Gamification in the Development of Accessible Software

Andreas Stiegler and Gottfried Zimmermann

Responsive User Interface Experience Research Group, Stuttgart Media University, Stuttgart, Germany
mail@andreasstiegler.com, gzimmermann@acm.org

Abstract. This paper describes a theoretical framework covering game design, game mechanics and game engines, linking examples from actual commercial games with a gamification application. The goal of the framework is to develop an online platform for software developers to aid them in designing accessible applications, finding help on the topic etc. A software stack will be derived taking a typical game development project as an example. We will further identify process requirements for implementing crucial game design rules, like immediate feedback. Finally, an outlook on the final project will be given and possible evaluation metrics will be described.

Keywords: Gamification, Game Development, Serious Games.

1 Introduction

Gamification has become a popular term in management, application design and usability in the last few years. Gamification refers to the application of methodologies and design patterns from game design to other contexts [3], such as the design of a business application. The idea behind this approach is to bring a possibly repetitive, non-rewarding work-task closer to tasks as performed in games, which are usually considered fun and entertaining. One has to be careful, though, on how to apply gamification, since the question how to apply it to its best effect is neither trivial nor straightforward. Game development is often split into three important fields: game design, dealing with the meta-level of a game, such as its topic, general approach, setting or features. Game mechanics are, roughly speaking, the rule sets that implement the game design into an actual game, which includes describing how characters move, what attributes an object has and what the action space of a game is. Finally, the actual game engine is the implementation of the game mechanics into executable code, including a renderer, a physics engine, artificial intelligence and more. Each of these three layers comes with its own requirements and restrictions and offers new challenges. Gamification is often described as just using metaphors and approaches from game design, which never happens in actual game development, as game design, game mechanics and the game engine are closely tied together. An effective approach of gamification will therefore have to take all of them into consideration and adapt them to a business scenario.

C. Stephanidis and M. Antona (Eds.): UAHCI/HCII 2014, Part I, LNCS 8513, pp. 171–180, 2014.

2 Gamification

The term gamification is used in various contexts with different definitions. [3] defines gamification as "the use of game design elements in non-game contexts". Other definitions classify certain design approaches as gamification, leading to games themselves being gamified if they include systems like achievements or badges [5]. While the definition of [3] is more generic, it also covers the evolution of gameplay and game mechanics best. Typical cooperative or competitive reward systems like avatar upgrades in the form of player levels were used in the very first games [11] and are commonly adopted in gamification, while other reward systems, like badges and medals, were used in internet forum software a lot more frequently, before they became mainstream in computer games. Both game design and game mechanics are evolving and changing over time and there is also a flow back from non-game applications into games. Recent "blockbuster" titles like Battlefield 4, for example, use a web page with a browser plugin as their game menu, only launching the actual game application when a player picks a game server to participate. Thereby most of the avatar development like configuring weapons and checking unlocked rewards are moved into the browser. Typical non-game design patterns are used in this context and the game closely ties into social media. There is also a tablet application for Battlefield 4 that uses typical design patterns for the mobile platform. This is a good example of adopting non-game design patterns in games, beyond the very basics of user interactions.

Gamification became a popular research topic in human-computer interaction, starting in the 1980s with work by [8]. Today, computer games are becoming a social phenomenon. Massively multiplayer online role-playing games (MMORPG) like World of Warcraft® are attracting player bases of several millions and eSport is becoming more and more popular. Research in gamification aims to identify the metaphors and methods used to make games successful and apply them in a work or educational context. In general, gamification not only applies to computer software, but also to physical real-world examples [7], although this paper will focus on digital game development and derives patterns from these applications.

We will follow the gamification definition of [3]. Gamification is not a single method that can be applied to a specific problem, unlike, for example, a software design pattern that can be applied when certain problem criteria are met and that offers a path to solve the problem. Instead, gamification describes the repurposing of many different "game design elements", which themselves vary greatly from genre to genre and even from title to title [4].

2.1 User Interface

The most obvious "game design elements" taken from games are interface design patterns [1]. A typical game interface looks different from the average software application at first glance. While the basic interface elements are the same, particularly in more interface-intensive game genres like massively multiplayer online role-playing games or real-time strategy games, their visual design differs greatly. Figure 1 shows

the most important part of the Diablo® III user interface, the action bar, where players can find the most vital information about their character: the current experience gathered to advance to the next character level is displayed as a conventional progress bar (small orange bar, center). The character's health (red globe, left) and resource to use abilities (purple globe, right) are both represented as slightly different (non-linear) progress bars, as the fill level of the globe represents the current status of the respective resource. Due to the different volume distribution of spheres along an axis, a 90 percent fill state of the health sphere does not represent 90 percent of the maximum health points. This suits gameplay well, as players see larger visual drops in their fill level when their character has only few health points left and is close to death. The ability buttons (bottom center) are an array of buttons that grant access to selected abilities. When an ability is used, the button also displays the cool down until the spell is ready for use again, as well as the hotkey binding. Small conventional buttons to the right grant access to other important parts of the user interface, such as the character sheet or the inventory screen.

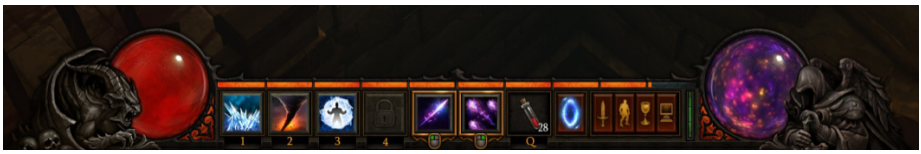

Fig. 1. Diablo III Action Bar

The user interface also includes purely aesthetic elements like the demon and angel figure (very left and right) which serve no game play purpose, but still are quite prominent and about the size of the most vital health-globe and even overlap them. Game user interfaces also follow the "five E's" principle of interface design [10]: Efficient, Effective, Engaging, Error tolerant, and Easy to learn. Yet, comparing a typical office application to a game shows that their weights and goals for these categories differ.

The "Engaging" criterion is probably the most obvious one. In a game, this category also includes weaving the interface into the setting and atmosphere of the game. This aspect is very crucial for games, as the user interface is often the primary interaction channel for a player: if a game's goal is to achieve immersion, the very interaction channel has to transport the setting and atmosphere of the game, just as much as the 3D graphics on the rest of the screen. This results in a large portion of the interface being concerned with reassembling symbols of the game setting, like demons and angels in the case of Diablo III.

The "Effective" criterion is another user interface aspect in games that often differs in weight and goals. Obviously, the user interface has to be effective to fulfill the goals assigned to it. Yet, in a game, these goals themselves are restricted. Consider the example of Diablo III again, compared to a typical office application like Microsoft Excel®. Introducing a "Solve Problem" button to Excel that automatically solves any problem encountered in a spreadsheet (for simplicity, let us assume there is a very advanced artificial intelligence running in the background that can solve any problem you might encounter) would actually be a great benefit to the application and

a superior selling point on advertisements. In a game like Diablo III, a “Solve Problem” button replacing the action bar would ruin gameplay, as combining abilities and using them in correct order is one of the basic principles of the game. This difference originates in a popular concept known as flow [2]. Flow describes matching the challenge of a game to the skill level of the player. If the game challenges are too low for the skills of the player, he may become bored. On the other hand, if the challenges are too hard, the player may become frustrated. Including a deliberate challenge, even if there would be automated solutions, is a very important aspect of games [6].

Another aspect of user interface design that game developers are particularly concerned in is symbolism. Unlike most applications, many games have quite complex virtual 3D worlds, where navigation and user interaction are more difficult to handle than on a flat 2D plane. Though most gamified applications will not require offering a 3D world for interaction, there are certain principles used in 3D interaction design that can be transferred to 2D environments. Symbolism, i.e. using the same iconic object for any instance of a respective interaction possibility, is already well established in application design. A game example from Diablo III would be the reward chests hidden in dungeons sharing the same visual appearance, no matter in which dungeon the player is.

Another method, which is less frequently used in gamified applications, is color coding. A typical example can be seen in Figure 2, showing a game scene from Mirror’s Edge. The game features a vast, light-colored environment, where players have to navigate through a city using procuring techniques. Certain interactive game elements are colored in bright red. Color coding can be a very effective method to achieve navigation, but comes with obvious accessibility constrains. One could imagine pattern coding or similar approaches to achieve the same effect without relying on color perception.

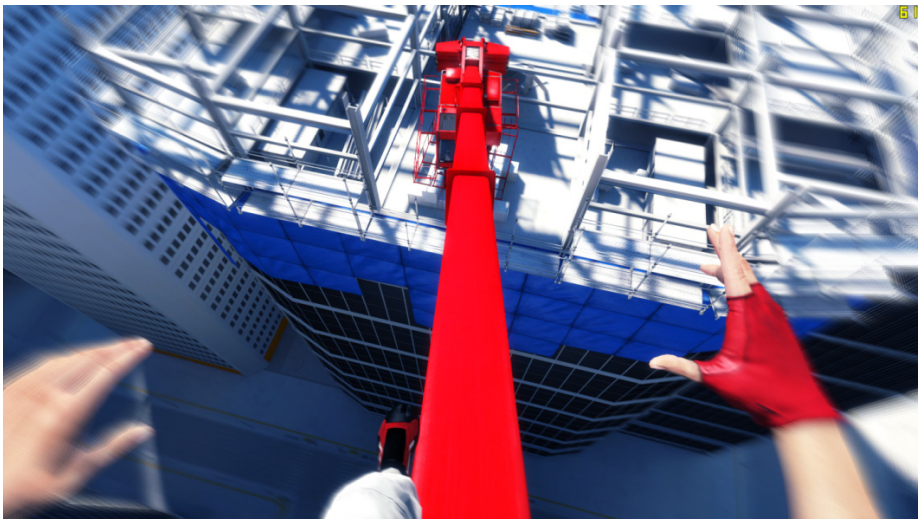

Fig. 2. Mirror's Edge Color-Coding

A gamified application might be limited in purpose and screen estate, and may lack opportunities for adding aesthetic elements to the user interface. Yet, in the context of a certain gamification metaphor – e.g., displaying a road map as an actual map that developers uncover while fulfilling milestones – it can help to underline this metaphor by adding additional visual elements, e.g. wrinkled map edges.

An often-neglected aspect of game user interfaces is that they should be usable and accessible for as many users as possible, including those with disabilities. Game accessibility guidelines, e.g. [9], and software frameworks, e.g. [12], can support the game designer in achieving this goal. It is obvious that some games cannot be made accessible to all users without changing core features. For example, a person who is blind will not be able to play a game in which flying a fighter jet in real-time is a requirement. However, with some creativity, it may be possible to deliberately change the game to include alternative (and more inclusive) playing modes beyond the "traditional path". For example, an "auto pilot" function could be added to the game, and the challenge for the user would be to continuously adjust the parameters of the auto pilot in such a way that the fighter jet performs best in the current situation.

The inclusive approach in designing for gamification is important since we want all users to benefit from the gamification features, not just a subgroup. Games sometimes deliberately address a specific user group only, e.g. young persons with good vision and quick reaction; or older persons who need to jog their brains to prevent dementia to progress. However, if we apply gamification to an application that is used by a broad variety of users, we need to make sure that it is for the benefit of all these users.

2.2 Game Play

In game development, game play is often compared to the script of a movie. Game play describes the high-level design aspects of the game, like the genre, the story, the environment, and setting, what important goals a player should pursue, what aspects of the game should be the most relevant for a player, how social interaction with other player works etc. Notably, game play does not define the actual rules that implement these goals. This is a major difference to application design, where the rules of the underlying business logic often dictate closely what the application should look like.

Challenge, is a very basic principle of game design. In order to design for flow, the game play often specifies what kind of challenges a player can expect and how they scale along a players experience and increasing skill. A gamified application has to offer some kind of challenge to its users, as most established gamification mechanisms, such as reward systems or competitive comparison become meaningless if there is no challenge involved. This requirement might actually contradict the efficiency of the application, which is often very crucial for business applications. Yet, in the context of an educational or communicational application, challenge might actually benefit the goal of the application and should be planned carefully from the very beginning

of development. It might be worth reducing efficiency for offering a meaningful challenge to the user, as the overall contribution of a user having fun with an application is increased.

Another important aspect of game play is replayability, i.e. describing methods that make a replay of the game entertaining for the player. This might involve challenge considerations, as a challenge already mastered is usually a lot easier than an unseen one, as in puzzles.

Game play development is very specific not just to a particular genre but also to a particular game. Different real-time strategy games, for example, differ greatly in their game play. Therefore, metaphors from game play cannot be transferred in a straightforward way to a gamified application. Yet, including a "game play" development step in the design phase of a gamified application, planning for challenge or replayability (or repetitive tasks in a gamification context) is an important advice for designing a gamified application. Adaptable vs. adaptive system

We call a system *adaptable* if it lets the user adjust its user interface aspects by dedicated user actions, e.g. in a configuration dialog, or by activating a button or menu item. A system is *adaptive* if it automatically adapts some aspects to the current context of use; which may involve asking the user for confirmation of the suggested change. In brief words, a system is adaptable if the adaptation is user-initiated, and it is adaptive if the adaptation is system-initiated.

Oftentimes, systems are both adaptable and adaptive. This will typically result in mixed dialogs in which a user changes some aspect of adaptation, and the system will react upon this by proposing to change another (complementary) aspect to the user.

	Example: chess	**Example: StarCraft® II**
Game play	Two players/sets of pieces, turn-based game on limited piece of terrain, focus on tactical and strategic usage of figures. Goal: capture the enemy king, …	Three distinct alien races, real-time game on limited piece of terrain, two resources to gather to produce units, focus on tactical usage of units and economy, clear visuals to support eSport, …
Game mechanics	How the figures move, 8x8 squares, how to take a figure, rules for win/loose/draw, …	Damage and hitpoints of units, unit categories, techtree layout, economy and costs, …
Game engine	(the actual physical figures)	Component-based engine, deferred rendering, …
Game content	(the actual physical figures)	3D models for units, voice recording, cinematics, …

Fig. 3. Game examples: Chess and StarCraft II

2.3 Game Mechanics

The game mechanics of a game describe its set of rules implementing the goals and design concepts of the game play. Figure 3 shows a non-virtual example illustrating the difference between game play, game mechanics, and the game engine, although the latter is quite artificial in the context of a physical game. While the game play of a game specifies the more abstract rules, goals, and the setting of a game, the game mechanics implement it into actual rules, though these rules are still not implemented in code.

Two crucial aspects in game mechanics are difficulty and balancing. While those are rather generic terms, they are usually referred to balancing the action space of a player against either the environment and non-player characters (difficulty) or against other human players in a multiplayer game (balancing). For gamification, game mechanics mean to select appropriate tools for the user to achieve a certain goal, for example a badge system or avatar improvement. Yet, balancing or difficulty are often ignored in the process, though they are crucial for most games (there are examples of games where difficulty and balancing are hard to identify, for example Tetris® or many puzzle games in general).

Difficulty ties closely with flow, as it describes how challenging it is for a player to beat a certain portion of the game. Optimizing a game for flow would mean to linearly increase the difficulty of the game as the player's skills improve. This is a difficult if not impossible task, as it is neither possible to measure the skill of a player effectively, nor to measure how difficult a certain portion of the game is for a certain player. Many games therefore offer optional, explicitly challenging levels, which players can attempt to beat. Doing so increases the reward a player would usually get (like experience or a golden badge). Gamification could offer similar difficulty approaches, such as adding extra time to an imminent deadline, if a developer manages to reach an important milestone.

Balancing is always relevant when a game offers a multiplayer mode. It is of particular importance in eSport games, like StarCraft II, where a game should be perceived as "fair" even though the players may have differing skills. This can be achieved by offering the same chances of winning for all players, although their methods to reach victory might differ. In the context of accessibility and gamification, balancing becomes a crucial task. Accessible gamification cannot be limited to the user interface layer, which is trying to display the chosen interface elements in a way that they can be perceived, operated and understood by anybody. Gamified accessibility will also have to respect balancing, where the deployable methods per participant can vary greatly. This obviously influences game play decisions, as not any possible game play can be implemented with game mechanics that take this requirement into account.

2.4 Game Engine

The game engine, finally, is the technical implementation of the game mechanics in executable code. This is probably the clearest difference between serious games and

gamification, as serious games use the same underlying technology as a common game, while gamification usually builds on different software stacks, like the Web or an office application. In game development, there is an important feedback channel from the engine development to both game play and game mechanics, since no game mechanics – and therefore game play concepts – about what can be implemented. The game play goal of vast and continuous game worlds to explore will clash with the technical requirement of a rendering system that may require a maximum visible distance or a limited complexity of the game world.

Game content is an only vaguely defined term, used in game development to describe material that is used by the game engine to implement the game mechanics. While the game engine contains a rendering system that is capable of drawing a space ship on the screen, the actual model of the space ship would be the game content. Roughly speaking, everything that is supplied by an artist or not required for the engine to run without errors could be regarded as content. Yet, this definition becomes a bit unclear when thinking about scripting languages. These are often used in game engines to allow developers to work on difficulty and balancing without having to compile the whole game engine over and over again. Such scripts may define how the weapons and engines of the star ship work. These scripts are no longer technically part of the code of the game engine, as they are interpreted at run-time, but they are still required to get a meaningful game running. For the purpose of gamification, however, the game engine and game content can be construed as a single module.

Just as with game engines, the software stack of a gamification project varies greatly from project to project. There may be external constraints, like having to use a certain software framework or having to run in the browser. These constrains are not different from constraints that game engines have to face, for example having to get a certain rendering technique to run on consoles and PCs alike. A great difference between game development and a gamification project, however, is that game engines are usually the first thing that are specified and developed in a game project. This could either happen because parts of the game engine, like the rendering system or the physics simulation, are middleware, or because developing a game engine from scratch is both expensive and risky. This leads to game studies knowing very precisely what their technology is capable of when designing game play and game mechanics. In addition, if a certain game play or game mechanics concept seems worth it, the game engine can be extended to cover the new technological requirements. This is very different from many gamification projects that either start with requirements like "gamify this application" or just deliver design guidelines as their outcome.

In the first case, where the task is to gamify an existing application, the underlying software stack – the whole game engine and content – is set into stone, and there are only few successful game titles that build entirely on existing code. If gamifying an existing application is the goal, it is important to closely analyze the existing software and make changes to the code if necessary. Gamification is not an add-on that can be installed any time.

In the second case, gamification is used on a purely theoretical level and only delivers mock-ups or concepts on how to gamify a certain usage metaphor. This is similar to an experience that most enthusiastic young game developers have to go through,

i.e. when they first realize that their "splendid" game design and game mechanics they came up with are actually not implementable given the external constraints (usually runtime performance). Gamification cannot be done on a theoretical level and then be expected to run under the technological requirements.

2.5 From Game Development to Gamification Development

Above, we gave an overview over the three fundamental modules in game development: game design, game mechanics and the game engine. While all three of them have their unique aspects to cover, they also have clear and strong influences on each other. A successful game cannot be developed while ignoring one of these modules. Development in each of these modules can have impact on the development of the other two, requiring all three of them to be under continuous observation until the very end of the development phase – and even afterwards as maintenance (patching) is vital in today's gaming communities.

Gamification development shares many of the fundamental requirements of game development. The game engine may be very different, but both game mechanics and particularly game play are very similar. There is no reason – and no excuse – for not taking the challenge of developing all three of them continuously when creating a gamification solution. This is a significant challenge, but a challenge that the games industry has learned to master. It is neither easy nor are there any recipes granting success.

But at the end of the day, challenges reward golden badges.

3 Application and Outlook

Within the Prosperity4All project, we will create a gamified online platform for software developers to collaborate and learn about deploying accessibility in software. Care will be taken to make the platform suitable to use for all developers, including those with disabilities. The platform will link with popular developer sources like Stack Overflow and offer libraries, coding tools and tutorials on how to integrate into GPII and build software allowing integration of accessibility solutions. This platform will use gamification approaches derived from actual game development as described above in both its development phase and implementation. Evaluation of the described gamification approach is one of the key goals of the project.

Acknowledgments. The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2011) under grant agreement n° 610510, Prosperity4All ("Ecosystem infrastructure for smart and personalised inclusion and PROSPERITY for ALL stakeholders"). This publication reflects only the authors' views and the European Union is not liable for any use that may be made of the information contained herein.

References

1. Crumlish, C., Malone, E.: Designing social interfaces: Principles, patterns, and practices for improving the user experience. Yahoo Press (2009)
2. Csikszentmihalyi, M.: Flow and the Psychology of Discovery and Invention. Harper Perennial, New York (1997)
3. Deterding, S., Dixon, D., Khaled, R., Nacke, L.: From game design elements to gamefulness: defining gamification. In: Proceedings of the 15th International Academic Mind Trek Conference: Envisioning Future Media Environments, pp. 9–15. ACM (September 2011)
4. Elverdam, C., Aarseth, E.: Game Classification and Game Design Construction Through Critical Analysis. Games and Culture 2(1), 3–22 (2007)
5. Hamari, J., Eranti, V.: Framework for designing and evaluating game achievements. In: Proc. DiGRA 2011: Think Design Play, vol. 115, pp. 122–134 (2011)
6. Johnson, D., Wiles, J.: Effective affective user interface design in games. Ergonomics 46(13-14), 1332–1345 (2003)
7. Juul, J.: Half-real: Video games between real rules and fictional worlds. MIT Press (2011)
8. Malone, T.W.: Heuristics for designing enjoyable user interfaces: Lessons from computer games. In: Proceedings of the 1982 Conference on Human Factors in Computing Systems, pp. 63–68. ACM (1982)
9. Ossmann, R., Miesenberger, K.: Guidelines for the development of accessible computer games. In: Miesenberger, K., Klaus, J., Zagler, W.L., Karshmer, A.I. (eds.) ICCHP 2006. LNCS, vol. 4061, pp. 403–406. Springer, Heidelberg (2006)
10. Quesenbery, W.: The five dimensions of usability. Content and Complexity: Information Design in Technical Communication, 81–102 (2003)
11. Salen, K., Zimmerman, E.: Rules of play: Game design fundamentals. MIT Press (2004)
12. Vickers, S., Istance, H., Heron, M.J.: Accessible gaming for people with physical and cognitive disabilities: a framework for dynamic adaptation. In: CHI 2013 Extended Abstracts on Human Factors in Computing Systems, pp. 19–24. ACM (April 2013)