Kandidatarbete i Medieteknik, 30 hp

Vårtermin 2013

Dynamic difficulty adjustment for roleplaying games

Andreé Hallengren

Marcus Svensson

Handledare: Pirjo Elovaara & Fredrik Gullbrandson

Examinator: Peter Ekdahl

Blekinge Tekniska Högskola, Sektionen för planering och mediedesign

Abstract

This thesis explores the possibilities of employing a dynamic difficulty adjustment system in a role playing video game. The purpose is to aid developers in designing games that can detect and lead players into their own flow zone. We have developed theories and ways on how a system could do this, and then proceeded to make our own prototype to see if the theories would work. We then applied this to a game prototype to explore how this affects other game design choices. Results show that role playing games can adopt the system in different ways, that are all personal and individual to each game, but that the system in the end always should provide aid in game balance, and ultimately, leading the players towards their flow zone. It also resulted in some serious issues and loopholes that would require much attention if it is to be deployed in a commercial product.

Key phrases: adaptive difficulty, challenge, flow, digital games, role playing games

Denna uppsatsen utforskar möjligheterna av att använda ett dynamiskt svårighetsgradssystem i digitala rollspel. Syftet med arbetet är att ge utvecklare stöd med att designa spel som kan finna och hjälpa till att leda sina spelare till deras respektive flow-zon. Vi har utvecklat teorier och tillvägagångssätt för hur ett system skulle kunna göra detta och sedan fortsatt med att göra en egen prototyp för att vidare utforska hur det påverkar andra val i speldesign. Resultaten visar att rollspel kan anpassa sig efter systemet på ett antal olika sätt, där alla olika sätt är unika till spelet i fråga, men att systemet alltid ger stöd i spelbalansering och i slutändan leder spelaren till dess flow-zon. Det resulterade också i ett antal allvarliga problem som kräver mycket uppmärksamhet innan systemet kan användas för kommersiellt bruk.

Nyckelord: adaptiv svårighetsgrad, utmaning, flow, digitala spel, rollspel

Table of Contents

1. Introduction

From its infancy and onwards to present time, more than likely also in the future to come, video games have, in different shapes and forms, always depended on the goals and the challenges and obstacles that stand in the way of the player reaching these goals. From iron spikes and deadly gaps to enemies seemingly too large to fit on the surface of the screen, these obstacles have served as sub-motivations for the player to strengthen themselves, in their role as the world's last hope for salvation, or as the mere rotator of blocks in a game of endless wall building/collapsing.

Whatever ultimate goal the game may present to you, the importance of these motivations is quite apparent once you think about it. There's always a goal and a road towards it, filled with obstacles to overcome, whether it's a pure puzzle game in the vein of *Tetris* (1984), or if it's a game more driven by a story, like *The Elder Scrolls V: Skyrim* (2011).

The design of these challenges has in itself always been a challenge for the developers. Different types of players possess different levels of skill, so the balancing of the challenges is always difficult. Because of this, we have chosen to study ways to make a system that adapts to the player's skill level by itself, in order to deliver a more tailored experience for the entire audience.

2. Problem Area

In this chapter we will present the research question and its history, as well as all of the problems around that we deem the most important.

2.1 Background

Keeping a player interested in your digital game is something that we consider to be important and an issue that is also hard to tackle. How does one define interest and ultimately, how do we as game developers maintain that interest? By employing techniques to adjust difficulty in games, it is possible for us as developers to deliver a more tailored experience of challenges in the context of the player's performance and that in turn will increase the motivation and interest in the game.

Although video games are of young age, the medium has grown enormously since it was first popularized in the 80s. This development is not just one that we can observe from how the hardware has evolved, we can simply observe the kinds of games we have played throughout our lives, noticing how they've grown seemingly more complex and emotionally engaging, from a purely subjective standpoint. We also feel that the research revolving around games and all of its surrounding components have since the 21st century started to really take off, perhaps thanks to a growing interest in game development and the expansion of game development communities and blogs, like *Gamasutra* and *TIGSource*.

One of the most important factors when playing a game, both for players and developers, is for the player to have a stimulating experience. We can ensure that the player is stimulated by maintaining the initial motivation that we may assume that the player already has, since they have bought and started the game (Ghozland, 2007, p.1.) It is important to note that while others may use strong words of value like *fun* to describe the stimulation and motivation of the player, we feel that *fun* is a too specific word to describe the experience of stimulation and interest for a game. Instead, we replace the fun factor with a more general description of a *stimulating experience,* which we feel does a better job at encompassing the varying states of disposition that a player may have towards a game. For instance, a horror game might not rely as much on

enforcing a feeling of having fun as it does trying to maintain stimulation and interest by instead employing emotions of suspense and fear.
If the game is interesting and the player feels motivated to progress, the player will keep playing the game. This is important for the players gaming experience, and to make the player feel they get a lot of value from their invested time and money. As for the developers, it’s important for longevity. Obviously, you want to make a good game as a developer. A well thought out production motivates the team and promotes creativity, which is often prominent in the product. At least from our experiences, we’ve found that when you are motivated and having fun with the production, the result feels more polished.
A well thought out product will sell, and in turn increase monetary assets, but more importantly motivation for the team and opportunity to develop new games.

2.2 Research Question

How do you design and apply a balanced
and dynamic gameplay system in digital role playing games?

2.3 Aim

The aim of this research is to capture the essence of what makes a game interesting. Although this is a very wide subject, and is covered by multiple kinds of aspects, we will look try to look at it from a design and technical point of view. To accomplish this, we will develop and apply a method of gameplay that will adapt to the players performance. Should the system be successful, the player should have a more tailored experience.

It is important to note that we do not expect the final gameplay system to stand on its own as a complete experience. We still believe that graphics, sound and level design has its place in games, and our system is supposed to be implemented in conjunction with the aforementioned components.

We’ll be working in a team to develop a proper digital game, with audio and visual elements. With all these elements combined, we believe that the experience may, only then, be complete and immersive.

2.4 Previous research

This sub-chapter details all the research that we've used during our work. We will also be detailing how these sources have been relevant to our research, what we've learned from them and how we will/have applied it to our own research and project. Note however that we won't be going into detail about the design and workings of the system that we will develop. There will be more about that in the approach/procedure chapter instead.

Since player performance in a game depends on so many variables, such as what kind of game it is, and most importantly that some people are flat out better at games than others, there's no one true way to define what makes a game "difficult". So how do you quantify difficulty? If you want to design a system that adapts to player performance accordingly, you have to be able to quantify difficulty as some kind of value.

2.4.1 Defining challenges and difficulty

To define what difficulty is, you have to look at what modules difficulty consists of. As you will see below, we mention challenge as one of the key aspects of difficulty. We can consider that a challenge is a set of sub-games; a rule based system with variable and/or quantifiable results. Performance is the second aspect of difficulty and would then be defined as the players abilities to pass a given challenge (M.-V. Aponte, Guillaume Levieux and Stephane Natkin, 2011).

A general definition of the difficulty has to allow quantification, such as binary (win or lose) or discrete (0 to *N* points). Having a binary condition is the most common way to define the finish of a game, so we can consider that the designer of a game can always define a binary function to determine if the player has won or lost. This leads to the following definitions: a challenge is a dynamic rule based system, with two outcomes - win or lose. At a point in time a challenge can be in four states - not started, in progress, win or lose (M.-V. Aponte, Guillaume Levieux and Stephane Natkin, 2011).

So what about gameplay? Gameplay is a very abstract term that has a lot of different meanings to different people, and so we would like to give our own definition to it so that it may fit into the context of this paper. Using the previous definitions of challenge and difficulty, we can evaluate the term gameplay as the rules and mechanics of the game, and the way the player/players interact with them by the use of input.

2.4.2 Designing for role playing games

To design a difficulty adjustment system for a role playing game is different from other genres because of the nature of the genre. Obviously it has its differences to other genres, and you can use those differences for increasing and decreasing the level of challenges. Bostan and Öğüt (2009) explain that the first thing to be kept in mind when determining difficulty levels is various player styles. There's four groups of styles; Explorers, Socializers, Killers and Achievers. As the name implies, explorers want to see every corner of the world. Socializers prefer to role play and converse with friends, while Killers prefer to trouble others for their own satisfaction. If difficulty is only adjusted through numerical values such as hit points and damage output, varying difficulty will affect Socializers and Explorers a lot less - almost not at all - and numerical differences is much better suited for Achievers who focus on game-related goals like levels and treasure hunting.

Exploiting the fact that you can categorize players into different playing styles makes it easier for the developers to not only affect certain groups of players when adjusting both numerical and more abstract data, but also to balance the game. Balancing is a tedious task that's incredibly difficult, because applying a balancing action very often have consequences somewhere else, and being able to group the player base helps with focusing balancing issues.

2.4.3 DDA (Dynamic Difficulty Adjustment)

One way to adapt the experience according to the player's performance is the use of DDA, Dynamic Difficulty Adjustment. This essentially means that the game measures and evaluates the player's performance and then adjusts the difficulty accordingly. Depending on the game's systems and its gameplay, this adjustment may affect things such as enemy behavior or health, or the layout of the environment. When searching for papers and articles on the subject of adjusting the difficulty, the DDA term was not uncommon. As such, it would be the closest to an established terminology of the type of system we're seeking to design as well.

Although DDA is a general term for methods that alters the gameplay to fit the player's performance, the one described here is more focused on level design. Our system won't have a lot of impact on the level design, but more on the gameplay and statistical attributes, however the techniques used in DDA still mentions a few good points.

For the system to work, it needs some basic performance values. For this and to define if the player needs more challenge or not, a short level was introduced, and while playing it, a tool would collect data. At the end of the level, either by finishing or dying, the player was asked to answer how difficult the level was: from 1-Easy to 6-Hard. The system would then use this information together with the information gathered by the tool as input for models and systems that rank or rate things such as obstacle difficulty or player performance (Martin Jennings-Teats, Gillian Smith, Noah Wardrip-Fruin, 2010).

When using the data collected from the levels, it was believed that the difficulty in the game relates to the combination of adjacent level components, rather than the presence of a given component by itself.

2.4.4 PNRC (Player state, Needs, Rewards & Challenge)

Whilst our main goal is to design and employ an adapting difficulty system in order to tailor the experience, the player's motivation and will to continue in the game is of the essence to ensure that they even want to play the game. Motivation is the keyword, which could perhaps be defined as the balance in the player's disposition towards the game and how the game's systems treats the player. If the player feels too challenged or too rewarded, this will increase frustration and in turn work against their motivation.

For his article on Gamasutra, David Ghozland (2007, p.1.) mentions that the reward must be in proportion to the challenge and the player's expectation on the reward. As such, as Ghozland (2007, p.2.) also mentions, a system that rewards the player with the legendary Excalibur when slaying a Goblin, or opting to randomized loot based purely on a luck/chance system will lead to the rewards system collapsing. For instance, given the above example of a Goblin dropping the legendary Excalibur, being given a weapon of such strength at a point in the game where you're expected to be a lot weaker would lead to breaking the difficulty of the game. Even if a player would opt for an easier difficulty, this would more than likely bore the player due to the foes being much too easy to overcome, which in turn would decrease the motivation of the player, since the challenge factor is too low.

Surely anyone that has played an RPG from the nineties can relate to this dilemma. We remember spending sometimes full hours just battling the same type of enemies, desperately trying to get them to drop that particularly good sword or magic tome, often to no avail.

To tackle this issue, Ghozland presents a system, or formula rather, that rewards the player in accordance with the difficulty and the needs. This is what he calls the **PNRC System** or the **Motivation Loop.** This stands for *Player state, Needs, Reward* and *Challenge.* Ghozland argues that the player's motivation is dependent on these four functions. He describes these four functions as the following:

- *PLAYER STATE (P): This is the state of game variables of the player's avatar. His life, armor, and the quality of his equipment, etc. It is also his talent, his knowledge of the world and of game mechanisms. We can say that this is the strength of the player.*
- *NEEDS (N): These are the needs at the moment when the challenge arises. These needs depend on the player's state and on his advancement into the game. There are also the needs added, by the game design, as the player goes along.*
- *REWARD (R): This is the player's expectation of the reward. The value depends on the type of reward (function of the needs); it depends on the estimated difficulty and also on the player's past experience with the reward system.*
- *CHALLENGE (C): This is the player's expectation regarding the challenge. The value is high if the player believes in his own capacities. However, if he does not feel comfortable or if he doubts in his skills, the value is low.* (Ghozland, 2007, p.2)

Of course, there is the prerequisite that the player must understand the mechanics and rules of the game world in order to have an expectation of its challenges and rewards. If we don't know the rules of the game, how should we know what kind of rewards to expect? This is most commonly handled through a series of tutorials at the start of the game, which should ideally familiarize the player with the game's mechanics *and* what they should expect in terms of challenges and rewards.

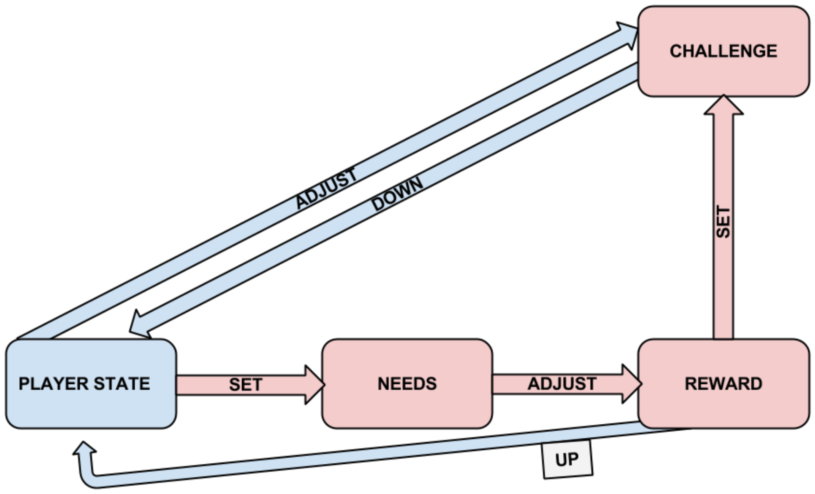

Fig. 1 The PNRC Loop

The values of the PNRC-model are dependent on each other in different manners, depending on, for instance, the game's genre and what established conventions they sport, Ghozland means (2007, p.3.) In one of his examples on how RTS (Real Time Strategy) games approach the motivation loop, Ghozland states that *StarCraft (Blizzard Entertainment, 1998)* at its core is based around acquiring resources and controlling territories or areas. At the very start of a mission, the player state (P) is low, since the player starts usually with just a small base and some basic troops. Therefore, the need (N) is very high, the player wants to become stronger in order to expand their base and conquer the enemy, which relates back to the core goals of acquiring resources and controlling territory. Expectations on the rewards (R) are high. The player expects their hard work to pay off in the form of stronger armies and bases. The challenge (C) is expected to be scaled to the player, though a little bit stronger, as to actually present a challenge rather than a stalemate in terms of skills. As such, the initial motivation in a *StarCraft* mission is very high.

Of course, such a system is not without its flaws. As Ghozland mentions, the game is decided once all resources are exhausted. From there on, the player's motivation decreases. Since it requires an actual victory or defeat for the game to end, in a multiplayer match, it might lead to the losing player leaving the game, which is frustrating to the other player. Winning a match through the other player leaving is not nearly as satisfactory as winning by putting your acquired resources (and thus, the time you spent) to use and defeating your opponent.

Ghozland (2007, p.4-5) also provides examples on the application of the Motivational Loop in practice. More specifically, he provides examples of how to maintain the motivational factor with the help of standard mechanics, such as game score, which traditionally is a numeral value that increases when you collect items or kill enemies in a game. In the same section, he also mentions the pros and cons of an experience points system, which in essence is a derivative of the usual score system. The pros of experience points is that it has a more direct tie to the game system and character progression, whilst it has a downside in that it comes with a limitation, a max level to be more precise.

2.4.5 The motivational loop in RPG's and a possible application

This motivational loop could very well be extremely useful to our application in a digital RPG. The RPG genre as a whole does often sport a reliance on loot and reward systems. Think of an action RPG, like for instance *Diablo (Blizzard Entertainment, 1996)* or *Torchlight (Runic Games, 2009)*. Both of the aforementioned have an emphasis on their loot system and the gameplay relies a lot on this system to stimulate and keep the interest of the player. As a player, you may find it sparks interest in the game even during the less interesting dungeons or quests, as it grants an opportunity for you to strengthen your character and see how the newly acquired "Wand of Life Stealing" or the like changes the pacing and/or circumstances for your character during battle. Alongside the equally or more so emphasized character building/experience system, these two work very well together, which might be the reason for the combination of them being seen in a lot of RPG's.

The combination of the aforementioned systems also provides at least a slightly longer lifetime for the player's motivation in that, when you reach the apex of the experience system, the max level, you may still find loot that strengthens your character even further, at least for a while. Such a reliance on the loot system as seen in the aforementioned games does come with a downside as well, in that the sheer volume of the loot results in sometimes just one weapon out of the thirty you just picked up being actually useful.

There are several ways in which we can employ the motivational loop to our system. By measuring the needs of the player, we may for instance find that they are low on health but don’t have any potions to heal themselves with. This way, we can tell the system to spawn the next enemy carrying a health potion that should get them back into the comfort of not being nearly dead.
By comparing the player state to the challenge being presented, we may also find less straightforward ways to help the player. Take for instance a boss battle that proves rather difficult for the player given their current player state (equipment, experience, etc.) If the player retreats back to the easier enemies, we could inform the system that the player could use some loot that improves their chances towards the difficult battle ahead. This could be done either by spawning enemies that drop the sort of ingredients that the player may need to improve their weapon and armor, or even by spawning a little treasure chest with useful items off the main road for the player to find.

2.5 Research summary and future adaptation

Throughout the researching phase, we found that there was a lot of useful information to gather, not just on dynamic difficulty adjustment systems, but also on how to keep the interest of the player alive and well. By broadening the research we could take more factors into account, which will hopefully mean that our own system will feel well tailored for the player. Of course, depending on too many variables could potentially mean that the system will be hard to keep track of, but we feel that we have found the right amount and types of variables necessary for us to be able to shape and employ our system without too much hassle.

To develop our system, we first needed to define a few key terms, for ourselves and for the readers. Broad terms such as gameplay demanded that we explained it in the way that we perceive it, whilst we had our research sources to help us define other terms such as challenges and difficulty. By defining these terms, we also have a solid base through which we can abstract the very essence of the system and get a better overview of how they are all connected.

Looking at Martin Jennings-Teats, Gillian Smith and Noah Wardrip-Fruin’s model will help us to develop our model further than basic tweaking of numerical parameters. Although their model is particularly designed for alteration of level elements in mind, which is potentially much more difficult in a 3D environment than a 2D environment, they have adopted an interesting approach of collecting the players' performance in data, and analyzing it. Looking at their model will make it a lot easier for us to form our own tool for data collection. We can also use their knowledge of potential loopholes to ease our process of development.

Taking Ghozland's motivational model into account could potentially greatly simplify the handling of the challenges and rewards of our game, given that we find the right means to incorporate the values into our system. Working alongside the dynamic difficulty adjustment, it could be used to pass very useful values and ultimately directives to the DDA on how many more or less, stronger or weaker enemies to spawn in the path ahead.

3. Process

This section of the thesis will present our methods of choice and the procedure to get the result we got. We will also show and explain our reasoning behind the choices we've made along our way, regarding everything from planning and research to the actual production.

3.1 Overview

The primary focus of our work was and had always been planned to be the actual development and implementation of our own dynamic difficulty adjustment system. Because of this, we've developed a game that implements our adjustment system. A practical approach enables us to test the system within a game environment, as it was intended to function. One could argue that employing the system in a more conceptual form, on its own outside of an actual game environment, would enable us to more thoroughly test the components of the system. However, we felt that such an approach would fall short in the long run. We were given the opportunity to put not just our most recent learnings to use, but all of those that had come before it and we decided to follow that path. This is why we opted for the development of a full game, along with two other groups of students, though our current research focus was given the highest priority, of course. We also believe that it enhances the impression of the system, when it is part of a larger whole as it would be in a game development situation. In the end we ended up with our own difficulty adjustment system, "*Player adjusted difficulty",* from here on denominated PAD.

3.2 Methods for ideation

Alongside the research phase of our essay writing we also started planning the elements of the actual game we were making with the other two groups. Before getting started on writing down specifics such as the story of the game, its visual style, etc., we had to come up with the basic idea; what kind of game we wanted to make. What genre and theme would the game be? Our methods of ideation were largely composed of making mind maps and brainstorming, both of which we'll describe below.

Making mind maps (John W. Budd, 2004) involves coming up with a core word of sorts, around which you make several branching lines with other words that come up that are associated with the core word. For instance, if a team were generating ideas for a game's storyline, let's say that they arrive at a genre as their core word for their mind map. From this genre, let's use science fiction as an example; they would branch out lines as they came up with ideas or words surrounding that given word. Someone might suggest a dystopian setting and from that word someone might suggest a post-apocalyptic, metropolitan environment and so on, so forth.

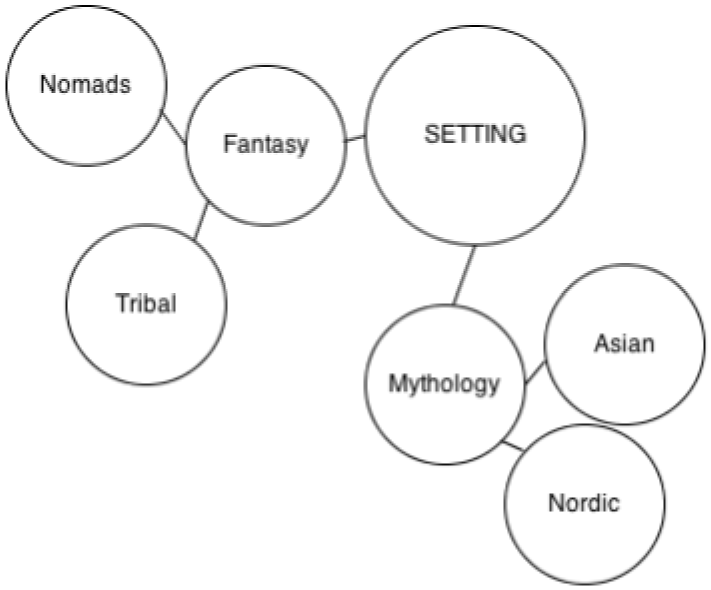

Fig. 2 An example of a mind map

Brainstorming is largely similar to mind maps, which should be of no surprise since as of late, it has become more of a common denominator for creative ideation in groups or individually. What is important to note about brainstorming is its philosophy, which is how we've employed it during our work. Alex F Osborn, in many ways the forefather of such creative problem solving and ideation, came up with a sort of basic rule set that should be followed for all brainstorming sessions (Alex F. Osborn, 1963).
These involve; a focus on the quantity of ideas, meaning that the group should not withhold any ideas that they come up with.
Withholding criticism is also favored, as the focus of brainstorming is to generate ideas and later iterate and refine them. By withholding criticism, Osborn also argues that participants will feel more comfortable with coming up with more unusual ideas.
The brainstorming and its participants should also welcome these unusual ideas. By looking at issues from new perspectives and suspending assumptions, they may provide better solutions.
Lastly, the combination of ideas is believed to be of great use, as this enables the team to combine several good and similar ideas into one larger idea or concept.

We employed these methods in symbiosis, combining the practical procedure of the mind map method with the philosophy of the aforementioned rules of the brainstorming. We began with a basic idea that we had come up with before; that the game would be a role playing game. From there we started writing down all ideas that came to mind. When we felt satisfied with the amount of ideas we had, we started isolating the things we'd written down that we wanted to use, while also combining the ideas that were similar or could easily be grouped together.
From this method, we eventually landed at having the basic setting and theme of the game, from which we could then start writing the game design document, which we'll get to in the following section.

3.3 Methods for planning

When we had established a foundation from which we could further develop ideas for the game, we started organizing and writing them down in the form of a game design document, looking at different templates from across the internet to get an idea of what sections we would have use for. When the basic layout of the game design document was ready, we started writing down and elaborating upon the ideas that we had come up with during the ideation sessions.

For those not familiar with the term, a game design document could best be described as the following, written by Tim Ryan for the blog Gamasutra:

"A design document is a bible from which the producer preaches the goal, through which the designers champion their ideas, and from which the artists and programmers get their instructions and express their expertise."

Essentially, it's a compound of descriptions of several essential components, such as the game's art direction, its story (usually in the form of a synopsis), how the game answers to input, etc. These components differ depending on the type of game and the people writing it, since it dictates what would be deemed essential by the conventions of the genre and what the authors themselves deem important. The team then uses the respective areas of document relevant to their roles, as guidelines for developing the game, to ensure that everyone follows the same vision. For instance, a gameplay programmer will refer to the design document to find out what buttons should be used for steering the character. (Gamasutra, 1999)

When planning the actual PAD system, we used a type of diagram called flowcharts to illustrate the different components of the system and their relation to and effect on one another. A flowchart is just that; a diagram that details a process or a solution to a problem by illustrating the components of the system, how they're related to each other and how they affect each other. Although there are standardized conventions for the different types of flowcharts (Australia Standards, 1985), we employed a simpler approach.

We used flowcharts for describing the components of the PAD system itself and which variables they would be composed of. This way, we had an overview of the available variables of the system and could thus plan for how they would affect the difficulty in the game and when.
The different, proposed ways for the system to affect difficulty was another field in which a flowchart proved to be very useful. It provided us with a good overview of some ways in which

we could employ our system and also would inspire us to come up with more ideas for adjustment events.

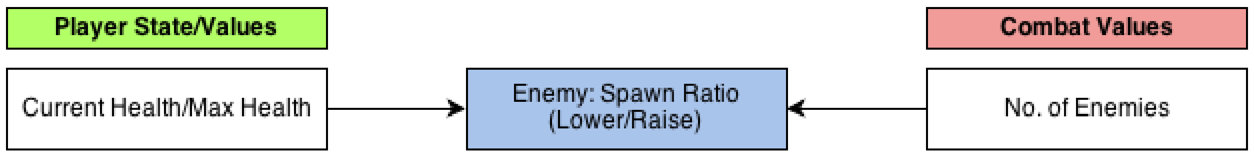

Fig. 3 A sample of a simple flowchart used to describe how the player's current health versus the number of enemies present dictates the spawning ratio of new enemies, as part of PAD system.

3.4 Methods for production

Developing games is an incredibly creative process, which means that sequential development models such as *Waterfall* usually are discarded - as is in our case. Ultimately, we used a mix of two different development models that are in the forefront of software development.

The first model we took influence from is *Scrum*, which is an agile development pattern that allows for reiteration of the product, so that the goal of the product is always shared between the members of the team. In scrum, a development phase is called a *sprint*. Sprints last for a set amount of time - typically around a month (varying to fit the project in question). It also features roles like Managers and ScrumMasters, along with a few more ancillary roles, all of which deal with different issues and topics within the model. From this model we based most of our planning - a sprint was a week long, however we handled our sprints slightly differently from the traditional scrum model. Instead of putting our topics and work-to-be-done on paper and moving them into different stages, we reevaluated the project planning with the team in the beginning of each sprint, and combined our topics with a traditional priority list. This allowed for the creative iterative process to be kept alive, while the priorities were kept straight as time passed. In addition to this, we didn't assign any typical scrum roles, but instead decided to let each team deal with their own scheduling according to their area of development, and then sync them up so it fit with the overall schedule of the project. (Schwaber, K., 2004)

The second model we used is a much more broad perspective model that strengthens the outline of production model. While the scrum methodology brings structure to the project as a whole,

the *Iterative and incremental development* combination of methods gives us structure within each sprint - every day of a sprint we iterate through what needs to be done to determine what is required for the sprint to be completed within the set timeframe. This allows us to reflect and review what has happened at different points in time in a sprint, which in turn allows us to adapt and change priorities as new problems and solutions arise.
A lot of agile development models are based off of this model because of its iterative nature. While it is not considered a model by itself, it described our workflow best together with the simplified scrum model. It has also seen widespread use throughout software development history, which further speaks for its advantages. (Larman, C. & Basili, Victor R., 2003)

3.5 Technology of choice

We chose to use an already existing game engine and editor, *Unity 4* (Unity Technologies, 2013), because it allows us to work with ready-to-use content and assets that comes with the engine, as well as take advantage of the Unity Asset Store that comes along with it - which allows users to buy and sell pieces of game-ready systems, models, textures and other game assets. This proved useful, since we found scripts for subsystems that were needed for the game. It also allows us spend time developing our difficulty adjustment system, instead of the game we need to apply it on. Unity is however a very open engine, in the sense that it allows us to produce any game we want - we never felt restrained by the game engine or otherwise - it simply allowed for a more efficient workflow and increased time spent on developing PAD. It also features intuitive ways to extend the editor, allowing us to make our own tools that later helped us to create content for the game. Unity also has the ability to easily prepare the game for distribution on multiple platforms, which meant we never had to worry about user input being an issue.

We also have previous experience with the engine which allows for a smoother workflow rather than using a completely new framework - which in turn would require even more time to adjust to, to use in an efficient manner. Unity is also a well-established game engine, which means it comes with a big community, making it easy to find content on the Unity Asset Store to act substitute to self-developed systems. Given the timeframe allowed to develop our game, we deemed Unity the most optimal tool to use.

To increase team availability and ease management of parallel development, we used *Git*, an open-source decentralized version control system (DVCS). A version control system (VCS) is a program that supports creation of different versions of software. Git has support for rapid branching (creating a new version) and merging (unifying multiple versions) of the project, making it easy to develop in a non-linear fashion. Compared to its centralized counterparts, DVCS's don't operate towards a central server - instead each client is its own separate system, which means a client never impairs another client.

Ultimately, the two systems in conjunction were meant to allow for parallel development and to save time when unifying features, and were therefore deemed the best fit.

4. Result and discussion

The following section will contain detailed information on what results we have reached, how we reached it and how one could go forward from where we have left off. We will also be discussing the result, how the system ended up working and why we chose to develop it as we did.

4.1 Result

Before we could get started on developing the PAD system, we had to collectively, in the group, establish the foundation for the game itself, in order to know how to target the PAD systems features.

This process largely consisted of writing the game's design document, through which we would have a point of reference for developing these foundations. The writing of the design document was then followed by laying out the base for the game's control and combat system and shaping the basic layout for the game world, so that we would have a basic overview of the game both technically and visually. With a solid foundation, we could starting building upon it with all the proper assets, one of which was our own PAD system.

4.1.1 The philosophy of the PAD system

There is a psychological expression called *Flow*, which describes a mental state where the person is feeling completely immersed and has full involvement and enjoyment in the process of the activity. It is a concept that is applied to a lot of different fields, and is described as completely focused motivation (Csíkszentmihályi, M.).

Csíkszentmihályi mentions eight major components of Flow:

- A challenging activity requiring skills
- A merging of action and awareness
- Clear goals
- Direct, immediate feedback
- Concentration on the task at hand
- A sense of control
- A loss of self-consciousness
- An altered sense of time

(Chen, J., 2007)

It was argued that the antecedent factors of flow are interconnected - for example a perceived balance between challenges and skills requires that one knows what he or she has to do (clear goals), and how successful he or she is in doing it (immediate feedback). Thus, a perceived accommodation of skills and task demands can be identified as the central precondition of flow experiences. (Keller, J., & Landhäußer, A., 2012).

Flow, however, does not have to include all of the components - they are merely components that may be combined in different ways from which Flow arise. A flow state can be entered when performing almost any activity, although it is much more likely to occur when performing a task with intrinsic purposes. (Chen, J., 2007)

When treated as content, the definition of flow is too broad. However, if applied correctly, it can happen in every game. To make a game special, however, requires content that is more refined than a flow experience.

If you instead treat flow as a system, it explains why people prefer some games over others, and how they become addicted towards these games. If a games meets all the core elemental requirements of flow, any game becomes fun - as proven by simple games like *Arkanoid (Taito, 1986)* as well as complex games like *StarCraft II (Blizzard Entertainment, 2010)*. (Chen J., 2005)

Flow is not a state that can be enforced; it is merely a state that is entered when the challenge is appropriate to the skill level of the particular task at hand, for each person. As illustrated by the graph in fig. 3, Flow is more likely to occur in cases where the challenge at hand is more difficult than average, and when the skill level of the individual is above average. The further from the center an experience is, the greater the intensity of that state. (Csíkszentmihályi, M. 1988)

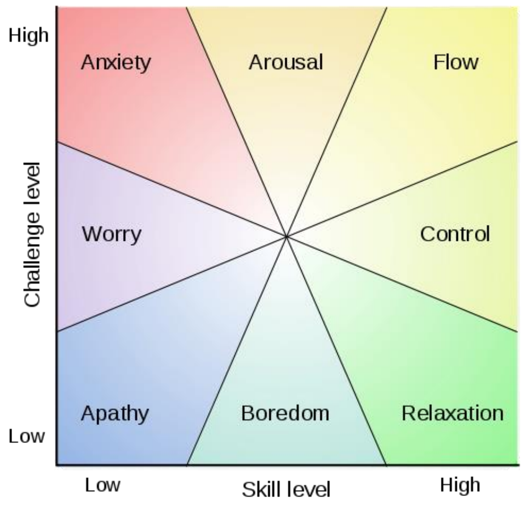

Fig. 3 The different mental sates

Flow is one of the main reasons people play videogames, and the purpose of PAD is to emphasize that - to help the designers of a game push each individual player towards their own flow state. It is the original motivation for the development of PAD, and the primary effect that the PAD system should invoke. If the PAD system is balanced the right way, it should help the player achieve the flow state, since the challenge in the game is proportional to the player's skills. Thus the player should be fully immersed in the game experience.

4.1.2 Planning

The planning of the PAD system consisted largely of outlining the components of the system itself and outlining several suggested relations through which adjustment events in the game would take place.

An adjustment event is the time during the game in which the adjustment system has gathered and analyzed information. It then uses that information to determine if aspects of the difficulty (such as the rate at which enemies appear, for example) should be adjusted, that is increased or decreased, typically.

When designing the system as a whole, we chose a modular system design because of its scalability and ease of change. By choosing this layout, we gain flexibility to alter the input and output of the system without having to affect the system as a whole, allowing for highly customizable components, as long as the components follow the rules and guidelines of each component type. This design is employed in the system as a whole, but also in the adjuster module (as described in chapter 4.1.3) itself, making all adjuster modules standalone - which in turn makes it very easy to expand the PAD system as a whole.

Parallel to the planning of the system, we worked on putting together a basic foundation of sorts for the game. With the base for the game and the flowcharts ready, we had a clear vision of how we would proceed with developing the actual system.

4.1.3 Components of the system

We opted for designing a system that was modular in its features, as discussed in the section above. Following the plan for the layout of the system, we ended up having three different modules that consecutively processed and/or generated new data from the previous data. The three modules of the system were as follows:

- The data collector - This module collects and stores the relevant data that the system needs to determine whether the difficulty needs to be adjusted. The data that is being collected involves the player and enemy stats, such as health, strength and so on. When a data collection session is finished, the data is sent to the next module, the analyzer.
- The data analyzer - The data that is stored in the collector is processed by the analyzer module. The analyzer takes the raw data collected and compares it in different manners. This data comparison results in a set of new variables that measure such things as performance ratio. The given example compares the time that the player was expected to take defeating the foes versus the time that they actually took defeating them. This new data is then sent to the final module, the adjuster module.
- The difficulty adjuster - This is where the actual adjustment of the difficulty happens. The adjuster module is built up of several rule sets of sorts, which determine what, when

and how to alter the difficulty. Again using the previously mentioned example of the measured performance ratio of the analyzer module, the system would for instance find that the player’s performance was well above the expected one (a percentage value above 100%). With this information, the system communicates with the spawning point of the enemies, telling it to throw more enemies at the player since they’re performing very well.

In addition to the three components, there’s a master component that acts as a manager to all of the other components. It gives the system extra structure, and makes resources available from a common location.

The system is linear in its execution and the modules are interdependent. This is advantageous to us due to the fact that the system is very easy to follow, it goes in a step-by-step pattern and the respective names of the modules are clear. In a way, the system itself is also adaptive, since the layout and components of the system could be adapted to work with games other than our own. With a bit of work, it could be repurposed to fit games outside of the RPG genre as well, so long as the pattern of collection - analyzing - adjustment remains, as it is the systems very essence.

4.1.4 Implementation

While not part of the actual PAD system, it is relevant to bring up how things such as health and power scale in the game. This is a point that is very specific to each and every game, and is more or less impossible to abstract to a level that would work for any game. In our game, *Eira’s Tale,* all entities have levels. A level is basically a value that reflects the overall strength of an entity. Since PAD is working closely with these values, it is required to keep in mind how all of this works.

At a point in development, we faced a situation where we had to make a decision between letting PAD do all of the adjustments to values. This would make PAD a truly adaptive feature - however it limits the game to the point that all enemies could be of the same worth - whether it be a big dragon or a small mouse. Instead, we opted to go with a more traditional way of value scaling in role playing games.

The level dictates all the other values of an entity, so as the level increases, so does your health, strength and resistances. Level values themselves have a maximum value, a value that should be tailored for each particular game. Essentially, this gives us as designers much more control in how the game works. We tell PAD what boundaries exist for each enemy, and they don't even have to be the same for all enemies - and then PAD will be free to operate as it wishes within those boundaries. This comes with a tradeoff, of course. If we want an adjuster to increase the offensive values of an enemy, it also has to increase the defensive ones, since the only value it can change is the level. It cripples the system in the sense that it removes some of the adaptive aspects, but it's a tradeoff we deemed worthy because of the control you get in return.

However, the scenario described previously is only applied to entity strength adjusters – values such as spawning frequency of enemies are free values for PAD to change and does not have boundaries, as they are not part of the level system.

4.2 Discussion

This section will discuss how one could go forward with the work from where we have left off. This includes further development of our own PAD system, how it could be expanded upon and adapted to work in more situations or even how it could be reworked to work in more situations. Aside from the system itself, we will also discuss the possibilities of expanding upon our research and how that could help with the development of a similar system.

4.2.1 Further development of the PAD system

Although the PAD system is working, it is well in the early stages of a fully fit system that would work in a commercial situation. The biggest problem with the system right now is that has no safety mechanism for cheating - you could underperform by choice, to reduce the overall challenge of the game. This problem completely negates the philosophy of the system and is therefore the most crucial point to fix. It is, however, also one of the more difficult problems to fix. How do you come up with a filtering technique that could remove performance entries that shouldn't be permitted? You could use a base sample of the first N performance entries to provide an average performance rating, but even then you could cheat that and just perform particularly bad in the beginning, which would make the whole system assume the skill level of the individual is lower than it really is.

Another point that would need improvements is how to rate performance. At this point in time, the system uses a self-developed, simple algorithm that calculates a performance rating based on the time you kill an enemy, against the time expected to kill it. It works well to prove the system works, but for a bigger project an algorithm that could use more variables to calculate performance rating would be required. The biggest issue with the current algorithm is that has a gigantic margin of error - if something happens mid combat that requires you to not hit for a second or two, PAD has no clue about it and will still calculate that as ‘combat time’. A proven ranking system like Elo could be implemented to prevent this, or have a non-linear falloff of the performance rating that takes in more variables could also help.

4.2.2 Possible future research

Although one could argue that our research is well expansive on its own, there is of course always room for further improvements.

One such way that the research work could be expanded upon would be to branch it out across genres, to tackle more than just the area of digital role playing games. Since roleplaying games are very dependent on numerical values, it is relatively simple to know which ones could be adjusted to affect the perceived difficulty of the game.
In the case of, for instance, a platform game, such an adaptive difficulty system could potentially become more complex and a lot more directly visible to the player. One way to adjust the difficulty of a platform game would be to simplify obstacles, if the system would find that the player had difficulties passing them. For example, let’s take the classic example of a dangerous gap that you need perfect timing to get across. If the system found that the player failed to cross this gap, it could move the platforms on either side of the gap closer to each other, making it easier to jump across. The motivational loop, PNRC, could again be useful here, in order to determine the affected elements of the game. Like the aforementioned narrowing of a gap, one could also scale the reward (R in the motivational loop) given to the player after crossing such an obstacle. If the player would successfully cross the gap on the first try, we could tell the system to give the player loads of coins to collect as a reward. If the player had difficulties crossing the gap, we could instead give the player an item that increases their jump length temporarily as opposed to a pile of shiny coins.

This is just one example of the ocean of genres out there. All of them sport different conventions, from which further research could reveal potentially endless ways to rework such a system. Another way to expand upon our research would be to get more into detail with the relation between the theory of Flow and how the challenges of the game versus the skills of the player are related to that theory. This presents the potential to collaterally generalize the actual difficulty adjustment system to focus more on fulfilling one or more of the prerequisites from which the player transcends into the Flow state of mind. Extensive theoretical work could potentially lead to finding a globally adaptable challenge and player skill measurement formula, though this would probably require several more months of work and intricate evaluation, if not years of it.

5. Glossary

AI (Artificial Intelligence)
Used in video games to produce an illusion of intelligence and simulate different kinds of behavior on entities that aren't controlled by players.

Environment
(Game) Environment is a term used when describing the visual environment in the game, such as landscapes or other visual objects. Usually this encompasses all the static, non-interactive visual elements of the game world.

EXP (Experience Points)
A numerical value which represents the players progress in a game world. Traditionally experience is used as a reward for completing different objectives.

Levels (Experience)
Experience levels are a common method of increasing the player character's attributes, such as health, defense and strength. In relation to EXP, experience levels are commonly given as the player reaches a certain amount of EXP. The required amount of EXP to level up, as it is called, increases exponentially with every level gained.

Levels
A level is a series of adjacent objects that together make up an environment, complete with obstacles and a goal for the given level. Games usually consist of several levels and each one must be completed to progress further.

Game Mechanics
A term used to describe a rule (mechanic) or collection of rules (mechanics) of the game. These mechanics determine the features of the game and how the player may interact and manipulate them.

Gaming Experience

A term used to describe the feelings of the player as he or she interacts with the game.

Hit Points / Health Points / Health

A numerical value which represents the players or characters health. Traditionally the character dies if the value reaches 0.

Immersive/Immersion

A term describing how engrossed the player is in the game world and the game experience while they are playing.

Loot

A term used to describe items in the game that the player receives through overcoming challenges. This could be as simple as defeating an enemy or by completing a more complex quest/mission. Loot is often tiered into different levels of rarity, meaning how easy or hard they are to get a hold of (usually decided by the chance of the loot spawning/being dropped.)

Multiplayer

A game mode where more than one player can play in the same environment at the same time.

RPG (Role Playing Game)

A genre of games where the player assumes the role of a character in a fictional environment. RPG’s often feature character development mechanics that increase the strength of the character the more that you play the game and overcome different challenges.

RTS (Real time strategy)

A sub-genre in strategy type video games that doesn’t progress in turns.

6. Bibliography

Aponte, M. et al. (2011) Measuring the level of difficulty in single player video games. *Entertainment Computing*, 2 (4), p.205-213. [Accessed: 11th February 2013].

Arkanoid (1986) [video game]. Japan: Taito

Australia Standards. (1985) *Information processing - Documentation symbols and conventions for data, program and system flowcharts, program network charts and system resources charts.* Homebush, NSW: Standards Association of Australia.

Bostan, B. and Öğüt, S. (2009) *Game Challenges and Difficulty Levels: Lessons Learned From RPGs*. [report] p.6.

Chen, J. (2007) Flow in Games (and everything else). *Communications of the ACM*, 50 (4), p.31-34.

Csikszentmihalyi, M. and Csikszentmihalyi, I. (1988) *Optimal experience*. Cambridge: Cambridge University Press.

Diablo (1996) [video game]. USA: Blizzard Entertainment

Keller, J., & Landhäußer, A. (2012) *The flow model revisited. In S. Engeser (Ed.), Advances in flow research (pp. 51-64).* New York: Springer

Gamasutra.com (1999) *The Anatomy of a Design Document, Part 1: Documentation Guidelines for the Game Concept and Proposal*. [online] Available at: http://www.gamasutra.com/view/feature/3384/the_anatomy_of_a_design_document_.php [Accessed: 10 May 2013].

Ghozland, D. (2007) Designing for Motivation. *Gamasutra*, [blog] 7th June, Available at: http://www.gamasutra.com/view/feature/129852/designing_for_motivation.php [Accessed: 11th February 2013].

Goleman, D. (2006). *Emotional intelligence*. New York, Bantam Books.

Jennings-Teats, M. et al. (2010) *Polymorph: Dynamic Difficulty Adjustment Through Level Generation*. [report] Santa Cruz: Expressive Intelligence Studio, p.1-3.

Jenovachen.com (2005) *Welcome to Flow in Games*. [online] Available at: http://jenovachen.com/flowingames/designfig.htm [Accessed: 24 Apr 2013].

John W. Budd (2004) *Mind Maps As Classroom Exercises*, *The Journal of Economic Education*, 35:1, p.35

Larman, C. & Basili, Victor R. (2003) Iterative and Incremental Development: A Brief History, *Computer*, [online] Available at: http://www.craiglarman.com/wiki/downloads/misc/history-of-iterative-larman-and-basili-ieee-computer.pdf [Accessed: 8 June 2013]

Osborn, A. (1963) *Applied imagination; principles and procedures of creative problem-solving*. New York: Scribner.

Schwaber, K. (2004) *Agile project management with Scrum*. Redmond, Wash.: Microsoft Press.

StarCraft (1998) [video game]. USA: Blizzard Entertainment

StarCraft II: Wings of Liberty (2010) [video game]. USA: Blizzard Entertainment

Tetris (1984) [video game]. USSR: Alexey Pajitnov

The Elder Scrolls V: Skyrim (2011) [video game]. USA: Bethesda Game Studios

Torchlight (2009) [video game]. USA: Runic Games

Um, S. et al. (2007) Dynamic Difficulty Controlling Game System. IEEE Transactions on Consumer Electronics, 53 (2), p.812-818.