ATLANTIS PRESS

Advances in Social Science, Education and Humanities Research, volume 134

2nd International Conference on Innovative Research Across Disciplines (ICIRAD 2017)

An Alternative Rendering Solution in Animated Movie Making for Final Year Students: A Case Study

I Made Putrama[1], I Gede Mahendra Darmawiguna[2], Gede Saindra Santyadiputra[3]
[1, 2, 3]Informatics Engineering Education Department
Universitas Pendidikan Ganesha
made.putrama@undiksha.ac.id[1], mahendra.darmawiguna@undiksha.ac.id[2], gsaindras@undiksha.ac.id[3]

***Abstract*— The rendering stage in creating 3D animated movies using Blender is a phase with time-consuming process. The more complex the animation, the longer it will take. The same obstacle experienced by final year students of Informatics Engineering Department, Undiksha, who mostly use one computer at a time for rendering. This inhibits the animation making process and causes the progress of their final project takes too long. This problem can be reduced by utilizing Blender network rendering to help complete the process become faster. Unfortunately, the generated output by this feature has to be processed further to obtain the desired result and not much flexibility available to extend it for customization. This paper discusses a proposed alternative by integrating a grid-computing framework that can be programmed and customized as needed. With this approach, the students also will be able to retrieve the results right away without doing some further work on the output.**

Keywords—network rendering, grid computing, Blender 3D, animation

I. INTRODUCTION

Decision to raise a topic related to 3D animated movie making in a final year project becomes a challenge for students especially in Informatics Engineering Department (PTI), Undiksha. In general, 3D movies which have duration for about 15-20 minutes require a very long work even for months so the project that ideally should be completed within one semester generally could only be completed within for two semesters or more. In addition to the external factors of the students themselves are the creativity and skills of students in using the software as well as the level of complexity of animated movies that will be made are parts of the common obstacles faced by students. However, the support of existing facilities specifically for animated movie making becomes an equally important factor in providing support to complete the final project in timely manner. A survey conducted on five students who were doing animated film development at the time argued that the minimum specification required for a PC should be at least with i-7 processor and a minimum of 16 GB RAM. On the Blender official website, the best production specification environment to use Blender software is a PC with 64-bit eight core CPU, 16 GB RAM and Dual OpenGL 3.2 Graphics Card with 4 GB RAM[1]. Unfortunately, PC specifications available at LCI Laboratory in PTI, Undiksha are as follows: Processor Intel Core i7-3770 CPU @3.4 GHz with 6 GB RAM.

During the 3D animated movies making with Blender, the most time consuming stage is in the rendering stage. Rendering is a process of converting data from objects seen in Camera View in Blender 3D into image files or animated movie files. Where setting the desired image or movie output format, whether it is the quality or the setting of the number of frames per second, will affect the length of time required to complete the whole process [2]. With the facilities, the PTI department has and using only one PC at a time, to produce an animated movie with 300 frames, the PTI students need about 6-12 hours for the rendering alone. Some of them even need up to 3 days when they render 980 frames.

The Blender 3D actually provides a feature to do rendering on a network. It works by using a computer as a server/master, one another as a client and the rest as a slave. However, the configuration of this feature is somewhat complicated when used such as the started service of the master and slaves often get disconnected and needs to be re-enabled every time a new file is loaded. The rendering result format is also different with the selected one during the input file creation that is in an EXR format (a dynamic range raster format developed by Industrial Light and Magic). The format is suitable for purposes such as further digital compositing, but in most cases, the PTI students need the output to be in the selected format as in the Blender during the file creation such as PNG or AVI format.

Therefore, to address the limitation, the authors proposed an alternative solution by utilizing a programmable grid-computing framework by adopting some existing solutions and exposing them to the following sections. Section 1 is an Introduction, Section 2 discusses some case studies and approaches conducted in several related studies, Section 3 discusses the used method, Section 4 deals with the design and implementation results in prototype form and Section 5 includes the conclusions.

II. RELATED WORKS

Application of network-based rendering in 3D animated movie making has been done in many cases. Even cloud-based grid rendering services are also easy to find nowadays. However, there are several reasons why these solutions are not feasible to be used especially by students in the department of PTI, Undiksha. First, online services require adequate internet facilities. In Undiksha, this alternative would be expensive for the students because the internet for everyday needs is still insufficient. Secondly, cloud-based rendering service is mostly an average of commercial services so that there is a relatively expensive price for the students so that the use of this alternative is still considered as a less economical choice. Third, the topic of student thesis is sometimes a topic with

original themes whose expectations can be further developed by the students concerned. An online rendering service would require that animated movie material be uploaded first. This allows copyright abuse of animated products uploaded online so this alternative is also hardly considered among students at PTI. Related research has also been done by Bui [3] which in the research, the author designs a distributed rendering system with algorithm made based on Python library. Bui points out that the constraint in the design is the challenge to specify parameters in order to reduce stagnant conditions when a computer inside the cluster fails to complete the working part. Ginty in his research has compared several alternative of distributed rendering mechanisms by using supporting software such as Autodesk, Yadra with Blender, Cinema4D and NewtTek Lightwave which has managed to provide efficiency to the rendering process that previously took 70 to 90 hours to only about 30 minutes [4]. Patoli in his research has successfully used an open source alternative that allows rendering in the network that is done by utilizing Condor and Blender 3D software. It is just that by using a Command-based Condor, rendering results still exist in the slave computer. New program implementation is required by using Python to transfer all rendering results to the source computer. In addition, the job monitoring system used does not come from the framework that can be utilized directly, but must be developed separately [5]. Sheharyar has designed a rendering farm system that works really well by adding software scheduler with firstly comparing some of available options. It is just that in the prototype tested, the author has chosen to use Cube and Open FBS, in which the Cube framework does not support 3D Blender software. So the presented rendering test results is produced from using 3D animation software Autodesk 3D Maya which is a commercial software [6]. Another study was conducted by Xiong who uses JPPF framework as a basic technique for retrieving data from MS SQL Server databases. Additionally in his study, Xiong implements query processing using a range partitioning scheme which uses several different tablespaces to store data in a given table [7].

In this paper, authors proposed an alternative rendering system design that is adopting the existing solutions which is specifically for rendering animations built using the Blender 3D application. In addition, the authors will discuss the way in the design so as to provide an increased time efficiency to the rendering process when compared to using the network rendering provided directly by Blender 3D.

III. METHOD

In the design of the system in the form of prototype, the authors follow the framework of Software Development Life Cycle (SDLC) approach with prototyping model. According to Agarwal, this model is well used to develop systems with technical solutions that are not yet known clearly at the beginning. In addition, this model is suitable for use when it is not possible to make ready-to-use products in one manufacturing cycle so that a starting product is made and then discarded to produce a more perfect product thereafter [8]. Stages in this model consist of the Quick Plan, Quick Design Modeling, Construction of Prototype, Deployment Delivery & Feedback and Communication. The whole stages are done continuously until the product prototype is obtained as expected. In the planning and design stage, the authors explore several open source software frameworks to be able to make the system prototype faster.

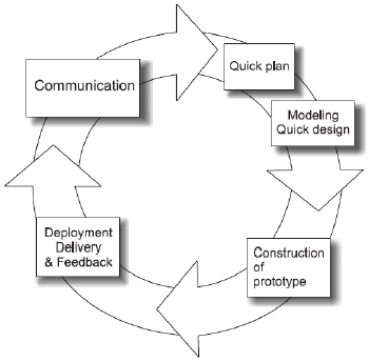

Figure 1 Prototyping Model

To examine the case study problems undertaken at the LCI Lab, a case study method as elaborated by Yin in his book entitled “Case Study Research: Design and Methods” was followed and modified as depicted in Figure 2. During the design and development stage of the model, separate rendering using 3D Blender was prepared. On the other hand, a new proposed rendering system was built. Each system was tested using the same input files to generate outputs. Both outputs were then compared, analyzed further to obtain a conclusion report.

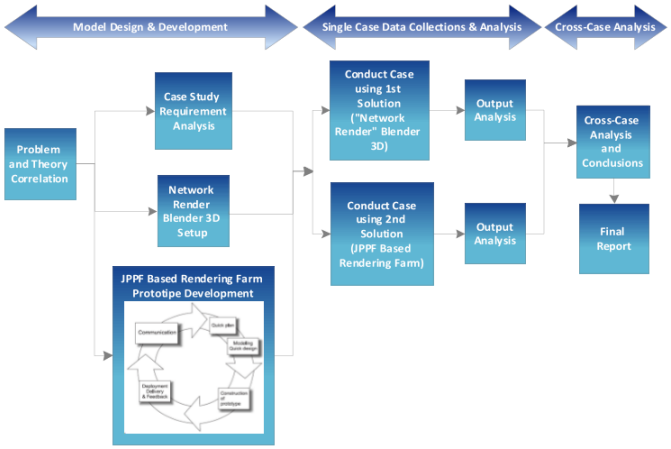

Figure 2 Research Method

To build the system prototype, the Grid Computing framework, JPPF that was used by Xiong in his research [7] is adopted. The framework architecture is a driver / node topology that simply consists of three components i.e. client, driver and node components. The illustration is as shown on Figure 3.

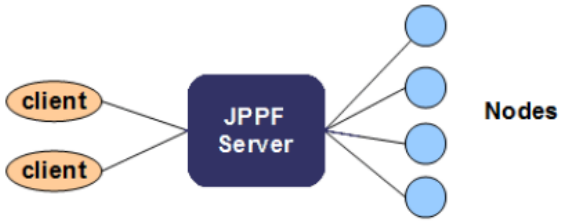

Figure 3 Architecture Topology of the JPPF Framework [9]

A LAN network is required to connect some of the regular computers to be used within this system. A computer on the Client side is used to start rendering the job by uploading a 3D Blender file to a computer that acts as a Driver. Next, the Driver will set the scheduling to divide the work into smaller units called Tasks. A 3D Blender file generally consists of several Scenes where each Scene consists of several Frames that make up an animated file into a single Blender file. The Driver divides the total Frames that must be rendered based on the total Nodes connected proportionately according to the exclusion algorithm that has been configured in JPPF. Furthermore, each workmanship by each node is reassembled by the Driver and forwarded to the desired output folder on the Client computer. To combine output from multiple Nodes, the author uses the FFMPEG open-source software assistance [9]. Furthermore, the author also utilizes the technique of load distribution based on in-memory data grid by using Hazelcast [10] especially when transmitting input and output files from Client side to Node side, and vice versa.

IV. Results

The overall system is a set of programs combined as an application that is the result of integration of JPPF, Hazelcast and FFMpeg framework as in Program Details.

TABLE I. PROGRAM DETAILS

No	Program Name	Description
1	JPPF Driver Application [9]	This set of software is readily available from the JPPF framework that must be configured and used right away. It is executed on one of any computer in the network to act as the Driver
2	JPPF Node Application [9]	This set of software is readily available from the JPPF framework that must be configured and run on each of the computer that we use as the nodes. The node would connect to the Driver to execute a given task(s).
3	JPPF GUI Application [9]	This set of software is readily available from the JPPF framework that must be configured and run on any one computer that we use to monitor the grid system.
4	Client Application	This is the developed application in the form of prototype to be used to send the rendering job to the Driver.

The system architecture is depicted in Figure 4.

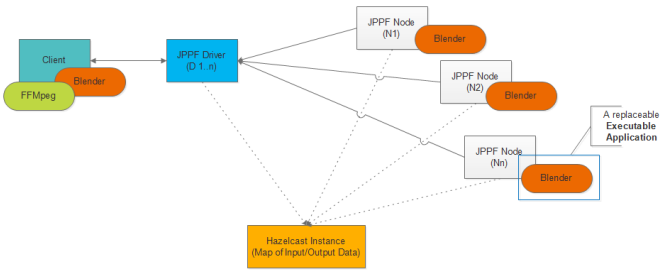

Figure 4 System Architecture

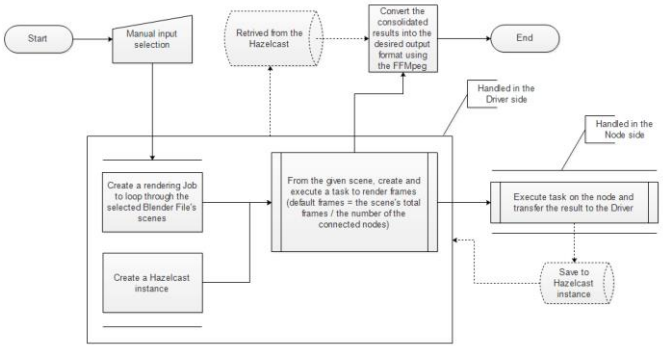

Figure 5 the algorithm flowchart of the new system

The main benefit of using this architecture is the flexibility of the overall supporting frameworks that are programmable and based on the same Java programming language so it can be ported in any platform without the need to redevelop the solution. Each of the programs can be configured and run as a daemon or service in the Operating System that is used. This will give the opportunity to use the computer for other purposes while it is helping in the rendering process. Another benefit is the use of the Hazelcast's in-memory data grid technology that allows the input as well as the chunk of the output files are stored in the grid so it is accessible directly by a connected computer used as the client for the final consolidation. The design also integrates the FFMpeg to convert the final output as the desired audio/video format such as AVI/MPEG [10]. Furthermore, the JPPF framework that is used would allow an external application such as Blender to be attached and called as a task forwarded to the nodes. It can actually be replaced to use other application to make up a new solution for other use cases. As shown in Figure 5 the Hazelcast and the FFMpeg are another frameworks which help in storing results and converting them into the desired movie format.

The client application of the new system is depicted in Figure 6. The monitoring interface of the Blender 3D is depcited in Figure 7. For the new system, the JPPF framework that is used has been accompanied with a comprehensif user interface for the grid monitoring as depicted in Figure 8.

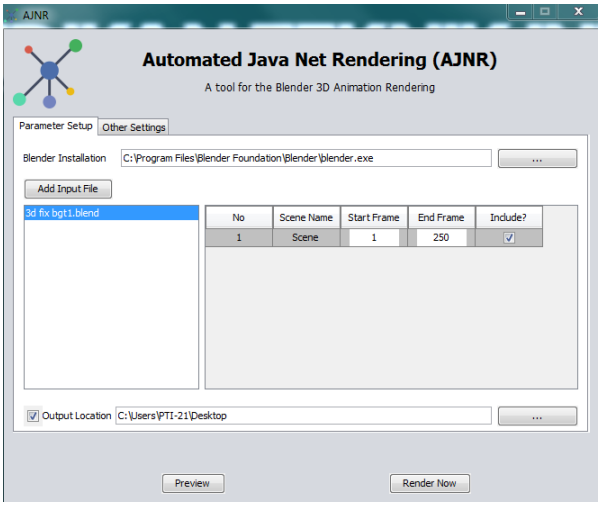

Figure 6 the new system client application

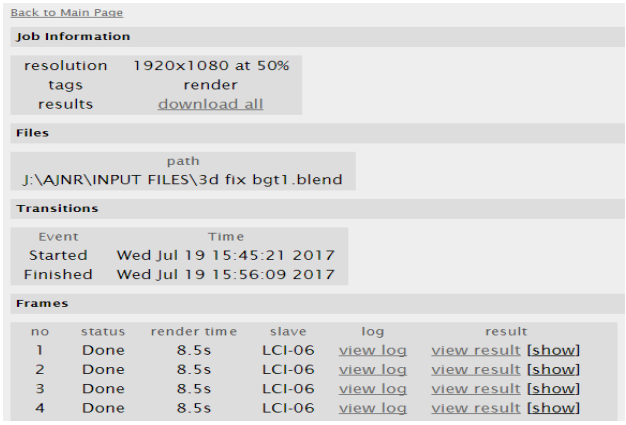

Figure 7 the Blender 3D Network Render Monitoring Interface

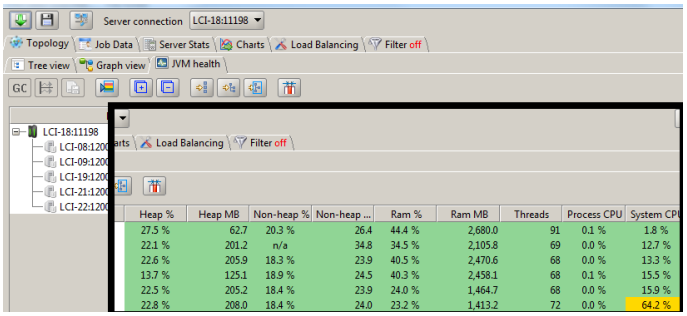

Figure 8 JPPF Grid monitoring interface (JPPF GUI)

To perform a rendering using the new system, the following steps need to be done: (1) *Configure the JPPF Driver and Nodes.* There are mainly three sets of program that comes from the JPPF frameworks and need to be configured before the rendering activity can be started. They are JPPF Driver, Node and GUI programs. In the Driver configuration file, one important aspect that we can select the algorithm used by the framework to distribute the workload to the nodes. This later can be done during runtime on the JPPF Grid GUI as well. In the Node as well as the GUI programs, we need to set computer Driver. Once it is done, ensure to start the Driver beforehand followed by the Nodes and the GUI if we want to monitor the job through the interface. The rendering can be started through the Client application which can be reside on any computer of the network.

For a preliminary result, the authors conduct some observation by rendering the same Blender 3D file in a network with five computers using both of the Blender 3D network-rendering feature and the prototype system as in

.

In term of time efficiency, results of both systems show that, the Blender 3D excels better than the new system except for the Low Poly Animation. However, the output files still reside in the slave computers and its format is in the EXR format although the selected output file format for the used blender input file was in AVI-Raw format. While the result from the new system converted as the desired output which is AVI format in this case.

TABLE II. PRELIMINARY TESTING RESULTS

Input File Info	No. of the testing cycles	No. of computers on the network	Blender Network Render		The Prototype System	
			The average elapsed Time (in minutes)	Result Format	The average elapsed Time (in minutes)	Result Format
(file 01) 250 frames, High Textured Animation, AVI output	5	5	10.8	EXR	11.2	AVI
(file 02) 1336 frames, Sketching Animation, AVI output	5	5	2.37	EXR	3.15	AVI
(file 03) 140 frames, Low poly Animation, AVI output	5	5	2.45	EXR	1.13	AVI

The longer spent time in the testing scenario when the new system was used is due to the additional round trip impossed during the output file transferred from the node to the clinet computer. As opposed, the Blender 3D rendering

version does not transfer the output file from the slaves to the master at all unless a manual download is performed thereafter. As seen, there is a metadata logging information if access from the monitoring interface as shown in Figure 7 which tells that the saved output file by the Blender 3D rendering version are still in the respective slave computers. There would be also another spent time if we download the file from the download link provided. But, instead of done automatically, in the Blender 3D version, it must be downloaded manually. Therefore, a bit longer time taken when using the new system is still acceptable as the convertion process are done automatically and the result is delivered as expected.

The following are some sample of the generated output file from both systems.

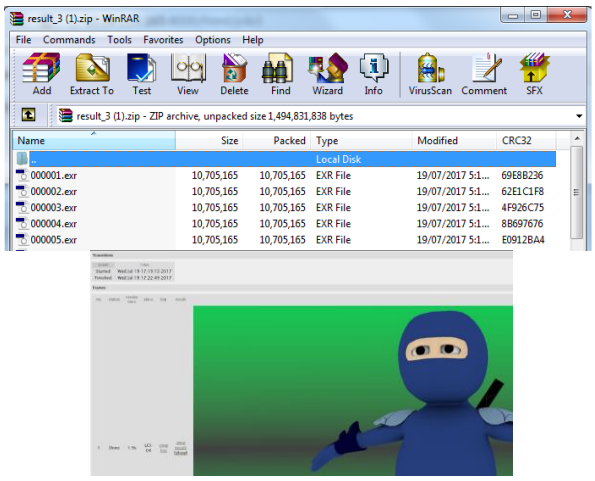

Figure 9 Blender Network Rendering' result in EXR format

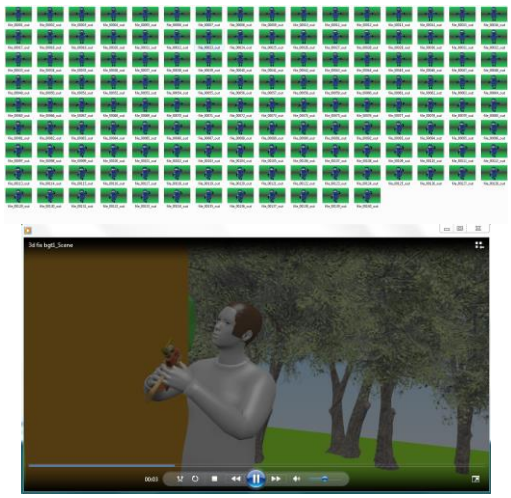

Figure 10 new system results in PNG and AVI formats

In term of the framework stability, the JPPF integration with the in-memory data grid framework, Hazelcast, has been observed that the connection between the Nodes and the Driver are getting lost and suddenly disconnected after some input file processing. And if the input file is considerably large, the Driver would face a timeout issues and need to be restarted.

Apart from that, in term of the ease of use functionality, the new system with the framework collaboration is a good approach to achieve better result for PTI students case requirement if compared to the Blender 3D feature. In order to provide the similar information and setting as what can be done in Blender 3D, the new system needs to access and integrate more Blender API into it.

In future this can be enhanced to follow the same mechanism as done in the Blender 3D whereby the round trip can be minimized by allowing the node to transfer the result files once the whole job is considered done and the client to perform the convertion thereafter. But this means that the algorithm should utilize the frameworks capability in detecting whether the nodes could potentially not sending the results at all. Another consideration is the distributed processing mechanism. The current approach is using the blender installation instances which must be available in the computer nodes as well as client and they are called as external processes during the system run. This makes another complication during the system setup although it is done once on the first time. Moreover, the input file downloading is required on each of the nodes side before the actual rendering could be done. This can be enhanced to perform the computation without the needs of transfering data between the Driver and the Nodes computers which shall overall save more time.

V. CONCLUSION

The new proposed system is considered promising to be developed further as an alternative solution for the distributed network rendering into a final working product. Some benefits can be drawn from the solutions. Such as, the flexibility to customize the solution for e.g. it could be extended as a volunteer grid computing application that allow anyone to connect and contribute during the rendering process as well as adding some features for enhancing the usability such as giving notification of the render progress to the student through email. The use of the prototyping methodologies instead of the traditional method for e.g. the *waterfall* model has proven to be very effective to address the system shortfall during the prototype system development. The grid framework used for this case as compared to the available Blender feature is considered more stable although a further analysis is needed to fix and enhance the connection issues. Another challenge is to analyze further a similar grid-computing framework for the in-memory data grid such as Apache Ignite and exploring more on the algorithm to minimize the time required to transfer the result file from the node to the client before considering building the working product.

ACKNOWLEDGMENT

Thanks to the PTI laboratory assistant and students who did volunteer works in helping us fixing the network related configuration during the system testing.

REFERENCES

[1] Blender.org, "Requirements — blender.org." [Online]. Available: https://www.blender.org/download/requirements/. [Accessed: 02-Aug-2017].

[2] J. M. Blain, *The Complete Guide to Blender Graphics*. CRC Press, 2013.

[3] P. Bui, T. Boettcher, N. Jaeger, and J. Westphal, "Using clusters in undergraduate research: Distributed animation rendering, photo processing, and image transcoding," *Proc. - IEEE Int. Conf. Clust. Comput. ICCC*, 2013.

[4] K. Ginty, J. Tindle, and S. Tindle, "Rendering 3D Computer Graphics on a Parallel Computer," *20th Int. Conf. Syst. Eng. ICSE 2009*, 2009.

[5] M. Z. Patoli, M. Gkion, A. Al-Barakati, W. Zhang, P. Newbury, and M. White, "An open source grid based render farm for Blender 3D," *2009 IEEE/PES Power Syst. Conf. Expo. PSCE 2009*, no. March, 2009.

[6] A. Sheharyar and O. Bouhali, "A Framework for Creating a Distributed Rendering Environment on the Compute Clusters," vol. 4, no. 6, pp. 117–123, 2014.

[7] J. Xiong, J. Wang, and J. Xu, "Research of distributed parallel information retrieval based on JPPF," *Proc. - 2010 Int. Conf. Inf. Sci. Manag. Eng. ISME 2010*, vol. 1, no. August, pp. 109–111, 2010.

[8] B. . Argarwal, S. . Tayal, and M. Gupta, *Software Engineering & Testing (An Introduction)*. Jones AndBartlett Publishers, 2010.

[9] JPPF.org, "JPPF Framework." [Online]. Available: http://jppf.org/. [Accessed: 19-Jul-2017].

[10] FFmpeg.org, "FFmpeg." [Online]. Available: http://ffmpeg.org/. [Accessed:19-Jul-2017].