Game Design Document Format For Video Games With Passive Dynamic Difficulty Adjustment

Pratama Wirya Atmaja[1], Daniel Oranova Siahaan[2], dan Imam Kuswardayan[3]
[1]Teknik Informatika UPN "Veteran" Jawa Timur, Surabaya
[2,3]Teknik Informatika ITS, Surabaya

E-mail: [1]pratamawiryaatmaja_kuliah@yahoo.com

Abstrak

Permainan video adalah perangkat lunak hiburan, sehingga kepuasan pemainnya adalah tolok ukur utama kualitasnya. Satu elemen penting dari kepuasan pemain adalah tingkat kesulitan yang tepat, yang tidak terlalu mudah maupun sukar. Dewasa ini, cara termutakhir untuk menerapkan tingkat kesulitan yang tepat adalah Pengaturan Kesulitan Dinamis (PKD), yang dapat memodifikasi tingkat kesulitan permainan pada saat run-time. Tipe PKD yang paling populer pada saat ini adalah PKD pasif. Di sisi lain, Dokumen Desain Permainan (DDP) adalah artefak penting dalam pengembangan perangkat lunak permainan video, dan belum ditemukan format DDP yang mendukung perancangan mekanisme PKD pasif. Tujuan penelitian ini adalah menemukan format DDP baru yang mendukung perancangan tersebut. Kami memodifikasi sebuah format DDP yang bersifat umum dengan menambahkan bagian-bagian baru untuk perancangan mekanisme PKD pasif. Format hasil modifikasi tersebut diuji dalam proses pengujian yang melibatkan sejumlah pengembang dan sejumlah pemain. Para pengembang mengembangkan sejumlah permainan video menggunakan format DDP yang dimodifikasi dan format yang umum. Proses pengembangan yang mereka jalani diamati dan dinilai kelancarannya. Permainan-permainan video yang dihasilkan dengan kedua format DDP dimainkan oleh para pemain untuk menguji kualitas mekanisme PKD pasifnya. Hasil pengujian oleh para pengembang menyatakan bahwa format DDP yang dimodifikasi lebih baik dari format yang umum. Hasil pengujian oleh para pemain menunjukkan keunggulan permainan-permainan video yang dihasilkan dengan format DDP yang dimodifikasi, walau keunggulan itu tidak signifikan. Berdasarkan hasil tersebut, kami menyatakan bahwa format DDP yang dimodifikasi berhasil mencapai tujuannya.

Kata kunci: *permainan video, rekayasa kebutuhan, Pengaturan Kesulitan Dinamis, dokumen desain permainan, pengembangan perangkat lunak.*

Abstract

Video game is a type of entertainment software, and therefore the satisfaction of its players is the primary mean to measure its quality. One important element of player's satisfaction is a proper difficulty level, which is neither too easy nor too hard. The current state-of-the-art way to implement it is with Dynamic Difficulty Adjustment (DDA), which allows the difficulty level of a video game to be adjusted at run-time. Currently, the most popular type of DDA is the passive one. Meanwhile, Game Design Document (GDD) is an important artefact in the development process of a video game software, and there is still no GDD format that supports the design of passive DDA mechanism. The aim of this research was to find a new GDD format that supports the mechanism. We modified a general purpose GDD format by adding new parts for designing passive DDA mechanism. We tested the usefulness of the modified format in a testing process involving developers and players. The developers developed video games using the modified GDD format and the general purpose one. Their development processes were observed and evaluated to know if there were any difficulties. The resulting video games were played by the players to find which are better in terms of passive DDA mechanism. The result of developer testing showed that the modified format is better than the general purpose one. The result of player testing showed that the video games made with the modified format are better than their counterparts, albeit by an insignificant margin. Based on the results, we declare that the modified GDD format is successful.

Keywords: Video game, requirement engineering, game design document, dynamic difficulty adjustment, software development.

1. Introduction

Video game is a type of software intended as an entertainment medium, and therefore the satisfaction of its players is the primary mean to measure its quality (Callele & Neufeld, 2005). A method for measuring player's satisfaction has been found in the form of GameFlow model, which was adapted from Flow theory (Sweetser & Wyeth, 2005). One important element of GameFlow is a proper difficulty level, which means a player should neither become bored from playing a too easy game nor become frustrated from playing a too hard one (Chen, Flow in Games (and everything else), 2007).

Nowadays, the state-of-the-art way to implement a proper difficulty level is through Dynamic Difficulty Adjustment (DDA) mechanism (Chen, Flow in Games, 2006). The mechanism allows the difficulty level of a video game to be adjusted at run-time, thus making it easier to ensure that every player gets the right difficulty level. Currently, the most popular type of DDA is the passive one, which does not directly involve players in its execution (Alexander, Oikonomou, & Sear, 2013) (Arulraj, 2010) (Hao, He, Wang, Liu, Yang, & Huang, 2010) (Sha, et al., 2010) (Wu, Chen, He, Sun, Li, & Zhao, 2011) (Yu, et al., 2010).

In the development process of a video game software, the requirement engineering stage plays a crucial part in determining the success of the process (Bethke, 2003) (Callele & Neufeld, 2005). Therefore, the stage needs to be performed carefully by the developer. The most important requirement engineering artefact of a video game is Game Design Document (GDD), which records the detailed design of the video game's gameplay.

In relation to passive DDA mechanism, there is still no GDD format that supports it. To improve the situation, we set to find a new, specialized GDD format for the purpose.

1.1. Video Game Development Process

In principle, the development process of a video game is no different from that of other kinds of software. If the developer follows waterfall methodology, the process will then be split into three stages (Bethke, 2003):

1). Pre-production, which consists of:
   a. Determining business parameters;
   b. Determining the basic concept of the video game;
   c. Writing Vision Document;
   d. Writing Game Design Document to design the video game's gameplay;
   e. Writing Technical Design Document to design the technical implementation of the gameplay;
2). Production, which consists of:
   a. Performing design implementation;
   b. Developing the early version of the video game;
   c. Developing the alpha version of the video game;
   d. Developing the beta version of the video game;
   e. Developing the final candidate of the video game;
3). Post-production, which consists of steps related to releasing the video game.

Pure waterfall methodology, which requires the developer to follow the steps linearly, has gained a negative reputation in video game industry. Because the quality of a video game depends on its players' satisfaction, performing a strictly linear development process is not realistic because such subjective satisfaction is hard to pinpoint at the beginning of the process. A more preferred alternative is a modified waterfall one (Royce, 1970), which allows the developer to leave the production stage and revisit pre-production in case the video game's design needs to be revised.

1.2. Game Design Document

GDD is an artefact commonly created during the pre-production stage of a video game. The document records a detailed design of the gameplay players will get to experience. In a video game pre-production workflow, GDD is positioned after Vision Document, which outlines the concept of the video game, and before Technical Design Document (TDD), which translates the contents of GDD into a more technical language that will be used in the production stage (Bethke, 2003).

So far, there is no unified GDD format for all purposes. Every developer in the industry may create their own format that suits their needs. On the other hand, Salazar et al. has created a GDD format that follows best practices of Software Requirement Specification (Salazar, Mitre, Olalde, & Sanchez,

2012). The format is intended for video games in general, although some types of video game might have special characteristics not supported by it. To solve the problem, the format can be adjusted to suit a specific type of video game.

Arif (Arif, 2014) modified the general GDD format to make it more suited for developing educational games. The resulting format was tested by two developers, each used it to develop an educational video game. Both games were the results of reverse engineering of two other, earlier educational games. All four games were then tested by players in relation to their educational aspect, and the new ones were rated higher than the earlier ones. Both developers also rated the modified format positively for being useful and helpful. It can therefore be said that the adjustment was successful.

1.3. Flow Theory and GameFlow

In the field of psychology, Flow theory has been used to understand how humans experience enjoyment optimally from doing their tasks (Csikszentmihalyi, 1990) (Engeser, 2012). According to the theory, Flow is the state a person is in when he is able to concentrate so much, to the point of becoming "absorbed" into the activity at hand. When a video game player experiences Flow, he feels a strong emotional connection with the video game he is playing.

In video game field, Flow has been adapted into GameFlow model (Sweetser & Wyeth, 2005) that can be used to measure the quality of a video game. The model consists of eight elements:

1). Concentration
2). Challenge
3). Player skills
4). Control
5). Clear goals
6). Feedback
7). Immersion
8). Social interaction

GameFlow exists as a union between video game heuristics and the field of positive psychology as represented by Flow theory. The union is perfectly logical because video games engage their players emotionally and therefore involve many aspects of human psyche (Altizer) (Bowen) (Freeman, 2003) (Frome, 2007).

1.4. Dynamic Difficulty Adjustment

Nowadays, DDA has already become a popular method to implement the "challenge" aspect of GameFlow (Browne, Simon, Michael, Gow, & Baumgarten, 2014). The aim of DDA is to make a video game more flexible in adapting to different types of player, who have different skill levels and different tolerances to gameplay challenges. DDA itself can be implemented in an active, player-oriented manner, or in a passive one (Chen, Flow in Games, 2006). The difference is in whether players can actively influence the adjustment to the difficulty level. In a video game with active DDA, its players can choose, as a part of the gameplay, to increase or decrease the difficulty level. On the contrary, a passive DDA video game does not provide any visible choices of difficulty level during its gameplay session, and instead works in the background by monitoring changes to gameplay variables.

In relation to GameFlow model, the quality of the DDA mechanism of a video game can be measured with the "challenge" element and its criteria:

- Challenges in games must match the players's skill levels;
- Games should provide different levels of challenge for different players;
- The level of challenge should increase as the player progresses through the game and increases their skill level;
- Games should provide new challenges at an appropriate pace.

1.5. Passive Dynamic Difficulty Adjustment

Passive DDA has been gaining popularity and has been actively researched (Alexander, Oikonomou, & Sear, 2013) (Arulraj, 2010) (Hao, He, Wang, Liu, Yang, & Huang, 2010) (Sha, et al.,

2010) (Wu, Chen, He, Sun, Li, & Zhao, 2011) (Yu, et al., 2010). Passive DDA has also been implemented in commercial video games such as Left 4 Dead (Booth, Replayable Cooperative Game Design: Left 4 Dead) (Booth, The AI Systems of Left 4 Dead) and God Hand (Bycer).

As seen in Figure 1, a passive DDA mechanism is commonly implemented as a special module which works separately from main gameplay module. Because it's not an integral part of the gameplay module, the mechanism can be inserted or removed without breaking the gameplay module. Players cannot control the mechanism directly and may not even realize its existence, as there is no direct interaction between them and the mechanism's module.

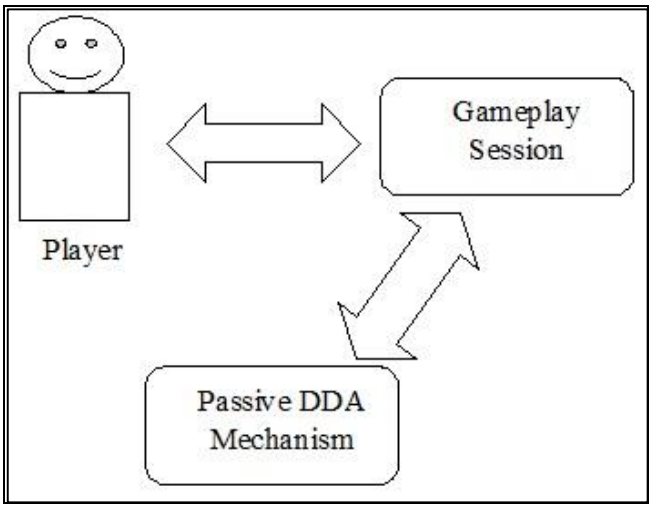

Figure1Common passive DDA implementation

In principle, there are three aspects of a passive DDA mechanism that need to be prepared (Browne, Simon, Michael, Gow, & Baumgarten, 2014):

1). Gameplay variables that reflect the performance of the player (the number of defeated enemies, how many times the player has died, etc);
2). Gameplay variables or elements that influence the difficulty level of the game (the number of enemies in a level, how often the enemies shoot or attack, etc);
3). Formulas or mappings determining the relations between the two kinds of variable (for example: the number of enemies in a level equals a constant divided by how many times the player has died).

1.6. GameMaker

GameMaker is a popular family of video game software development tools, intended primarily for easy and quick development of two dimensional video games (Habgood & Overmars, 2006). The latest in the family is GameMaker Studio, which allows developers to deploy their games onto various mobile platforms.

In order to develop video games with GameMaker, a developer needs to understand some terminologies (GameMaker, 2015). An *object* is a class of object that can be deployed as instances in the game. An object may have a *sprite*, which is a picture representation of the object. Another kind of picture is *background*, which is used in a level and not related to any objects. There are two types of sounds: *sound*, which means a sound effect, and *music*, which plays in the background. Each level in the game is represented by a *room*, which contains instances of objects. Program codes are written inside objects or as *scripts* which can be called by any objects.

2. Research Methodology

In order to find the right GDD format for passive DDA video games, this research followed these steps:

1). Literature study, which was mainly done to understand GDD and passive DDA;
2). Creating a new GDD format for video games with passive DDA mechanism, which was based on the general format by Salazar et al.;

3). Preparation for the testing of the new GDD format, which involved preparing testing methodology, participants, development process methodology and materials, evaluation criteria, and development tools;
4). Performing the testing of the new GDD format;
5). Evaluating the new GDD format, which involved revising the format if deemed necessary.

Table 1 Modification to General GDD Format

Chapter	Subchapter	Description
Chapter 1 Overview		(no change)
Chapter 2 Mechanics		(no change)
Chapter 3 Dynamics		(no change)
Chapter 4 Aesthetics		(no change)
Chapter 5 Passive DDA Mechanism	5.1 Player's Performance Determiners	This subchapter lists gameplay variables that reflect the performance of the player.
	5.2 Difficulty Level Determiners	This subchapter lists gameplay variables or elements that influence the difficulty level of the game.
	5.3 Adjustment Formulas or Mappings	This subchapter explains the formulas or mappings between the contents of preceding subchapters.
Chapter 6 Experience	6.1 Intrinsic Gameplay 6.2 Mechanic Gameplay 6.3 Interactive Gameplay 6.4 Aesthetic Gameplay 6.5 Intrapersonal Gameplay 6.6 Interpersonal Gameplay	(no change)
	6.7 Dynamic Difficulty Adjustment Gameplay	This subchapter explains, from a player's point of view, the characteristics of the passive DDA mechanism implementation. Will the player feel sudden or slow but steady changes of difficulty level? Will the changes be moderate or drastic? Etc.
Chapter 7 Constraints and Assumptions		(no change)

2.1. Creating a New GDD Format

Based on our literature study, we found that developers who want to employ passive DDA mechanisms in their video games need to design passive DDA's three important aspects. Therefore, we decided to modify the general GDD format by Salazer et al. by adding a new chapter for the aspects, and also a new subchapter. The modification can be seen in Table 1.

We inserted the design of passive DDA mechanism as Chapter 5, located after the chapter for aesthetics. The reasoning for the placement was that the two kinds of determiner in the mechanism may include gameplay variables and contents, which are designed in chapter 2, 3, and 4. Therefore we saw that it is appropriate for developers to finish the design for those three chapters before start working on the passive DDA mechanism.

We also added a new subsection in Experience chapter which explains how players will experience the passive DDA mechanism. We saw the addition as necessary because the passive DDA mechanism will contribute to a player's overall experience.

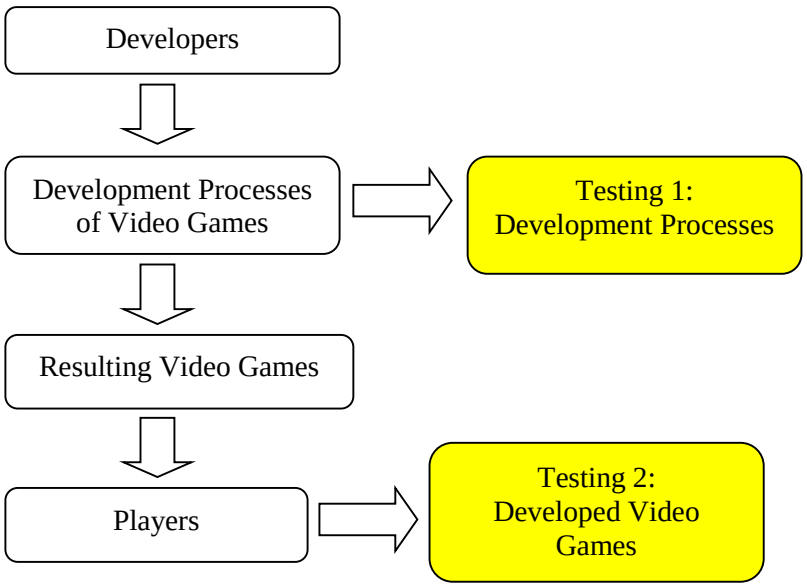

Figure 2 Testing Methodology

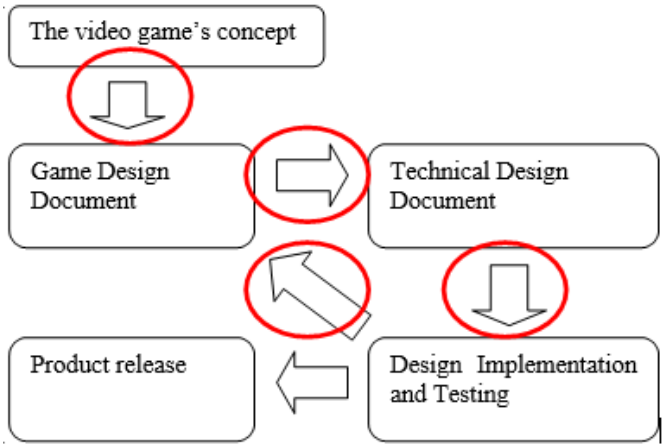

Figure 3 Modified Waterfall Methodology

2.2. Testing Methodology

Our testing methodology can be seen in Figure 2. We conducted two testing processes, one involving developers and the other involving players. Each developer started with a video game concept, which then was developed into two similar video games with passive DDA mechanisms, one using the modified GDD format and the other using the old, general one. Their development processes were observed and evaluated to know which ones were more successful and free of difficulties: the ones using the old GDD format or the ones using the modified one. On the other hand, the resulting video games were played and tested by the players to evaluate the qualities of their passive DDA mechanisms.

In each of those two testing processes, there were three possible outcomes for the modified GDD format: positive if it was proven to be better than the old format, neutral if it wasn't proven to be better or worse, and negative if it was proven to be worse. Ideally, the modified format should acquire positive

results in both testing processes. If this ideal condition couldn't be achieved, then at least the format could get one positive and one neutral result in order to be declared successful.

We see that it was important that two video games with the same basic concept but different GDD formats were developed by the same developer. This was to achieve two things:

1) Making sure the two video games were not very different, to prevent any biases from players that might arise from liking one of the games more than the other;
2) Making sure that there was no difference in developer's skill between the two games, so that any difference in quality score was caused solely by the used GDD format.

2.3. Selecting Developers

Five developers were participated in the developer testing process. Their characteristics were as follows:

1). Had never known or studied, at the start of the testing, the modified GDD format;
2). Did not interact with each other during the testing;
3). Had some experiences in developing video games, yet could not be categorized as professionals;
4). Had no prior experiences in developing video games with passive DDA mechanisms.

Tabel 2 Technical Design Document Format

Chapter	Subchapter	Description
Chapter 1 Software Outline	1.1 Video Game Software Description	Short description about the video game software to be implemented
	1.2 General Technical Specifications	A list of general technical aspects of the would-be video game software
Chapter 2 Objects	2.1 Parent and Child Patterns	Explaining the inheritance patterns between existing game objects
	2.2 Game Object Types	Detailed explanations about all game objects
Chapter 3 Sprites and Backgrounds	3.1 Sprites	Detailed explanations about all sprites, which are pictures representing game objects
	3.2 Backgrounds	Detailed explanations about all background pictures
Chapter 4 Sounds and Music	4.1 Sounds	Detailed explanations about all sound effects
	4.2 Music	Detailed explanations about all background music
Chapter 5 Rooms	5.1 General Room Specifications	Technical specifications of all existing rooms in the video game
	5.2 Room Flow	Explaining how the rooms flow from and to each other
	5.3 Room List	Detailed explanations about all rooms
Chapter 6 Scripts	6.1 Script List	Detailed explanations about all used scripts

2.4. Selecting Development Process Methodology

The developers used modified waterfall methodology to develop video games. The flow of the methodology can be seen in Figure 3. With modified waterfall methodology, the developers were allowed to go back from design implementation and testing phase to GDD. This were done to revise the content of their GDDs as needed. There are five steps, drawn as arrows in the above figure, in a modified waterfall development process. In order to know how successful the process is, the steps need to be observed and evaluated. For the purpose of this research, we did not observe the last step because the 10 video games were not released as products.

2.5. Evaluation Criteria for Developer Testing

To evaluate the development processes that were performed by the developers, we observed the aforementioned four steps, from writing to revising GDD. We also gathered the five developers' opinions on those four steps of their development processes. We gathered their opinions by presenting them with a questionnaire consisting of four questions, each related to one step in the modified waterfall methodology. The questions were:

1). Was translating the video game's concept into detailed design in GDD performed smoothly?
2). Could the design in GDD be translated into TDD smoothly?
3). Did implementation of the TDD's content run smoothly?
4). Could revisions to GDD be performed smoothly?

The questionnaire used Likert Scale, ranging from 1 for "disagree strongly" to 5 for "agree strongly".

2.6. Evaluation Criteria for Player Testing

The resulting video games were evaluated by players in terms of their passive DDA mechanisms. After playing the video games, the players were presented with a questionnaire with four questions, taken from the criteria of GameFlow's "challenge" element. As with the previous questionnaire, this one also used Likert Scale. We reworded the criteria of the GameFlow's element to communicate them easier to common players:

1). Did you feel that the difficulty level was appropriate for you, which means it was neither too easy nor too hard?
2). Did the game slowly and steadily become harder as you became better at playing it?
3). Did you feel that the difficulty level was specifically adjusted to make you feel comfortable playing the game?
4). Was the game able to deliver a continuously challenging experience without making you feel overwhelmed?

2.7. Development Tools and Technical Design Document Format

We used GameMaker 8.1 and GameMaker Studio as the development tools. Because both tools are similar to each other, there were no technical differences in the implementation processes of the video games.

Because Technical Design Document (TDD) is used to design the technical implementation of a video game software, its format depends on the used development tool. That is why the TDD format used by the developers was based on GameMaker 8.1 and GameMaker Studio. The structure of the TDD format can be seen in Table 2.

3. Results And Discussions

3.1. Developed Video Games

Five video game concepts were developed during the developer testing phase, each was developed twice using the old and the modified GDD format, resulting in two similar video games with some differences. In the end of the development processes there were 10 video game softwares in total. Four video game concepts were of vertical scrolling shooter genre and one was of maze exploration genre. The characteristics of the video games are as follows:

1). Alien Hunter

This is a vertical scrolling shooter game with an endless level. The player fights a neverending stream of enemies by shooting a machine gun with unlimited ammo or blasting limited but refillable bombs. The one developed with the modified GDD format presents five types of regular enemies, while the one with the old format presents only three but with addition of one boss enemy type. Passive DDA was implemented by monitoring player's lives, score, and the number of missed enemies, and adjusting the number of regular enemies on screen, the attack frequency of boss enemies, and the frequency of bomb refill items to match the player's skill.

2). Si Pitung in Space

The gameplay of this game is similar to Alien Hunter, except that there is no boss enemy and some regular enemies can shoot. The version of this game that was developed with the old GDD

format does not feature bombs for players, but instead the machine gun can be upgraded by taking upgrade items.

3). Soul Stealer
The gameplay of this game is also similar to Alien Hunter, except that player's bomb automatically refills itself over time and the player can gain additional lives. The version of this game that was developed with the modified GDD format gives the player an extra life after defeating a boss enemy, whereas the bonus is gained based on score in the other version.

4). Maze Evolution
This is a maze exploration game where the player explores mazes filled with traps and enemies. Its passive DDA mechanism monitors player's health, score, and how many times the player gets hit and uses the data to adjust enemies' aggressivity and attack frequencies. In the version developed with the modified GDD format, the player already have weapons but must search for bullets to shoot them, whereas it's the contrary in the other version.

5). Aurora War
This is a vertical scrolling shooter game but with four levels with definite length each. At the end of each level the player faces a strong boss enemy. The player can use one standard weapon and three special ones. The version of the game developed with the modified GDD format requires the player to gather ammunitions for his special weapons, whereas the other version requires him to gather the weapons themselves. This game's passive DDA mechanism monitors player's lives, scores, and the number of escaped regular enemies and uses the data to adjust the speed and attack frequency of enemies.

3.2. Observation of Development Processes

We observed how the development processes were performed by the developers.

1). In step 1 of development processes, the developers wrote detailed gameplay design in GDDs based on the basic concept of each video game. For the old GDD format, all developers experienced two difficulties. The first was when the developers tried to fill in the designs of passive DDA mechanisms of their games, which involved improvisations to find the places for the mechanisms. The developers experienced confusions as to how the improvisations should be done, and the confusions delayed the design processes slightly. On the other hand, there was no such difficulty when the modified format was used, because the format provides a special chapter for the designs the mechanisms.

2). In step 2, every GDD was translated into TDD. The passive DDA mechanism of every video game was implemented in TDD as a program code in one or several objects. This means that the technical design of the mechanism was written in chapter 2 of the TDD.
For GDDs with the old format, three developers experienced no problems in translating passive DDA mechanisms. The other two, however, were confused for a while about confusions about how to do the translations. Should the mechanisms be coded as a part of a single object? Or should it be split into a number of program codes as parts of various objects? The cause of this confusion was identified as the uncertainty in placing the designs of passive DDA mechanisms in GDDs. Different places for the designs influenced somewhat how the developers thought the designs should be translated in TDDs. This problem did not happen when the developers were translating GDDs with the modified format.

3). In step 3, the technical designs in TDDs were implemented into video game softwares and then tested to find flaws. There was no differences observed between implementations of video games designed with the old GDD format and those with the modified one. Both implementations were performed smoothly by the developers.

4). In step 4, the developers went back to existing GDDs to revise their contents, based on the results of testings in step 3. For GDDs with the old format, four developers experienced a problem when trying to revise the designs of passive DDA mechanisms. As the designs involved improvisations, the developers had to recall how the improvisations were done before they could revise the passive DDA mechanisms. As a result, the revision processes were slightly delayed. On the other hand, no such problem happened during the revisions of the GDDs with the modified format because the location of the designs of passive DDA mechanisms was known immediately, which was in chapter 5.

Table 2 Developers' Evaluation of the GDD Formats

Video Game	GDD Format							
	Old				Modified			
	Step 1	Step 2	Step 3	Step 4	Step 1	Step 2	Step 3	Step 4
Alien Hunter	3	4	4	4	4	4	4	4
Si Pitung in Space	3	4	4	3	4	4	4	5
Soul Stealer	3	4	4	3	5	4	4	4
Aurora War	2	3	4	3	4	5	4	5
Maze Evolution	2	3	4	3	5	5	4	5
Average	2.6	3.6	4	3.2	4.4	4.4	4	4.6
Average of All Steps	**3.35**				**4.35**			

Table 3 Evaluation of Developed Video Games

Crite-rion	GDD Format									
	Old					Modified				
	Alien Hunter	Si Pitung in Space	Soul Stealer	Aurora Wars	Maze Evolu-tion	Alien Hunter	Si Pitung in Space	Soul Stealer	Aurora Wars	Maze Evolu-tion
1	107	108	110	103	129	100	101	131	102	118
2	100	102	120	90	99	102	106	80	102	134
3	99	98	115	90	110	99	97	125	102	119
4	106	111	95	89	110	103	106	119	80	109
Average	3.43	3.49	3.67	3.1	3.73	3.37	3.42	3.79	3.22	4
Average of All Games	**3.48**					**3.56**				

3.3. Developers' Evaluation of GDD Formats

We gathered the five developer's opinions on their experiences in using the old and the modified GDD formats. The results can be seen in Table 2. The modified format was rated higher than the old one in almost all development process steps. This result is in line with the observation of how the development processes were performed as described previously.

As seen in the table, the biggest score difference happened in step 1 of the development processes. The old GDD format's score is the lowest here compared to in other steps, and the difference from the modified format's score is also quite significant. This is in line with how the development processes were performed, because the developers were indeed having difficulties in writing GDD when they used the old format.

The second lowest score when using the old format is in step 4. Most of the developers experienced difficulties in revising GDDs written in old format, although the difficulties were minor. The third lowest score for the old format is in step 2, where only two developers reported problems. We can say that the most insignificant difference between the two GDD formats is in step 2.

Meanwhile, there is no difference at all in step 3. This proves that which GDD format was used did not influence the production stage of the development processes, where the video game softwares were being implemented and tested. Which format was used influenced only the preproduction or design stage. We see that the reason for this condition was because the old and the modified format are not very dfferent from each other, because they only differ by one chapter and one subchapter. If the differences are much more significant, we assume that there would be score differences too in step 3, because how the preproduction is performed greatly influences the success of the production (Bethke, 2003) (Callele & Neufeld, 2005).

3.4. Player Testing Results

Testing by players was performed for two weeks. In total, 30 players participated anonimously. The results can be seen in Table 3.

As seen in the table, three out of five video games developed using the modified GDD format scored higher than their counterparts. However, the score differences between individual games and between all the games as a whole are quite insignificant. This shows that which GDD format was used for which video game had little influence on the quality of said game. We see that the reason for this was similar to the reason for no differences in step 3 of development processes, which is that the two GDD formats are not very different from each other.

3.5. Final Evaluation of Modified GDD Format

The testing process by the developers acquired a positive result, because the modified GDD format was able to make the development processes run more smoothly. On the other hand, the testing process by the players acquired a neutral result because there was no significant difference in score between the games made with the old format and those made with the modified one. Based on the results of those two testing processes, we deemed the modified GDD format successful because it was able to satisfy the minimal condition for its successfulness, which was one positive result and one neutral result.

We saw that the modified GDD format did not need to be revised. Chapter 5 for three aspects of a passive DDA mechanism did it job well, judging from positive ratings from the developers. On the other hand, the new subchapter in chapter 6 did not seem to be utilized much by the developers. However, we still included the subchapter in the modified GDD format because it serves a purpose and does not have any negative effects.

4. Conclusions

We have modified general GDD format by Salazar et al. to make it more suitable for developing video games with a passive DDA mechanism. We have verified its usefulness in two testing processes, one involving five developers and the other involving 30 anonymous players. In order for the modified format to be successful, at least one of the testing must get a positive result and the other got a neutral one.

The developers used the modified and the old GDD format to develop video games with passive DDA mechanisms. The development processes were based on modified waterfall methodology. The developers rated the modified format higher than the old one in three, out of four, steps of the methodology. In relation to their passive DDA mechanisms, the resulting video games were also tested by players, but there were no significant differences in quality found between those made with the modified GDD format and those with the old one. Nonetheless, because the developer testing acquired a positive result and the player testing acquired a neutral one, we see the modified format as successful.

Our suggestions for future improvements:

1). Modifying our GDD format further to make it more suited for specific gameplay types;
2). Modifying our GDD format further to also support *player modelling* method, in order to support the development of video games whose passive DDA mechanisms are supported by that method.

5. References

Alexander, J. T., Oikonomou, A., & Sear, J. (2013). An Investigation of the Effects of Game Difficulty on Player Enjoyment. *Entertainment Computing, 4* (1), 53-62.

Altizer. (n.d.). *The Emotional Impact of Videogames*. Retrieved Juni 25, 2015, from http://playstation.about.com/od/features/a/emotionalgaming.htm

Arif, I. (2014). *Penambahan Unsur Edukasi pada Dokumen Desain Permainan untuk Perbaikan Kualitas Permainan Edukasi*. Surabaya: Institut Teknologi Sepuluh November.

Arulraj, J. J. (2010). Adaptive Agent Generation using Machine Learning for Dynamic Difficulty Adjustment. *International Conference on Computer & Communication Technology (ICCCT).*

Bethke, E. (2003). *Game Development and Production.* Wordware Publishing Incorporated.

Booth, M. (n.d.). *Replayable Cooperative Game Design: Left 4 Dead.* Retrieved Mei 27, 2014, from http://www.valvesoftware.com/publications/2009/GDC2009_ReplayableCooperativeGameDesign_Left4Dead.pdf

Booth, M. (n.d.). *The AI Systems of Left 4 Dead.* Retrieved Mei 27, 2014, from http://www.valvesoftware.com/publications/2009/ai_systems_of_l4d_mike_booth.pdf

Bowen, H. (n.d.). *Can Videogames Make You Cry?* Retrieved Juni 25, 2015, from https://www.bowenresearch.com/game-informer-article.php

Browne, C., Simon, C., Michael, C., Gow, J., & Baumgarten, R. (2014). Toward the Adaptive Generation of Bespoke Game Content. In M. C. Angelides, & H. Agius (Eds.), *Handbook of Digital Games* (p. 29). New Jersey: John Wiley & Sons, Inc.

Bycer, J. (n.d.). *Examining Subjective Difficulty: How Plumbers Can Fight Demons.* Retrieved Mei 27, 2014, from http://www.gamasutra.com/view/feature/6583/examining_subjective_difficulty_.php

Callele, D., & Neufeld, E. (2005). Requirements Engineering and Creative Process in the Video Game Industry. *Proceedings of the 2005 13th IEEE International Conference on Requirements Engineering*, (pp. 240-250).

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

Chen, J. (2006). *Flow in Games.* Los Angeles: University of Southern California.

Csikszentmihalyi, M. (1990). *Flow: The Psychology of Optimal Experience.* Harper and Row.

Engeser, S. (2012). *Advances in Flow Research.* Springer.

Freeman, D. (2003). *Creating Emotion in Games: The Craft and Art of Emotioneering.* New Riders Publishing.

Frome, J. (2007). Eight Ways Videogames Generate Emotion. *Proceedings of DiGRA 2007 Conference.*

GameMaker. (2015). *GameMaker Studio Documentation*, 1.4. (Yoyo Games) Retrieved June 27, 2016, from Yoyo Games: https://docs.yoyogames.com/

Habgood, J., & Overmars, M. (2006). *The Game Maker's Apprentice.* Apress.

Hao, Y., He, S., Wang, J., Liu, X., Yang, J., & Huang, W. (2010). Dynamic Difficulty Adjustment of Game AI by MCTS for the Game Pac-Man. *Sixth International Conference on Natural Computation (ICNC 2010)*, (pp. 3918-3922).

Royce, W. W. (1970). Managing the Development of Large Software Systems. *IEEE WESCON 26* (pp. 1-9). TRW.

Salazar, M. G., Mitre, H. A., Olalde, C. L., & Sanchez, J. L. (2012). Proposal of Game Design Document from Software Engineering Requirements Perspective. *The 17th International Conference on Computer Games.*

Sha, L., He, S., Wang, J., Yang, J., Gao, Y., Zhang, Y., et al. (2010). Creating Appropriate Challenge Level Game Opponent by the Use of Dynamic Difficulty Adjustment. *Sixth International Conference on Natural Computation (ICNC 2010)*, (pp. 3897-3901).

Sweetser, P., & Wyeth, P. (2005). GameFlow: A Model for Evaluating Player Enjoyment in Games. *ACM Computers in Entertainment*.