A Design Pattern Language for Oldschool Action Games

Daniel CERMAK-SASSENRATH
Interdisciplinary Unit, Faculty of Design & Creative Technologies
Auckland University of Technology
Auckland, New Zealand

ABSTRACT

This article discusses the application of an Alexandrian pattern language to the design of interactive systems. It grew out of an 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, NZ, in 2011. We sketch out the idea of design patterns and describe our experiences with the process of using them for designing oldschool action games, that is, finding patterns, making a language, using it for creating several game designs and realizing one of these designs collaboratively. We discuss the concept of the course and present 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 realized 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, not artificial exercises. 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 not aimed at fixed or frozen products, but on a process that is constantly in flux through collaboration by people who interact and share a common pattern language, use, test, revise and refine it while moving on.

Keywords: Pattern design, Methodology, Game Design and 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 the course we explored how patterns can be used generatively to inform creative processes for the design of interactive systems: Design patterns "are dynamic. They have force. They are generative." [3]

While Alexander created his pattern language for the domain of architecture ([3], [2], [1]), we aimed at computer action games. The focus was on game play, not on e.g. graphics, physics or coding. We analyzed 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). The patterns are also identical in old and new action games, it appears, 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*, *R-Type*) and maze games (*Gauntlet*), but excluded sports (*Kick Off*, *Projectile*, *Speedball*), race (*Super Cars*), karate (*IK+*) and sniper games (*Cabal*).

The concept of Alexandrian design patterns has been applied to a range of different domains. They also have already been applied to interaction and game design (e.g. [13], [7], [6], [9], [4], [10], [11], [12]). We could have used (part of) an existing language for computer game design [5] but we preferred to attempt to cover the whole process of finding 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 do 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.

THE IDEA OF DESIGN PATTERNS

Following Alexander [3], patterns are seen here as rules of thumb. 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." Include a wide double door for the hay wagon. "Devide the inside of the barn into three parallel aisles: two cow milking aisles down the outer sides, and a central hay-storage aisle [etc.]." 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". This invariant structure of 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 certain relationships with each other.

A pattern language is an explicit way of notation for design principles. Patterns attempt to express invariant concepts which apply to specific problems in certain situations. They try to "relate [...] context, problem, and solution, in an unchanging way". Each pattern is 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; they are not mysterious or complicated or to be used only by specialists. 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 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 realize, facilitate and support the large patterns.

The realisation of patterns depends on the given 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]". 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 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. [14]).

The process of adaption to specific local circumstances also favors a wholistic 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 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." [3] A design can be created on location, as close as we can get to the actual situation and to the shared, collectively experience.

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 but about creating one strong pattern at a time and relate 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. 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. 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.

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." The design starts and develops as a single entity. "The form of the whole, and the parts, come into being simultaniously."

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, [...] 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." This process 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 [6]). "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." [3] In creating a design, you then only need to follow (invariant) internal requirements and logic, not (changing) outward and external images, trends, styles or pressure. You can let go of your control over the design and "let the pattern[s] do the work." In this process nothing is to be added "except just what the patterns demand". This brings out the "natural, necessary order of a thing". If we already have all the answers before we start our work we cannot listen to what the design asks for. 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." Such 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".

A town or a neighborhood is always in flux, constantly changing, not a finished or frozen product. 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." (emph. added) Such systems are alive because they are tested and refined in use.

People can make, adapt and share pattern languages "*for any building task [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 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 [2] "can be shared, precisely because it is open to debate, and tentative." [3] But using patterns is not a mechanical process, guaranteeing anything or a magic bullet for success. "Pattern languages are the source of beauty and of ugliness." 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 the core of this experience. This will accurately tell us all we need to know about the pattern. This is not asking for our opinions or tastes but purely for feelings. Alexander claims a very high rate of agreement in the cases he did this experiment with 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", what 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". 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 centered around the idea of combining theory and practice in a natural way. The practical work should be carried by theory, and the discussion of theory should be informed by practical experiences. Exercises should not be artificial or detached from the design process, but everything we do should be presented and discussed, and feed into the next step of the process. All participants work collaboratively on a whole range of tasks differing in scope, difficulty and priority. This provides ways of engaging everybody, giving all participants ample opportunity to identify with the process and to make it their own project. Everybody can discover what he/she can contribute, and try out new things, learn, take risks. The participants were aware that a course like this can quite easily go wrong; it was conceived as an exploration into the unknown, and this definitely added a sense of discovery, surprise and thrill.

While the teaching time was formally devided 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. The theory was mainly focussed on Alexander's *Timeless Way* and *A Pattern Language*, but we also looked at, for instance, Borchers' example of a pattern language for blues music. A number of articles on game design patterns were also referred to.

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) to get going. Usually, we would start to work together on an exercise right after the lecture. This should provide an immediate positive hands-on experience. It makes people feel part of the process and helps to reduce both the distances between the topics of the lecture and the own work, and between the participants. It also helps to reduce the amount of time before people actually start to engage. The practical work should trigger the need for theory, create questions, and offer experiences that can be discussed in the lecture. This makes the theory appear less artificial and give the practice background and value, and also provides it with reasons and goals. We favored group work over individual work to not only to make people collaborate with and motivate each other, but also learn from one another and challenge each other's ideas.

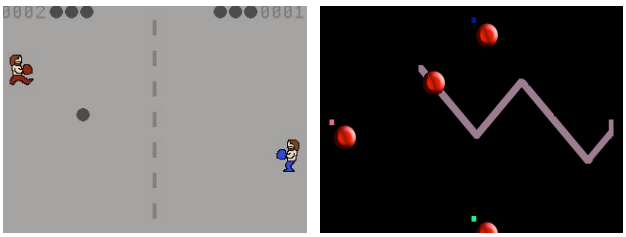

Figure 1: Boxing cavemen *Pong* and diagonal 4-player *Tron*

There were ten theoretical and practical exercises, nine of which had to be done to pass the paper. 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 paper relied heavily on students' participation, so expectations in this department were high.

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 final excercise, the reflective statement, are collected here. Students commented on different aspects of pattern design, some specific

to our course, some quite general. The discussion of how the concept of the course worked in practice is centered around the idea of creative and generative use of design patterns, not on everyday teaching and learning issues.

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. Many people said it opened their eyes to the concept of patterns. "[...] 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." (Reflective Statement) 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." The lecture clearly benefitted from this experience. We used a free real-time multiuser online text editor [8] for working on the pattern language. This facilitated a feeling of an onging process among the participants, because everybody could always access the latest version of our pattern language, use 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." We were constantly on the move on this trip.

Students enjoyed creating their own game designs. At that point, our pattern language was still very loose, and a number of patterns were certainly added to the designs as afterthoughts. Nevertheless, people were aware of patterns, and used them to some degree. And it was fun. "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. 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 really radical designs, but more detail tweaking, copying popular games, some transformation and sadly no multiplayer. Among the game designs were *Dragon Eggs*, a *Mario*-esk "Action Platformer" (Game Design Document) with "Medieval and Fantasy themes"; *Radical Hamster Force*, "Kind of like a mix of Alex the kid, 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 realized *Super Bush! Chronicles*. We voted for one of these game designs to be realized and integrated features (i.e. patterns) from the other designs. The voting was appreciated, and taken quite 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 numerous meetings. Contributions could be were very specific and to the point, e.g. in discussing the game designs. "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." Students came to see patterns as an interesting design methodology. They described them to be a very useful tool, a "powerful [...] development technique".

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 hard 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." Trying to get e.g. the sound working "was a waste of time that I didn"t have to waste." Students suggested using a game engine instead. An advantage was that everybody working on the code knew what, why and when something needed to happen, which was very helpful for collaboration. "Whoa, I thought, that's an interesting approach." People 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 quite magic to see how the patterns came to life.

Some students felt that patterns "restricted [their] creative freedom". As this was our first use of patterns, the process may have been quite mechanical, and not as fluent, spontaneous and radical as it could be. "The world is chaos and unorganized, putting everything into an organized list makes life boring and ends opportunity for innovation and creative thinking." Our language was not very 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". While the question of creativity and patterns was addressed in one of the lectures, it was apparently not discussed clearly enough. Patterns have nothing to do with "what I would call cloning", and a person using a pattern language is certainly not "playing it safe". Patterns are not only for people who "can't or don't need to think creatively". Of course, there is some truth to the matter. Patterns need a closed system to work, e.g. an engineering problem to solve. It would appear absurd to have a pattern language for creating art, for example.

Initially, people were sceptic 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."

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. Excluded are, although these are arguably also action games, sports, race, karate one-on-one 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 occurance 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 was 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 given.

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 rage 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, then the risk can be as high as you want.

Goes with patterns: ENEMIES, TRAP, INVISIBLE GOODIES, SECRETS

SHOP

In a game with many different enemies and/or levels, it might be interesting to offer the player the possibility to decide about what weapons he wants to have. Players can buy and sell weapons and other equipment, and if the prices vary between shops, they can even trade with them. Shops might be localized, e.g. 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 a basic, 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 you 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 game 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: Put a SHOP into your game when you want to add an element of strategy to the action game, and give the player control over his abilities/equipment. Vary offer and price between shops to enable trade. Place them at the end of levels or at the half-way point.

Goes with patterns: ITEMS TO COLLECT, POWER-UPS, WEAPONS, ARMOUR, SPECIAL ABILITIES

Examples: *Xenon 2*, *River City Ransom*

TRAP

Is part of the patterns: SECRETS

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.

Put TRAPS in your game to add a sense of discovery to it. Players will then carefully observe every detail in your level design. Traps should be visible (as in *Rick Dangerous*), 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 for 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, e.g. a bonus.

Goes with patterns: ENEMIES, REWARD FOR RISK

Examples: *Rick Dangerous*, *Lost Vikings*

ACTION GAME: *SUPER BUSH! CHRONICLES*

Super Bush! Chronicles (Figures 2 and 3) is a single-player jump n' run game. It is about a panda bear defending its jungle against fierce goblins who want to build a town at this location to "support their gambling needs" (Design Document). The game includes a number of patterns from our 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), ENEMIES (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), COMPETITION BETWEEN PLAYERS (through saved hiscores), GAME GETS HARDER (increasing number of enemies and traps), TRAPS, A SETTING FOR THE GAME (conflict over natural resources) and LEVEL THEMES (five levels with slightly different themes).

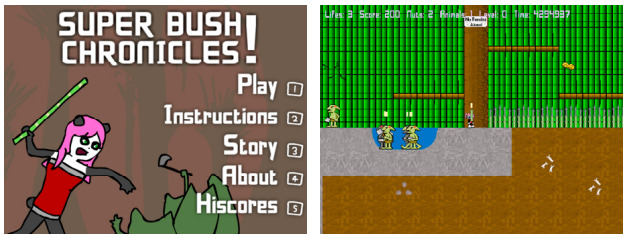

Figure 2: *Super Bush!* title screen and in-game screen shot

We recorded our own sound and drew original graphics. The game is controlled with the keyboard. It was implemented using *C++* and the *SDL* library. A PC download is available at www.dace.de.

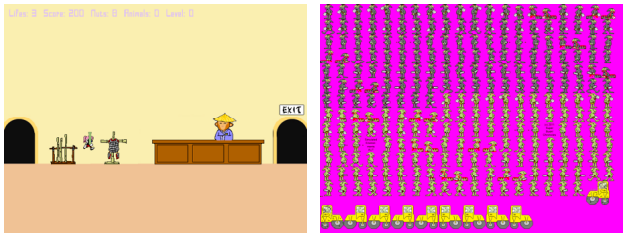

Figure 3: Weapon and armour shop, goblin sprite sheet

CONCLUSION

The course on design patterns was quite 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 realize one of the designs. It was an intense trip aiming to combine practice and theory, experience and reflection. The theory was in many cases the subject of the exercises, or informing 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 always 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 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 to the point. Students were keen to participate and many invested a lot of time and energy. Everybody could quite easily make a relevant contribution, and this got people deeply involved and interested. Students had the feeling of genuine discovery and ownership. We did everything by ourselves, and all of us shared the design and the process. The process of finding, describing and using patterns could have gone seriously wrong, and this definitely add some thrill to it. The course was not over-prepared. We faced real questions, issues and problems, and needed to find solutions. To include the possibility of real failure opened up the possibility for real success.

And it was fun. We gained more from the process than we invested, it appears. The process 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 turning into a working game, literally in the course of a few days (and nights). This was a quite impressive and also a strange experience.

REFERENCES

[1] C. Alexander, M. Silverstein, S. Angel, S. Ishikawa, D. Abrams, **The Oregon Experiment**, New York: Oxford University Press, 1975.

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

[3] C. Alexander, **The Timeless Way of Building**, New York: Oxford University Press, 1979.

[4] S. Björk, S. Lundgren, J. Holopainen, "Game Design Patterns", **Level Up: Digital Games Research Conference**, 2003, pp. 4–6.

[5] S. Björk, J. Holopainen, **Patterns in Game Design**, Hingham: Charles River Media, 2005.

[6] J. Borchers, **A Pattern Approach to Interaction Design**, Chichester: Wiley, 2001.

[7] D. Church, "Formal Abstract Design Tools", **Gamasutra**, July 16, 1999. www.gamasutra.com/view/feature/3357/formal_abstract_design_tools.php (16/6/2011).

[8] **Etherpad**, www.ietherpad.com (Nov 4, 2011).

[9] B. Kreimeier, "The Case For Game Design Patterns", **Gamasutra**. March 13, 2002. www.gamasutra.com/view/feature/4261/the_case_for_game_design_patterns.php (16/6/2011)

[10] B. Kreimeier, "Content Patterns in Game Design", **Game Developers Conference**, 2003.

[11] P. Lemay, "Developing a pattern language for flow experiences in video games", **Situated Play**, Proc. DiGRA 2007 Conf., 2007, pp. 449–455.

[12] R. Nystrom, **Game Programming Patterns** / Getting Started / Introduction. gameprogrammingpatterns.com/introduction.html. Last modified on July 08, 2010 (2/6/2011).

[13] Z.B. Simpson, "Design Pattern for Computer Games", **Computer Game Developer's Conference**, Austin, TX; San Jose, CA. Nov 1998, May 1999. www.mine-control.com/zack/patterns/gamepatterns.html (6/7/2011)

[14] L.A. Suchman, **Plans and Situated Actions: The Problem of Human-Machine Communication**, Cambridge: Cambridge Univ. Press, 1987.