

# Usability Patterns in Games

Eelke Folmer

Department of Computing Science, University of Nevada, Reno, USA

[research@eelke.com](mailto:research@eelke.com) | <http://usability.eelke.com>

*The ever increasing costs of developing games combined with a low chance of commercial success has forced developers to focus on producing fewer but higher quality games. Usability is an important aspect of game quality, but designing usable games is difficult and designers need usable and effective design tools based upon proven knowledge of design. Game design however, is still commonly implemented by applying esoteric heuristics gathered over many years of professional experience. Capturing the "art" of good game interaction design can help less experienced designers. Traditionally such design knowledge is described as guidelines or heuristics but these have several problems with regard to selection, validity and applicability. Patterns have emerged as a technique which can overcome these limitations as they specifically focus on the context and tell the designer when, how and why the solution can be applied. This paper presents usability patterns for games, a tool which can help interaction design for games.*

## 1. Introduction

Developing games is expensive and risky. Computer games have evolved significantly in scale and complexity since Pong was developed in the 70ties. Currently the average cost for developing a console game is estimated to vary between \$3 million and \$10 million [1]. In addition, the games industry is predominantly hits driven; A UK demographics study revealed the top 99 titles (only 3.3% of development) account for 55% of all sales [2].

High costs coupled with a low risk of success forces developers to focus on producing fewer, but higher quality titles. The success of a game depends on providing innovative game play, a good storyline, nice artwork, realism, reputation, marketing efforts. Software quality factors are also important; games have to perform really well; being able to deal with large volumes of complex data and graphics in real time.

Usability is important for games. Games are a popular form of entertainment as they add an elusive dimension to entertainment e.g. *interaction*, as opposed to music or movies. Providing interaction can be a unique selling point over other forms of entertainment, however providing it in the wrong way, e.g. different from how the user expects it, is usually detrimental to the sales of a game. Usability is important for games: if a player:

- Cannot figure out how to play the game.
- Gets killed ten times in a row because the game is too hard or difficult to control.
- Has to replay part of the game over and over again.

It will all break the commercial success of a game.

Game design (including interaction design) is still commonly implemented by applying esoteric heuristics [3] gathered over many years of professional experience and few game developers consider the target audience during the design process [4]. Interaction design for games is hard and often relies on the experience of senior game designers. Because the game business is risky and a game company is only as successful as its last release; the importance of short-term goals is often exaggerated; leading to skipping proper usability testing under the pressure of deadlines and shipping products, which is higher in the games industry than in any other IT sector.

Designers need effective usable tools which are based upon proven knowledge of design. By capturing what is considered the "art" of good game interaction design, and sharing this knowledge between designers, this process can be made more accessible to inexperienced designers; hence minimizing the risk of building a game with usability problems. Currently there are some efforts to capture such knowledge as guidelines or heuristics. As identified by Welie [5] guidelines have several shortcomings with

regard to selection, validity and applicability (see our discussion in the next section). Interaction design patterns [6,7] have proven to be more powerful than guidelines as tools for designers. We use the term interaction design patterns and usability patterns interchangeably throughout this paper as they are the same. Existing pattern collections tend to focus on web and user interfaces (UI) for general purpose software, which makes them hard to use as tools for game design. Because of the uniqueness of the games domain (and resulting game specific usability problems) we decided to develop a set of usability patterns which describe solutions for usability problems, players have when playing games.

Usability and gameplay are often confused but we consider them to be two different concepts which are strongly related. Gameplay is a much broader concept and really focuses on providing interaction in such a way that it is fun. For example, many consider it to be fun to move a paddle to reflect the ball in Pong. Usability however, deals with providing such interaction in the *right* way e.g. how the user would expect it e.g. moving a paddle with the mouse or arrow keys rather than with some strange key combination. Aesthetics forms an additional dimension by focusing on the visual representation of this interaction. Although current game heuristics tend to mix usability requirements with gameplay requirements, for separation of concerns, it would be best to separate them. In this paper we strictly focus on usability.

The remainder of this paper is organized as follows. In the next section, we discuss background and related work. Section 3 discusses the problems with guidelines. Section 4 presents our usability patterns for games. Section 5 discusses some of the issues we encountered during the definition of our pattern collection. Section 6 presents future work, and the paper is concluded in section 7.

## 2. Related Work

In the area of usability and games, prior to the year 2000 very little research has been performed. In 1980, Malone [8] developed a set of heuristics tailored to instructional

games, but research in this area has been quiet until it was picked up again recently. Several papers on games and usability have been published which take a process oriented approach [9] such as Peterson who [10] argues why documentation should be an integral part of video game construction in order to guarantee a satisfying user experience. Lucas [11] discusses the methods used for the usability testing of Ion Storms game *Thief*. Laitinen [12] describes two usability testing methods for games. Ryan [13] discusses some problems players have with game access which is related to usability.

Related to our product oriented approach [9] to capture relevant design knowledge is the definition of a set of game heuristics in chapter 1 of [14] which specifies "what users want". Federoff [15] has looked at how existing usability heuristics such as proposed by [16] apply to games and a new set of game heuristics is proposed. [17] proposes using patterns for game design. Houser & Deloach [18] present seven principles for effective game design. Focusing on the broader concept of gameplay are the 400 project [3] of Falstein which collects proven game design rules and Bjork's game design patterns [19] which does the same but in the form of patterns. Bjork uses patterns for readability but does not present its work in a problem-context-solution dichotomy so they are actually heuristics rather than software design patterns.

## 3. Capturing design experience

Traditionally best practices concerning interface design have been captured by means of guidelines or heuristics such as Nielsen's heuristics [16]. The purpose of guidelines is to capture design knowledge into concise small rules which can be used to inform interface and interaction design. With regard to game design, similar approaches have been taken (see related work). Game heuristics exhibit several problems [20,21]. Guidelines tend to become quite numerous, Falsteins 400 project [3] aims to capture 400 rules of gameplay. Federoffs [15] list consists of 42 heuristics. Because they are numerous, it's often hard to select the right rules that apply. In addition, guidelines often suggest a

general absolute validity but in fact they can often only be applied in a specific context [5]. For example, Federoff [15] specifies "*Do not expect the user to read a manual*" but this does not apply to online flash games that do not come with a manual. It is also often unclear what the problem is the guideline actually tries to solve and why. Federoff specifies "*Players should be able to save games in different states*" but this guideline does not explain why, and a game designer may be hesitant to take such a guideline for granted. All these problems make guidelines difficult to apply to a particular design problem.

Patterns are a richer format for describing design experience and can overcome some of the limitations that guidelines have. Patterns provide a context in which to use the solution and tell the designer *when, how* and *why* the solution can be applied [5]. Patterns are based upon proven solutions. Patterns and guidelines do not exclude each other; patterns just describe one or more rules in a very specific situation (depth) whereas guidelines are more high level (breadth) and are more suitable as requirements. Interaction design patterns have been successful and several books have been published about web and user interface (UI) design patterns [22,23,24,25]. Several pattern collections [26,27,28,29] are freely accessible on the web.

Like any other software domain, games present some unique usability problems. Some usability problems that we found are:

- The player has to watch a cut scene over and over again.
- The player gets killed or injured repeatedly.
- A player needs to control an object which is difficult to manipulate with a controller.
- The player forgets to make a save game and has to replay part of the game again.

These problems are typically contextual; a usability problem where the player has to watch a cut scene over and over again only occurs in games that have cut scenes.

Existing pattern collections tend to focus on problems in web and user interfaces (UI) for general purpose software, which makes them hard to use as tools for game design. Games

are different from traditional software systems in the sense that most traditional software systems are developed with the purpose to support a user in performing a particular task; such as writing a letter or ordering a book on a website. Games are developed for entertainment or educational purposes. The most important aspect of a game is fun; a game that is not fun to play is doomed to fail in a highly competitive market. Applying existing interaction design patterns to the domain of games is difficult as the focus of these collections is on performing particular tasks then on playing a game. Although some patterns such as a wizard [28] can be found in games (e.g. an installation wizard), most other patterns such as a shopping card or undo are typically not applicable to games.

Motivated by the success of domain specific pattern collections such as Yahoo [30] and the observation that in certain games solutions to these particular usability problems can be found, we decided to capture this design knowledge by developing an interaction design pattern collection for the games domain.

#### 4. Usability Patterns in games

Patterns originated as an architectural concept by Christopher Alexander [31]. Each pattern is a three-part rule, which expresses a relation between a certain context, a problem, and a solution. Our patterns are described using the following format:

- **Problem:** problems are related to playing the game.
- **Context:** the context extends the plain problem-solutions dichotomy by describing specific situations or types of games in which the problems may occur.
- **Forces:** within the context, a number of forces act, which need to be resolved.
- **Solution:** a proven solution to the problem which resolves the forces.
- **Principle:** the solution is usually based upon a 'principle' Nielsen's heuristics [32] have been chosen as these tend to be the most widely used and recognized. An overview of Nielsen's heuristics is provided in the first column of Table 1.

- **Why:** provides a reasonable argumentation for the specified impact on aspects of usability when the pattern is applied.

- **Examples:** Examples of games in which the pattern has been successfully implemented.

Our patterns are part of an ongoing effort to describe successful solutions that benefit the players of the game. Consequently, they are solutions some of us are acquainted with. The main purpose defining our collection is to identify the “building blocks” of usable interface and interaction components of games in order to aid design during an early stage.

With our collection we aim to inform UI designers as well as software engineers. Certain patterns in our collection, such as seamless Gameworld (see Table 4), have proven to be hard to *retrofit* during late stage development which may lead to prohibitively expensive implementation costs. By discussing, during requirements analysis, how this pattern could improve usability and what is required from the system, the mutual awareness of the restrictions that exist between software engineering and usability can be raised, which may prevent part of the high costs incurred by maintenance activities once the system has been implemented [33].

<b>Nielsen’s heuristics</b>	<b>Pattern name</b>
Visibility of system status (4)	Game Progress, Instant Replay, Closed Captioning, Visual Saves.
User control and freedom (3)	Free look, Pause, Skip Cut scene.
Error prevention (5)	Auto Save, Control Assistance, Playground, Slow, Rewind.
Recognition rather than recall (1)	Journal.
Flexibility and efficiency of use (8)	Adaptive Difficulty level, Arcade Mode, Customizable controls, Fast Forward, Quick Save/Load, Quick Restart, Quick Start, Seamless World.
Help and documentation. (1)	Tutorial agent.
<b>Uncategorized (1)</b>	Pre Loader Game.
<b>Not used:</b> Aesthetic and minimalist design (0) / Consistency and standards (0) / Match between system and the real world (0) / Help users recognize, diagnose, and recover from errors (0).	

Our patterns were “harvested” from existing games and are hence based upon proven solutions. No specific methodology was used other than selecting a set of candidate patterns and putting these online.

These patterns and their relevance have been discussed with the game development community on game development related sites and newsgroups such as gamedev.net and the IGDA forums. Our patterns were presented at a workshop on games and usability at CHI2006 [34]. Through these activities we received feedback and could iteratively improve and add patterns to our collection.

Table 1 lists the 23 patterns that we have currently identified. The patterns have been organized according to Nielsen’s heuristics, to create an entry point to our collection based upon usability principles. E.g. if a designer wants to fulfill a heuristic as a requirement (such as provide error prevention) easily patterns can be found that may full such a requirement. For each heuristic we stated the number of patterns that address it.

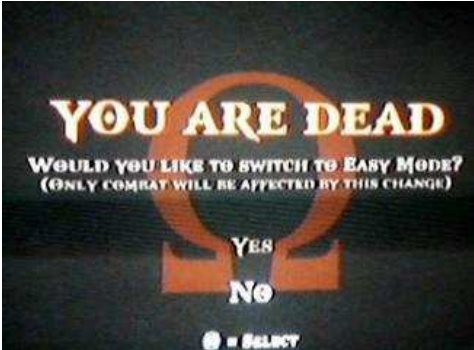
When categorizing the patterns, we identified that they map relatively well on Nielsen’s heuristics, though some categories (see bottom row of table 1) are not addressed at all. One pattern (pre loader game: playing a simple game while waiting for the game to load) could not be assigned to a heuristic. Some patterns like Rewind (which is a sort of undo for games) could map onto multiple heuristics (like user control and freedom). In a study on how existing usability heuristics apply to games Federoff [15] states that Nielsen’s heuristics appear to be helpful when analyzing the interface of the game but fail in the ability to address gameplay issues. With out pattern collection we have concentrated on usability problems in the interface and the interaction which is in accordance with that finding. Only one pattern (Pre loader game) was found that is more related to gameplay.

In Table 2 we provide an alternative organization of our pattern bases on game mechanics in general. We use three categories; modes of play, player actions (which are subdivided in generic and time manipulation related actions) and game elements.

Game Mechanic		Pattern name
Modes of Play (4)		Control Assistance, Playground, Adaptive Difficulty level, Arcade Mode.
Player Actions	Generic (7)	Free look, Quick Save/Load, Quick Restart, Quick Start, Auto Save, Seamless World, Instant Replay.
	Time manipulation (5)	Skip Cut scene, Rewind, Pause, Slow, Fast Forward.
Game Elements (7)		Tutorial Agent, Journal, Game Progress, Customizable controls, Subtitles, Pre Loader Game, Visual Saves.

Creating an alternative entry point in our collection may give our collection more focus for game designers and may allow identifying new patterns.

Because of page limitation only three patterns are included in this paper as examples. But the reader of his paper is encouraged to visit the rest of our pattern collection on our website (<http://usability.eelke.com>)

	
<p><b>Figure 1: adaptive difficulty level in God of War</b></p>	
<b>Problem</b>	The game is too hard to play on the current difficulty setting e.g. player gets killed/injured repeatedly.
<b>Context</b>	Games which offer different difficulty levels to cater for different types of players. The most common terms for levels of difficulty are easy, normal and hard. Depending on the difficulty level: <ul style="list-style-type: none"> <li>• Enemies are stronger or weaker.</li> <li>• Puzzles are harder or easier to solve.</li> <li>• More or less guidance is provided.</li> <li>• More or less control assistance is provided</li> </ul>

	(e.g. auto aim or auto steering).
	Usually the player chooses a difficulty level when starting the game, and usually it is not possible to switch to a different difficulty level halfway without starting over.
<b>Forces</b>	<ul style="list-style-type: none"> <li>• Players want the game to be challenging yet forgiving. E.g. it should be challenging to play a game but players do not want to have to try getting past a point in the game over and over again.</li> <li>• Some players may have experience playing similar games. Others have little experience.</li> <li>• Some players are better or worse than others at certain game aspects such as shooting or puzzle solving.</li> <li>• The game's difficulty level with regard to a certain aspect may vary during the game e.g. one part of the game may focus more on shooting and another part of the game on solving puzzles.</li> </ul>
<b>Solution</b>	<p><b>Adjust the difficulty level to the player</b></p> <p>The game should adapt to different players during different parts of the game. There are two options for implementing this pattern:</p> <ul style="list-style-type: none"> <li>• Suggest a different difficulty level according to the player's performance. E.g. the game could suggest an easier difficulty setting after the player has failed/been killed a number of times to pass a point in the game.</li> <li>• Automatically adjust the difficulty level based on the player's performance. This removes the "traditional difficulty levels". Every time the player dies or fails, there is a chance that the player will switch to the next easiest difficulty setting. If playing well for a while there is a chance the player will go up a difficulty level.</li> </ul>
<b>Principle</b>	Flexibility and efficiency of use.
<b>Why</b>	Adjusting the difficulty level to the player may increase satisfaction and efficiency and players will not become frustrated.
<b>examples</b>	God of War, Resident Evil 4.

**Table 4: Seamless Gameworld**



**Figure 2: seamless game world in Dungeon Siege**

<b>Problem</b>	Players need to wait before entering a (new) part of the game.
<b>Context</b>	Typical to "free roaming" games e.g. 3rd person shooters, role-playing, action or simulation games with the ability to move around freely in a large world. These games usually have a nonlinear game plot (if any). Usually such a world is partitioned into zones as such a world cannot be loaded in whole in the memory. When a player moves from one zone to another (e.g. goes into a house) the player usually has to wait for the new zone to be loaded into the memory.
<b>Forces</b>	<ul style="list-style-type: none"><li>• Players are impatient and do not want to wait.</li><li>• Players do not want their game to be interrupted.</li></ul>
<b>Solution</b>	<b>Provide a seamless world.</b> Instead of letting the player wait before entering a new zone, pre-load the level before the player enters the zone. Rendering a huge seamless world may have a significant effect on the game engine design; an example implementation could be as follows: The entire world is broken up into chunks (chunk size depending on available memory). <ul style="list-style-type: none"><li>• Only 9 chunks are loaded into memory at any given time, the chunk the player is currently in and the 8 surrounding chunks.</li><li>• As the player moves out of the central chunk into one of the bordering chunks, the 3 chunks farthest from the player are discarded and the chunk the player just entered then becomes the center chunk.</li></ul> The chunks can either be loaded in a separate thread or in the same as long as the player does not have to wait to load a chunk.
<b>Principle</b>	Flexibility and efficiency of use.
<b>Why</b>	Having a large, persistent world adds a level of constant immersion for the player, as the game never stops and never loads.

This solution increases efficiency and satisfaction.

**example** Dungeon Siege, World of Warcraft

**Table 5: Slow**



**Figure 3: Slow in Max Payne**

<b>Problem</b>	The player needs to successfully perform a series of actions in a short period of time.
<b>Context</b>	Common to action games such as first person shooters or platform games. Achieving a certain goal (such as finishing a level, going through a door) depends on successfully performing a series of actions often within a constrained period of time. For example, the player may push a button to open a door. The door closes in a certain amount of time. In order to go through the door, the player may need to jump over a pit, defeat an enemy etc. For advanced players such a challenge may not be an obstacle but (novice) players may find it very hard to accomplish and the player often has to try several times before the player succeeds.
<b>Forces</b>	<ul style="list-style-type: none"><li>• Time manipulation cannot be implemented in multiplayer games.</li><li>• Players don't want to have to play part of the game over and over again when they die.</li><li>• Making the game too easy ruins gameplay.</li></ul>
<b>Solution</b>	<b>Allow the player to slow down the time</b> Throwing the world into slow motion while moving around in real-time gives several advantages. <ul style="list-style-type: none"><li>• Faster movement - run/jump/dodge/avoid/aim/move faster giving unique advantages over enemies and obstacles. This is especially useful when trying to achieve time related goals.</li><li>• Increased damage - when fighting enemies one can do more damage as one can deal more blows/hits/kicks and</li></ul>

the enemy has a harder time blocking the attacks.

Care must be taken that the world can only be slowed for a brief period of time as slowing it all the time can make the game too easy. In order to achieve this one can consider letting the player sacrifice something in order to activate slow. At any given time there is only a limited amount of activation tokens available. This makes sure the player only uses slow sparsely and slow does not ruin gameplay.

**Principle** Error prevention.

**Why** Slowing down the game makes the game easier to play as the player has more time to respond and perform specific actions, increasing satisfaction and performance.

**examples** Max Payne, Prince of Persia: sands of time, Blinx: the time sweeper.

## 5. Discussion

The question remains how detailed or elaborated usability patterns for games need to be. Experienced game designers may find most patterns trivial and dismiss them. For beginners they might prove useful. The issue of detailedness of a pattern is a general 'problem' in design pattern research. Nonetheless, we feel that the level of detailedness we show in our patterns allows the patterns to be useful for both UI designer and software engineers while not being too detailed. The current implementation description allows a software engineer to assess the impact of this pattern on the existing game architecture.

Sometimes it's hard to describe a pattern without having to make some basic assumptions about an underlying mechanism. When describing our auto save pattern we found it very difficult to describe it without actually specifying something about "saving" in general. In our pattern description we assume players can save games but why is this? Is saving a usability feature or is this pure functionality? In the case of word processors one cannot imagine a system without the ability to save but for certain games this very well possible (such as beat em up games).

When we started this collection we initially focused on identifying patterns that are "unique" to the games domain. This is very hard as certain patterns that we identified such as autosave, and visual saves are

occasionally also found in desktop software. Autosave surely has its roots in the games domain. The preview [7] pattern is related to our visual saves pattern. It can be argued that visual saves is a domain specific implementation of this generic pattern. For the completeness of our collection we decided to include this pattern even if it is a child of a more generic pattern. It is not our purpose to translate generic patterns to game specific patterns as patterns should be defined as domain independent as possible.

## 6. Future work

For empirical validation of our collection we should analyze the usability of a game with and without a particular pattern. Such an experimental setting is not easily found, especially taking into account the cost of developing games. At the moment we are performing a case study where we analyze the presence of our patterns in commercially successful games, to verify whether there is a relationship between commercial success and a number of patterns implemented assuming games with usability problems are not commercially successful.

When the number of patterns for a particular domain grows on which everybody agrees and patterns are related to each other a pattern language for a domain emerges. We currently focused on the development of patterns, the next step, if a sufficient number of patterns have been developed would be to create a network of patterns that are related. Our alternative organization (see Table 2) could already be used to relate patterns that use the same game mechanic such as time control.

An issue that has been facing the game industry recently is the need to provide more accessible games as many games are not accessible to disabled gamers. Usability patterns can be useful in this area too as they describe a better way of interacting a player and the game. Accessibility problems are to some extent usability problems for a special category of gamers i.e. disabled. Patterns could be useful for describing accessibility requirements and design solutions. A pattern such as control assistance or slow could be useful to make games accessible for physically disabled users.

## 7. Conclusions

Designing usable games is hard and often relies upon years of experience. Existing heuristics & guidelines tend to be quite numerous and prescriptive, e.g. do this or don't do this. Usability patterns can capture design knowledge in a much richer format; patterns tell a designer exactly when, how and why the solution can be applied. Existing interaction design pattern collections focus on web and user interfaces for general purpose software, which makes them hard to use for game design. We have developed a collection of usability patterns which describe solutions to typical usability problems in games. With our patterns we only focus on the "do this" and are hence prescriptive and constructive; providing more usable design tools for game designers. Implementation details are provided which allow a software engineer to assess the implementation effort for a given system. Our patterns have been harvested from existing games. The format and scope of our patterns can contribute to the development of a larger number of patterns than that we have now. Such a body of knowledge may then effectively be used to inform UI design for games; it may aid communication during requirements design and may lead to games with fewer usability problems.

## References

- [1] Austin Grossman, *Postmortems from Game Developer*, CMPBooks, San Francisco, 2003.
- [2] DTI, *From exuberant youth to sustainable maturity: competitive analysis of the UK games software sector*, [http://www.dti.gov.uk/industries/computer\\_games/computergames\\_appendices.pdf](http://www.dti.gov.uk/industries/computer_games/computergames_appendices.pdf), 2002.
- [3] N. Falstein, *The 400 Project*, [http://www.theinspiracy.com/400\\_project.htm](http://www.theinspiracy.com/400_project.htm)
- [4] J. Sykes and A. Patterson, *Considering the user in video game design*, *Proceedings of the Twelfth European Conference on Cognitive Ergonomics (ECCE)*, 2004.
- [5] M. Welie, G. C. van der Veer, and A. Eliëns, *Patterns as tools for user interface design.*, *International workshop on Tools for Working with Guidelines*, 2000.
- [6] J. Tidwell, *Interaction Design Patterns*, *Conference on Pattern Languages of Programming 1998*, 1998.
- [7] M. Welie and H. Trætteberg, *Interaction Patterns in User Interfaces*, *7th Conference on Pattern Languages of Programming (PloP)*, 2000.
- [8] T. W. Malone, *What makes things fun to learn? a study of intrinsically motivating computer games - phd dissertation*, department of psychology, stanford university, 1980.
- [9] T. Keinonen, *One dimensional usability - influence of usability on consumers product preference*, University of Art and Design, Helsinki, 1998.
- [10] M. Peterson, *Why Game Documentation is essential to a satisfying user experience*, <http://www.stcsig.org/usability/newsletter/0410-gamedocs.html>
- [11] S. Lucas and D. Fulton, *What We Learned Evaluating the Usability of a Game*, <http://www.stcsig.org/usability/newsletter/0410-gameevaluation.html>
- [12] S. Laitinen, *Better Games Through Usability Evaluation and Testing*, [http://www.gamasutra.com/features/20050623/laitinen\\_01.shtml](http://www.gamasutra.com/features/20050623/laitinen_01.shtml)
- [13] W. Ryan, *Barriers to Entry: Designing Player Access in Video Games*, *Workshop: On the process of game design*, 2006.
- [14] R. Rouse, *Game design in theory and practice*, Wordware Publishing Inc, Texas, Usa, 2000.
- [15] M. A. Federoff, *Heuristics and Usability Guidelines for the Creation and Evaluation of Fun in Video Games*, Indiana University, Indiana University, 2002.
- [16] J. Nielsen, *Usability Engineering*, Academic Press, Inc, Boston, MA., 1993.
- [17] B. Kreimeier, *The Case For Game Design Patterns*, [http://www.gamasutra.com/features/20020313/kreimeier\\_pfv.htm](http://www.gamasutra.com/features/20020313/kreimeier_pfv.htm)
- [18] R. Houser and S. Deloach, *Learning from Games; Seven principles of effective design*, *Journal of Technical Communication, STC*, 1998, pp. 319-329.
- [19] S. Bjork, S. Lundgren, and J. Holopainen, *Game Design Patterns, Level-Up: Digital Games Research Conference*, 2003.
- [20] A. Dix, G. Abowd, R. Beale, and J. Finlay, *Human-Computer Interaction*, Prentice Hall Europe, 1998.
- [21] M. Mahemof and L. J. Johnston, *Principles for a Usability-Oriented Pattern Language.*, *Proceedings of Australian Computer Human Interaction Conference OZCHI'98*, 1998.
- [22] J. Borchers, *A Pattern Approach to Interaction Design*, John Wiley & Sons, 16-5-2001.
- [23] I. Graham, *A Pattern Language for Web Usability*, Pearson Education, 31-1-2003.
- [24] D. K. van Duyne, J. A. Landay, and J. I. Hong, *The Design of Sites: Patterns, Principles, and Processes for Crafting a Customer-Centered Web Experience*, Addison-Wesley, Boston, 2002.

- [25] J. Tidwell, *Designing Interfaces : Patterns for Effective Interaction Design*, O'Reilly Media, Inc., 21-11-2005.
- [26] UK. The Usability Group at the University of Brighton, The Brighton Usability Pattern Collection. <http://www.cmis.brighton.ac.uk/research/patterns/home.html>
- [27] J. Tidwell, *Common ground: Pattern Language for Human-Computer Interface Design*, [http://www.mit.edu/~jtidwell/interaction\\_patterns.html](http://www.mit.edu/~jtidwell/interaction_patterns.html)
- [28] M. Welie, *GUI Design patterns*, <http://www.welie.com/>
- [29] Lancaster University, *PoInter: Patterns of INTERaction collection*, <http://www.comp.lancs.ac.uk/computing/research/cseg/projects/pointer/patterns.html>
- [30] Yahoo, *Yahoo design pattern collection*, <http://developer.yahoo.com/ypatterns/>
- [31] C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, and S. Angel, *A Pattern Language*, Oxford University Press, 1977.
- [32] J. Nielsen, *Heuristic Evaluation.*, in *Usability Inspection Methods.*, Nielsen, J. and Mack, R. L., John Wiley and Sons, New York, NY., 1994.
- [33] E. Folmer, M. Welie, and J. Bosch, *Bridging Patterns- an approach to bridge gaps between SE and HCI*, *Journal of information & software technology*, Kluwer, 2006, pp. 69-89.
- [34] E. Folmer, *Interaction Design Patterns for Games*, *Computer Human Interaction Conference 2006 - workshop on HCI and Games*, 2006.