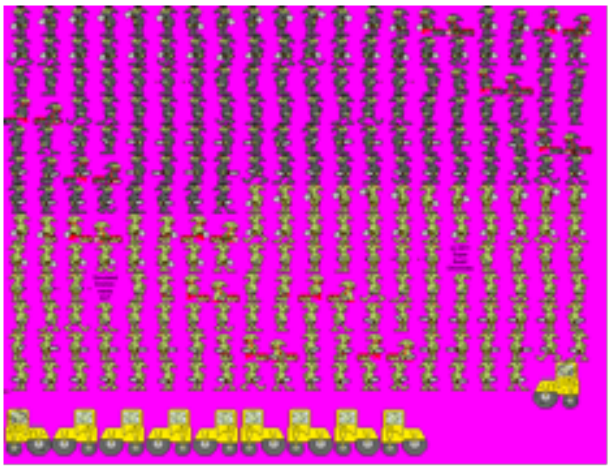


Figure 11. Goblin sprite sheet

Discussion

At the end of the semester, students were invited to give verbal feedback about their experiences and opinions on the use of design patterns. Additionally, views expressed in the reflective statement (the final exercise) are collected here.[7] Students commented on different aspects of pattern design, some specific to their experience on our course, some more general.

Students found the process of playing classic action games, looking for patterns and identifying their essential properties enjoyable and rewarding. It provided them with a sense of discovery and ownership. "[…] I thought it was going to be difficult to find the patterns. Instead, I found that once I started looking for the patterns, they were absolutely everywhere." Many said it opened their eyes to the concept of patterns. "In exercise 4 we were allocated with our first opportunity to genuinely identify and describe patterns present in specified games. Consequently, patterns have become increasingly consciously active and present in my mind. Prior to this I was completely oblivious to the presence and importance of patterns in gaming." Students also commented that this was the point they began to understand what we were talking about in the lecture. "Much of the class came up with similar patterns so we classified them and worked towards building a final pattern language. This helped us all tune in with each other on what a pattern language actually was and how things should be categorised. It worked well." We used a free real-time multiuser online text editor (*Etherpad*, n.d.) for working on the pattern language. This facilitated a feeling of an ongoing process among the participants, because everybody could always access the latest version of our pattern language, use it and change it. "With the online collaboration tool Etherpad we were able to alter and read the document in real time as edits were being made, and see the formation of a document. We were able to influence each other and be influenced as the document took shape."

Students enjoyed creating their own game designs. At that stage of the course, the pattern language was still very loose, and a number of patterns were added to the

[7] All quotes in this section are taken from the reflective statements or the design documents written by the students.

designs as afterthoughts. Nevertheless, students were aware of patterns, and all used them to some degree. “The most interesting part was when we all had to come up a game design idea to present for an action game.” Several game designs were created, presented and discussed by the students. All the game designs were for jump ‘n’ run games, which was surprising (apparently *Xenon*-style shoot ‘em up and *Gauntlet*-style maze games are out of fashion at the moment). There were no radical designs, but more detail tweaking, copying popular games and some transformation. Team work was mostly limited to two-person pairs, which was unexpected, and some people even preferred to work alone on this exercise. Among the game designs were *Dragon Eggs*, a *Mario*-esk ‘Action Platformer’ with Medieval and Fantasy themes; *Radical Hamster Force*: “[k]ind of like a mix of Alex the [K]id, Mario, and Kirby combined but with optional weapons”; *Krystal in the Hood*, a ‘classic platform game’ in which the player has to ‘move from left to right and from top to bottom’ through multiple levels; and the later realised*Super Bush! Chronicles*. We voted for one of these game designs to be realised, a process that was appreciated and taken seriously.

The single most successful aspect of using patterns was arguably their benefits for group work. “Everyone managed to work together in teams and made the best out of their abilities to achieve the game.” The use of patterns helped quite a heterogeneous group of more than ten people to identify and actively engage with a single project, and facilitated collaborative decision-making in our meetings. Contributions were very specific and to the point when discussing the game designs. The concept of patterns made the group focus on the process, and not only on the results. “I don’t think it would have been a bad or significantly different game if we hadn’t taken the process of developing and deciding on patterns prior to writing up a game design document. But it helped the process and I think it allowed for a more concrete development of the project. Using patterns you know what you want, then design around that idea.”

However, while patterns seemed to work well for creating game designs, it was different during the implementation phase, in which getting the game working became the focus instead. Students remarked that implementing the game simply had not much to do with the idea of using patterns. “The [pattern] language helped the game designers to not miss important parts of a game out and to think about how parts interacted. From there however, we stopped thinking about patterns.” Coding *C++* proved to be difficult as most people were inexperienced coders. Getting the basic functionality right was challenging enough, and possibly hindered access to the process on a higher, more abstract and interesting level. “While [C++ is] fast and powerful there are certain low level elements that one cannot avoid using.” For example, trying to get the sound working “was a waste of time that I didn’t have to waste”. Students suggested using a game engine instead. An advantage in the implementation phase was that everybody working on the code knew what, why and when something needed to happen. Students could be quite specific about what they wanted to do, and what they wanted other people to do. “At first I thought that the patterns weren’t going to be necessary, but as we progressed I found them to be quite essential in terms of laying out the game – because we knew what we wanted, where to implement it, how it worked and it was all written down and discussed with the group.” Despite the difficulties, people commented that making the game was a fun and very intense experience, and that it felt ‘magic’ to see how the patterns came to life.

Students came to see patterns as an interesting design methodology. They described them to be a very useful tool, a “powerful […] development technique”. But initially, they were sceptical of the generative force of patterns. “How could that possibly work? Sure, it was fascinating to look at already completed games and see how they could be broken down into patterns, but I didn’t completely believe that the reverse could be done – taking patterns and creating a game from them.” During the semester, students were questioning the idea of letting go and not trying to control the whole design top-down. They were surprised by the drive and the immediacy with which patterns asked for action. “[…] honestly I didn’t feel like it was going to work. I thought that something bad was going to happen that would stop the progress of the project and slow it down for everyone […] but fortunately I was mistaken.”

A Critique of Pattern Design

Today, pattern design appears to be a modern concept in a postmodern era. It is centred around a constructive positivism that accepts an all-encompassing system as an unquestioned given. It requires a systematic framework to function and is also, in turn, creating systems. Of course, these systems are potentially huge, with lots of variations and exciting possibilities, but patterns will never question, challenge or disorder their own systems by inversion, perversion or subversion. This is, obviously, their beauty and essential part of their quality and appeal. While pattern languages are defined by their everyday and ordinary use through their users, and are constantly challenged, developed, adapted and changed, and pattern design is a productive, effective and fascinating methodology that remains focussed on problem-solving. Attempts to make art will necessarily and by definition fall outside of its scope. Patterns are not catering to the illogical, contradicting, inspired, genius and weird ways of creating. Students articulated this aspect when they pointed out that they felt patterns “restricted [their] creative freedom”. As this was our first approach to patterns, the process may have been quite mechanical, and not fluent, spontaneous or radical. Our language was not deep and powerful. “[…] I feel my common sense and knowledge of how games work being much more helpful to me than using a set of rules. Creative decisions yielded better results and just experimenting until it feels right.”

Playwright Bertolt Brecht produced a number of model books [*Modellbücher*] containing draft designs for theatre productions. These books featured textual descriptions, commentaries, photos, and scenic arrangements to provide material for orientation and experimentation (Vassen, 2012). Similar to Alexandrian patterns, applying (e.g. changing, rejecting) the models was not intended to *replace* creative practice, but to enforce it (Brecht, 1991; quoted in Vassen, 2012). But while Alexander provides a carefully weighted, refined, balanced and harmonious system of patterns, Brecht offers rough material to work with; Brecht’s perspective appears to be much more open.

In a way, Alexandrian patterns are, as one student put it, ‘cloning’ successful solutions, and a person using a pattern language is “playing it safe”; not in the sense that a certain outcome*quality* is guaranteed, but in the sense that the *kind* of outcome is clearly defined *a priori*, depending on the pattern language used. Patterns have intrinsic limitations, and the limitations might be seen not as a fault of the method, but as a reason for its application. In our experience, patterns were most useful on an abstract level, accepting the limits of the system as the upper boundary, and the technical implementation as the lower. We used pattern design for idea development, group communication, planning, and generating project and program structure. We did not attempt to prescribe how to technically code different patterns as we felt that this was better left to the individual experience of the person doing the job.[8]

Conclusion

The course on design patterns was ambitious and challenging. We did not have much time to discuss theory, play games, find patterns, make a language, use it for designing action games and realise one of the designs. It was an intense journey, aiming to combine practice and theory, experience and reflection. The theory was in many cases the subject of the exercises, used to inform the practical work to a considerable degree. The practical work relied on an understanding of the theory, in a direct and natural way. We only did what the task at hand required and followed the internal logic of the process. We presented, discussed and used everything we did; everybody was aware why he/she was doing something and why this was necessary. The students were engaged and interested in the new perspective the concept of patterns could bring to their practice, and everybody was curious if it would work for us.

Of course, we had some problems that held us back. Most students were not experienced in game (or interaction) design, and many were novice coders. Only a few had an overview of classic action games. The students did not have a solid base of design knowledge that they could apply through the new perspective of design patterns. In creating and discussing the game designs, students did not feel the intrinsic necessity to strictly follow-through with design patterns; this points to our language being not complete, as new ideas were constantly suggested at all stages of the design process. On the other hand, there were solid benefits in using patterns. The collaboration was working well, and discussions were very specific and focussed. Students were keen to participate and many invested a lot of time and energy. Everybody could make a relevant contribution, and this enabled each student to become deeply involved and interested. Students had the feeling of genuine discovery and ownership. We did everything by ourselves, and all of us shared the process. The process of finding, describing and using patterns could have failed, and this added some thrill to the experience. The course was not over-prepared, we faced real questions, issues and problems and needed to find solutions as we progressed. To include the possibility of real failure opened up the possibility for real success.

[8] Another obvious critique of design patterns is the externalisation of knowledge and experience that they require; discussion of which goes beyond the scope of this article.

And from this lecturer’s point of view, it was fun. The group gained more from the process than we invested. The course developed a kind of momentum of its own. All participants saw how our various loose and general ideas for a game were transformed into a coherent design, and then, towards the end of the semester, how the design was turned into a working game, literally in the course of a few days (and nights). This was an impressive experience for everyone involved.

References

Alexander, C., Silverstein, M., Angel, S., Ishikawa, S., and Abrams, D. (1975). *The Oregon Experiment*. New York: Oxford University Press.

Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., and Angel. S. (1977). *A Pattern Language*. New York: Oxford University Press.

Alexander, C. (1979). *The Timeless Way of Building*. New York: Oxford University Press.

Björk, S., Lundgren, S., and Holopainen, J. Game Design Patterns. (2003). In *Level Up: Digital Games Research Conference*, 4–6.

Björk, S. and Holopainen, J. (2005). *Patterns in Game Design*. Hingham, MA.: Charles River Media.

Borchers, J. (2001). *A Pattern Approach to Interaction Design*. Chichester, UK: John Wiley & Sons Ltd.

Brecht, B. (1991). In W. Hecht (Ed.) *Große kommentierte Berliner und Frankfurter Ausgabe, 24.*Berlin, Germany: Aufbau/Suhrkamp.

Church, D. (1999). Formal Abstract Design Tools. In *Gamasutra*, July 16, 1999. Retrieved from http://www.gamasutra.com/view/feature/3357/formal_abstract_design_tools.php.

Coyne, R. (1998). Cyberspace and Heidegger's Pragmatics. *Information Technology and People,*11(4), 338–350.

Crogan, P. (2011). *Gameplay Mode. War, Simulation, and Technoculture*. Minneapolis, USA: University of Minnesota Press.

Etherpad. (n.d.) Retrieved from http://www.ietherpad.com.

Kreimeier, B. (2002). The Case For Game Design Patterns. In *Gamasutra*. March 13, 2002. Retrieved fromhttp://www.gamasutra.com/view/feature/4261/the_case_for_game_design_patterns.php.

Kreimeier, B. (2003). Content Patterns in Game Design. *Game Developers Conference 2003*. International Game Developers Association.

Lemay, P. (2007). Developing a pattern language for flow experiences in video games. In *Situated Play, Proceedings of DiGRA 2007 Conference* (pp. 449–455).

Nystrom, R. 2010. *Game Programming Patterns / Getting Started / Introduction*. Retrieved on July 8, 2010 from http://gameprogrammingpatterns.com/introduction.html

Simpson, Z. B. (1998). Design Pattern for Computer Games. In *Computer Game Developer's Conference*. Retrieved from http://www.mine-control.com/zack/patterns/gamepatterns.html.

Suchman, L. A. (1987). *Plans and Situated Actions: The Problem of Human-Machine Communication*. Cambridge, UK: Cambridge University Press.

Vassen, F. (2012). Bertolt Brechts Theaterexperimente. Galilei versus Lehrstück. In Kreuzer, S. (Ed.). *Experimente in der Künsten. Transmediale Erkundungen in Literatur, Theater, Film, Musik und bildender Kunst* (pp. 91–130). Bielefeld, Germany: Transcript.

List of Images

This article is based on articles presented at The 2nd International Conference on Design and Modeling in Science, Education, and Technology: DeMset 2012, Orlando USA, March 25–28, 2012; and at The 8th Australasian Conference on Interactive Entertainment (IE) 2012, Auckland NZ, July 21–22, 2012.

The Journal of Creative Technologies (JCT)
Issue 3: Interactivity, 2013

ISSN: 2230-2115

Colab, Auckland University of Technology, New Zealand

JCT is a research communication platform published by Colab at the Faculty of Design and Creative Technologies, Auckland University of Technology.
https://ctechjournal.aut.ac.nz