SHADER TWEAKING TOOL FOR ARTIST-PROGRAMMER COLLABORATION IN GAME DEVELOPMENT

Bayu Munajat[1], Jimmy Tirtawangsa[2], Angelina Prima Kurniati[3]

[1]Magister Teknik Informatika, Fakultas Teknik Informatika, Universitas Telkom

Abstrak
Shader tweaking process is a polishing phase in game development. It becomes a real problem in terms of collaboration and time for artists and programmers. There are many tools related to shader authoring and tweaking activities, but they don't accommodate both roles to get optimized shaders' values. On every optimizingactivity, the game source code has to be recompiled, and its compile time overhead consumes significant time compare to programmer and artist activity time. Traditionally, there are many such iterations before some optimized shader values are achieved. The use of tool-based approach is to assist both artists and programmers improving the shader tweaking process. It reduces the number of iteration during polishing phase in game development and the amount of spent time. However, factors like shader knowledge and collaboration experience affect the use of the shader tweaking tool. The tool provides single iteration polishing phase and visual interface for the artists to tweak shader script's parameters. It also reduces time for thinking and tweaking process. Shader tweaking process time is reduced by 50.36% for people already familiar with the tool and by 43.42% for people with no experience. By storing the values of associated variables of shader in the collaboration tool, programmers can adjust the shader of game in a short time.

Kata Kunci : shader, tweaking, tool, programmer, artist, game

Abstract
Proses penyesuaianshaderadalahsebuahfasepemolesandalampengembangan game. Inimenjadisebuahmasalahnyatadalamhalkolaborasidanwaktuuntuk para artisdanpemrogram.Ada banyakalatterkaitaktivitaspembuatandanpenyesuaianshader, tetapitidakmenyediakankeduaperanuntukmendapatnilai-nilai optimum shader.Padasetiapaktivitasoptimasi, kodesumberdari game harusdikompilasiulang, dankelebihanwaktukompilasinyamemakanwaktu yang signifikandibandingkandenganwaktuaktifitasdaripemrogramdanartis.Secaratradisional, terdapatbanyakiterasisepertiitusebelumnilai optimum shaderdiperoleh. Pendekatanpenggunaanberbasisalatadalahuntukmembantu para artisdanpemrogrammemperbaiki proses penyesuaianshader. Hal inimengurangijumlahiterasiselamafasepemolesandalampengembangan game dansejumlahwaktu yang terpakai.Namunfaktorsepertipengetahuanshaderdanpengalamankolaborasimempengaruhipenggunaanalatpenyesuaianshader.Alatmenyediakaniterasitunggalfasepemolesandanantarmuka visual bagiartisuntukmelakukanpenyesuaian parameter skripdarishader.Alatjugamereduksiwaktuuntuk proses berpikirdanpenyesuaian. Waktu proses penyesuaianshadertereduksisebesar 50.36% untuk orang yang sudahtidakasingdenganalatdansebesar 43.42% untuk orang tanpapengalaman. Denganmenyimpannilai yang bersesuaiandarishaderdalamalatkolaborasi, para pemrogramdapatmengaturshaderdari game dalamwaktulebihsingkat.

Keywords : shader, penyesuaian, alat, pemrogram, artis, game

CHAPTER 1

THE PROBLEM

1.1 Rationale

Indie (independent) game developers around the world [1] have been encouraged to reach as many as possible potential markets by competitive game development [2] [3]. Polishing the games becomes a way how to enhance gamers' experience [4] as well as shader for immersive visual effect optimization. Shader is an algorithm written explicitly to run on a computer Graphics Processing Unit [5] which enables extend rendering capability above and beyond what fixed-function pipeline can do.

Shader as part of game polishing phase is usually developed in last iterations of game development [6] and need collaboration from developers with different roles i.e. artists and programmers [7]. Since shader can only be observed visually when the game is running, developers have to rerun the game many times. Suppose that they know the shader doesn't work properly, they stop the game, change the scripts and other resource parameters, recompile the game, and rerun the game to see the results [8]. Tweaking is an activity to adjust value of variables inside the script or code so that shader will work properly. They may have to repeat the process until the shader is working properly. The number of iterations becomes uncertain if the shader result is not acceptable. So the shaders may not be included into the games by the end of the schedule if theirs don't work properly. Since the shaders tweaking process is in some last iterations of the development, the game might have included all resources. This condition causes longer compile time and run time. The process is too much time-consuming because shaders tweaking must be also in runtime gameplay and might need particular events to simultaneously visually identify shaders.

The current conditions of shader tweaking process are difficult [9] and slow [6]. Different background knowledge and skill between both roles make them difficult to understand each other and also need more time. In indie game studio, most artists are not technical artists as described in [10], they have most aesthetics education background [11] but they don't have basic game programming and technical qualification. In current process, artists use combination of values and programmers use trial-and-error method to get appropriate shader in their game. Verbal communication between them takes a lot of time and many activities during shader tweaking are repeated. Different domain collaboration is reported and handled by attuning the artist role through intimate iteration or tool development with appropriate artistic meaning [9]. COSTART reported that cross-domain collaboration work which involves art elements needs both technology-supported and well defined roles in collaboration [12]. In MuiCSer approach, multi-disciplinary project in one integrated process shall provide appropriate roles and communication among them [13].

In developing countries like Indonesia, the growth of game industry is mostly by local independent developers [14] [15] which have different characteristics than existing foreign game companies [16] [17]. The differences between local and foreign studios are number of developers, scalability, and game development process. Besides high turnover of developers, local indie studios are startup companies where their developers are usually fresh graduates. This means developers are divided into artists and programmers without specific roles such as technical artists [10]. Local studios have different scalability in development [18]. In game development process, indie studios have smaller development team, shorter time to release their games, and different development approach for each studio [19]. There aren't technical artists or other roles which can bridge when the need of collaboration between artist-programmer is emerged in local indie studios. There isn't collaboration process model which is practical enough for them when they face complex interaction of artists-programmers.

There are many existing tools for shader authoring and self-tweaking process. ASHLI [20] is a tool with more artist involvement which helps artist authoring new effect and adjusting value of shaders. It is a pixel shader compiler and encapsulates the output which means it still needs to relink and load the game. There are similar artist-friendly tools, RenderMonkey [21] and NVidia FX Composer [22] where they are professional tools from world class video-graphics-card vendors. Both of them provide many features for authoring shaders and robustness for 'AAA' game development. These tools provide shader functions and templates for 3D games and resources meanwhile they become unfit and too exaggeration in the need of variables adjustment in tweaking activity.

Microsoft Visual Studio [23] is one of game development environment with embedded HLSL (High Level Shader Language) authoring tool which help programmers compose the shaders and track the relations within. It provides effect template for 3D games and don't have tweaking capability. So, programmers must recompile the game each time they tweak the shader. Unity3D [24] is an integrated visual development environment which is especially dedicated for 3D game development. It helps programmers create shader visually and put it where it should be appeared in a game. Programmers usually have difficulty to develop 2D game in 3D development environment because it doesn't fit. It also has shader authoring embedded-tools so programmers can statically tweak shader and must recompile after they tweak the shaders.

There are also several current community-based tools which might help shader tweaking process such as Shazzam [25] and WPFFX [26] with instant tweaking for artist. Shazzam tool helps shader tweaking because it provides script editor and is artist-friendly. The idea behind these tools is to help artist and programmer collaboration for shader tweaking in a game. Both tools produce HLSL script although not in game development format. It may cause different effects to appear in the running game. Programmers will need extra effort for integration their output is not intended for game development but for Xaml (eXtensible Application Markup Language)-based application development. EffectArchitect [27] offers robustness and direct experiencing for on the spot feedback, but it is limited to associate the custom parameters to be used for further various tweaking process. It still needed HLSL scripting skill to create custom variables and adjust the values.

All above tools don't fit with important points that are needed in tweaking collaboration. They are store mechanism for actual values of related-variables and translation through visually adjustable related-variable. Some of them don't have synchronous viewer to get direct feedback after adjusting related-values. Somehow, the use of tools will improve the shader tweaking process as well as encourage both artist and programmer collaboration roles. Shaders that is developed for polishing purpose should be a way for indie game studio to boost the user experiences of their games, but the real problems come up as time-spent in shader tweaking process from uncertain iterations.

1.2 Theoretical Framework

Since understanding the interrelation between the backgrounds to the phenomenon this research should be conducted to raise the idea that might leverage the proposed problems scientifically. In this sub-chapter, the required steps to conceptualize this research will be discussed. There are three theory areas to be concerned in this research: shader development process baseline, using tool for shader tweaking process, and shader tweaking experiment.

1.2.1 Shader Development Process Baseline

Shader development process is one of polishing process after completing the gameplay. This process involves artist and programmer roles to make a shader work well. Baseline is the simplest possible of shader tweaking process. This baseline is used as basis measurement to be compared to proposed solution. Shader development process will be described in figure 1.1 below.

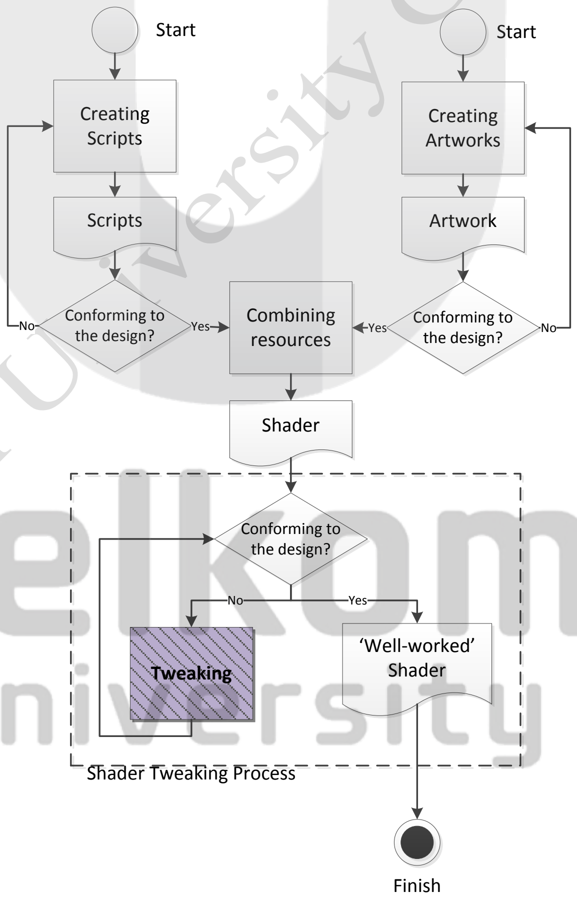

Figure 1.1 Shader Development Process

The process starts from an artist creating the artwork that will be used in shader and a programmer codes the script of shader. Both artwork and script must conform to the design. Then the programmer combines the artworks and the scripts so they appear together. The shader tweaking process involves both roles to get appropriate values for each determined parameter. Process uses trial-and-error methods and so it is hard to estimate number of iterations and total tweaking time.

1.2.2 Using Tool for Shader Tweaking Process

Using tool means that there is a tool that is used for shader tweaking process. The tool is developed as part of this research for artist and programmer to improve the baseline. The use of the tool will be observed during experiment to find out what happen in shader tweaking and its involving collaboration roles and relation with the improvement of shader identification, analyzing, and adjustment to eliminate defects. Besides, solution should improve information exchange in their communication since their skill is not interchangeable.

1.2.3 Shader Tweaking Experiment

In order to make sure this research show the impact of implemented idea enable to improve the shader tweaking process, there are series of experiment to investigate the use of this proposed tool. Hereafter the experiment should able to observe how the behavior can be presented directly, precisely, and systematically, besides it also controls in organized situation which simulate the appearance in real world including personals who are involved, when and where it would be executed, make an experiment is possible to be done. The experiment aims the result of the research might have better shader tweaking process in game development especially the process with artist-programmer collaboration.

1.3 Conceptual Framework

Conceptual framework on how the use of the tool will fit in the game development.

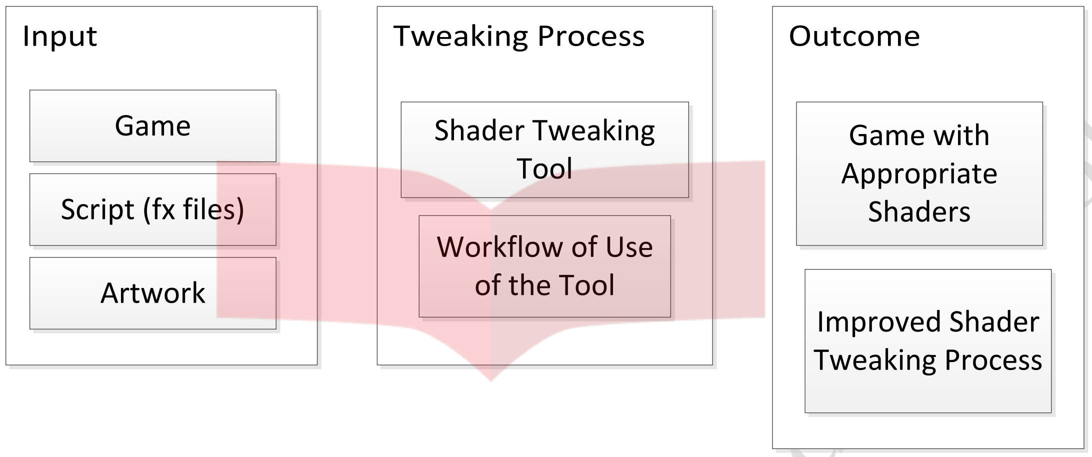
Input
Game
Script (fx files)
Artwork
Tweaking Process
Shader Tweaking Tool
Workflow of Use of the Tool
Outcome
Game with Appropriate Shaders
Improved Shader Tweaking Process

Figure 1.2 Conceptual Framework

1.3.1 Inputs

Inputs are game with complete gameplay source and usable features. This condition is required because shader is non-gameplay aspect and any gameplay bugs or defects will confuse the developers when they tweak the shaders. Important elements of the inputs that are fx files conformed to the design and artworks.

1.3.2 Tweaking Process

The tweaking process is the activity of both artist and programmer. In the baseline system the process includes from starting to recompile the game up to retrieving the final values for the related parameters. The tweaking process involves the use of the shader tweaking tool and the workflow of using of the tool.

Hereafter the research should able to observe how the behavior can be presented directly, precisely, and systematically. Besides it also controls in organized situation which simulate the appearance in real world including personals who are involved, when and where it would be executed, make an experiment is possible to be done. It is decided in this research to figure out scientifically what happens in shader tweaking process and its involving collaboration roles. The main goal of the research is to show the impact of implemented idea is able to improve the shader tweaking process. The solution should improve information exchange in their communication since their skill is not interchangeable; improve shader identification, analyzing, and adjustment to eliminate defects. The experiment aims the result of the research might have better shader tweaking process in game development especially in the process with artist-programmer collaboration.

There are variables related to the proposed problems above. First, it is the knowledge of artist and programmer in shader concept and experience which will affect their ability in shader tweaking. Second, it is the experience of artist and programmer in collaboration game development which will affect their interactions in artist-programmer pairs. Third, it is the existence of collaborative shader tweaking tool which affect the tweaking process if compared to manual tweaking (without any tweaking tool). All above mentioned variables, all of them can be controlled during experiments. There are variables which are needed to give responses about what happen and how something happen during experiments. Number of iterations and spent time will be recorded and measured as responses to find out the relations among controllable variables.

1.3.3 Outcome

The outcome will be the game from input with adjusted shaders which the parameters for appropriate condition when the game is played. The most important is that there is significant improvement in the shader tweaking process when using tool during game development than not using it which is measured in the mentioned variables. Improved shaders and its tweaking process also can be reflected in time spent and iterations from various experiences of developers and different conditions of tweaking process.

1.4 Statement of the Problem

There are identified problems in the baseline system according to previous explanation:

1. Shader tweaking process has collaboration problems between different roles. They are communication problem, difference in skill and knowledge, and difference in shading measurement. They influence time spent for tweaking the values of shader.
2. Current shader tweaking process has uncertain number of iterations which will affect the overall project's schedule and timeline.
3. Current shader tweaking process has to recompile the whole game as part of tweaking activity. Recompiling a complete game takes a significant amount of time.

1.5 Hypothesis

From the statement of the problem, there are problems that shall be solved in this research. The objective of this research shall be a collaboration system between roles which should improve shader tweaking process. They are collaboration of different roles and integrating tweaking process so they can achieve appropriate shader to be included into the game. Shader collaborative tool is designed and developed not to remove one of role but to encourage both roles so that the tweaking process can be improved. The use of shader collaborative tool should reduce the time and looped-activities that is needed to tweak the shaders.

1.6 Assumptions

The motivation behind the hypothesis is that the tool will provide dual interface with their semantic for each role that meant to enhance the information transfer between them but not to eradicate all verbal communication of them so the shader tweaking will be faster. It also isn't obstructed redundant communication for understanding each intention or to discuss inessential matter in tweaking process anymore. Both roles are expected to be able to analyze the shader faster and less iteration with new attempted value. The process should be direct and concurrent. They are also expected capable to capture the changes in shaders since it provides runtime situation which means no need time consuming for recompiling and stopping the game to make change.

1.7 Scope and Delimitation

The scope and delimitation describe what criteria in-or-out of boundary for the use of shader tweaking tool will work properly.

1. The games must compatible with HLSL (High-level Shader Lnaguage).
2. The programmer at least has capability in procedural programming.
3. The artist at least has knowledge how shader work in game, especially pixel shader.
4. There must defined input state as initial condition of effects and final state as stopping point of tweaking.
5. The tools run at least in Windows 7 OS with minimum .Net 3.5 Runtime Framework and Xna 3.1 Distributable.
6. The artwork may be Png, Jpg, Jpeg, or Bmp.
7. There is internet connection for collaboration.
8. The developers' computer must have VGA with minimum support for pixel shader 2.0.

1.8 Importance of study

1. The benefit of this research including the tool will be gained for any indie studio as long as the game is based on DirectX and its families.
2. The process of shader development, include the manner of collaboration and communication between different roles, might be improved even without the tool.
3. The tool is simple to operate for indie game artist without many button or shortcut such as other 3D tools.
4. The indie game will get the benefit for inexpensive tool that appropriate for small team and short time slot for shader development.
5. Concept of synchronous data storing mechanism could be implemented in many aspect for process improvement in game development.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

The collaboration problem between roles as artists and programmers has been handled by the use of collaborative shader tweaking tool. The tool translates adjustable parameters within a script created by the programmer to visually adjustable values that changeable by the artist. The programmers and the artists get instant feedback from runtime viewer, such that defects in the shader can be fixed immediately. However, the use of this tool has no exact same-time collaboration because both the artist and the programmer work on same fx files on the same workstation.

Only single iteration is needed for their tweaking process, so the process is more predictable. For expert participants, there are 45 iterations as differences of 65 total iterations which mean it is 69.23% reduction of iterations. While non-expert participants, there are 139 iterations as differences of 163 total iterations which mean it is 85.28% reduction of iterations. The proposed system by using tool has greater impact in number of iterations for non-expert developers.

The use of this tool also provides significant time reduction. Experiment data shows consistently time reduction in the differences between baseline and the using tool. It is by 50.36% time reduction for experience pairs and by 43.42% time reduction for inexperience pairs. The experiment sequence order has no effect on observable time reduction. There are time reductions using proposed system for either baseline first (41.14% time reduction) or using tool first (47.822% time reduction) when the experiment was being conducted.

The use of this tool also cuts the constant factor of recompile time for the whole game. Although the programmers and the artists still have to compile the whole game resources for final evaluation, but recompiling much time has been removed from the shader tweaking process on average we can save as much as 315.2 seconds (about 5 minutes) by not recompiling during shader tweakgin process using this tool.

There are effects of improvement regarding the whole game development phases. First, the proposed concept of using the tool could help a studio arranges the schedules with more definitive estimation since shader development can be fast and in only single iteration. Second, from the time and iterations differences between expert and non-expert developers, we can see that the learning curve effect of using the tool is very short. More skillful the developers and more frequent using the tool and always following the workflow, less time and less iteration they will take to develop their next games.

5.2 Recommendations

Related to the tool–based approach, there are still many domains in game development that can be improved using such a tool. For example fonts, inverse kinematics, atlas sprite management, and various deployment configurations can be created much faster if such tools are available. The research orientation can also be directed to more process improvement in shader development. For example can be constructing new or improving current shader development process such as to 3D environment, more particles integration, deep into pixel shader, using geometry shader for 2D environment, and shader compatibility tools for various device hardwares.

Bibliography

[1] D. Ltd Pty, "IndieDB," 2011. [Online]. Available: www.indiedb.com.

[2] Wayne, "The Indie Revolution: How little games are making big money," 2013. [Online]. Available: http://www.gameacademy.com/blog/the-indie-revolution/.

[3] M. J. Irwin, "Indie game developers rise up," 10 2013. [Online]. Available: http://www.forbes.com/2008/11/20/games-indie-developers-tech-ebiz-cx_mji_1120indiegames.html.

[4] S. Swink, Game Feel: A Game Designer's Guide to Virtual Sensation, Burlington, MA, USA: Morgan Kaufmann, 2008.

[5] W. Ritscher, HLSL and Pixel Shader for Xaml Developers, O'Reilly, 2011.

[6] C. Keith, Agile Game Development with Scrum, Addison-Wesley, 2010.

[7] S. Theodore, "Who among us shall build this shader?," Gamasutra, 6 3 2009. [Online]. Available: http://www.gamasutra.com/view/feature/132354/who_among_us_shall_build_this_.php. [Accessed 2013].

[8] S. St-Lauren, Shaders for Game Programmers and Artists, Boston: Thomson/Course Technology, 2004.

[9] G. Turner, A. Weakley, Y. Zhang and E. Edmonds, "A Social and Technical Study of Artist-Programmer Collaborations," *17th Workshop of Psycology of Programming Interest Group,* vol. 17, pp. 106-119, 2005.

[10] B. Cloward, "Joining the Dark Side," Game Development Conference, 2012.

[11] G. Turner and E. Edmonds, "Towards Supportive Technological Environment for Digital Art," Autralia, 2004.

[12] L. Candy, "COSTART: an investigation of art and technology collaboration," University of Technology, Sydney, Sydney, Australia, 2003.

[13] M. Haesen, K. Coninx, J. Van den Bergh and K. Luyten, "MuiCSer a Multi-disciplinary User-centered Software engineering process to increase the overall user experience," in *Engineering Interactive Systems*, 2008.

[14] A. Widhiyasa, "Agate Studio - About Indonesia Game Industry - SlideShare," 31 May 2012. [Online]. Available: http://www.slideshare.net/Clawford/about-indonesia-game-industry-agate-studio.

[15] D. Wong, "Altermyth TGS 2013 Indonesia Game Market & Industry," 18 10 2013. [Online].

[16] S. E. Smileworks, "Square Enix Smileworks," Square Enix Smileworks, 2013. [Online].

[17] Gameloft, "Gameloft Store," Gameloft, 2012. [Online]. Available: http://www.gameloft.co.id/.

[18] D. Yuliani, "Games in Asia," 12 7 2013. [Online]. Available: http://www.gamesinasia.com/11-startups-watch-indonesias-gaming-industry/.

[19] Kummara, "Segitiga.Net," Segitiga.Net, 2012. [Online]. Available: http://segitiga.net/tag/game-developer-bandung.

[20] A. J. Preetham and A. Bleiweiss, "Ashli – Advanced Shading Language Interface," ACM SIGGRAPH ATI Research, 2003.

[21] AMD, "RenderMonkey," 2013. [Online]. Available: http://developer.amd.com/tools-and-sdks/archive/archived-tools/gpu-tools-archive/rendermonkey-toolsuite.

[22] NVidia, "NVidia Fx Composer," 2012. [Online]. Available: https://developer.nvidia.com/fx-composer.

[23] Microsoft, "Microsoft Visual Studio," 2012. [Online]. Available: http://msdn.microsoft.com/en-us/library/hh873117.aspx.

[24] Unity3D, "Unity3D," Unity3D, 2013. [Online]. Available: http://unity3d.com/.

[25] S. Crew, "Shazzam Tool," Shazzam, 10 2009. [Online]. Available: shazzam-tool.com.

[26] Jaimer, "WPFFX," 1 11 2007. [Online]. Available: wpffx.codeplex.com. [Accessed 2012].

[27] Default_Ex, "FX Architect," 13 4 2009. [Online]. Available: fxarchitect.codeplex.com. [Accessed 2013].

[28] L. Antii, "Collaborative game development environment for educational purposes," 2013. [Online]. Available: https://www.theseus.fi/handle/10024/64404.

[29] Parker and Felan, "Indie Game Studies Year Eleven," in *Authors & Digital Games Research Association DiGRA*, Toronto, 2013.

[30] R. Graebsch, "The Indie Game," 12 2012. [Online]. Available: http://romangraebsch.de/.

[31] L. Hunt, "Predictive and Adaptive Game Development: A practical application of development models to the independent video game industry," Edith Cowan University, Perth, Australia, 2011.

[32] A. Hajden and I. G. Juricic , "Elements of Indie Game Development," 4 6 2013. [Online]. Available: http://bitserum.com/design_development/video-game-development-process/.

[33] D. Yu, "Make Games," 31 10 2011. [Online]. Available: http://makegames.tumblr.com/post/12182911828/on-polish-spelunky.

[34] Z. P. Juhara, Pemrograman Game dengan DirectX, Yogyakarta: Andi, 2011.

[35] A. Reed, Learning XNA 4.0, O'Reilly, 2010.

[36] Microsoft, "Reference for HLSL," Microsoft, 12 12 2013. [Online].

[37] R. Grootjans, XNA 3.0 Game Programming Recipes: A Problem-Solution Approach, Apress, 2010.

[38] B. Munajat, "Report of Shader Tweaking Tool Experiment in Indie Game Studio," Bandung, 2013.

[39] A. Fernandez, B. Garzaldeen, I. Grutzner and J. Munch, "Guided Support for Collaborative Modeling, Enactment and Simulation of software Development Process," *Software Process Improvement and Practices,* 2004.

[40] D. Sefton, "Real-time Collaboration for Games Development," 20 6 2011. [Online]. Available: http://www.altdev.co/2011/06/20/real-time-collaboration-for-games-development/.

[41] S. L. Pfleeger, "Experimental design and analysis in software engineering," *Annals of Software Engineering,* vol. 1, pp. 219-253, 1995.

[42] I. Parberry, Max B. Kazemzadeh and Timothy Roden, "The Art and Science of Game Programming," *SIGCSE,* 2006.

[43] M. Q. Tran and R. Biddle, "At Ethnographic Study of Collaboration in Game Development Team".

[44] C. Coleman, K. Harris, A. Hinton and A. Ephanov, "Automated Shader Generation using Shader Infrastructure," in *I/ITSEC*, 2007.

[45] S. Theodore, "Who among us shall build this shader?," Gamasutra, 6 3 2009. [Online].

Available:
http://www.gamasutra.com/view/feature/132354/who_among_us_shall_build_this_.php.