RENDERING ANIMASI PADA LINGKUNGAN KOMPUTASI KINERJA TINGGI

Yogi Nugraha[1], Fitriyani[2], Nurul Ikhsan[3]

[123]Program Studi Ilmu Komputasi, Telkom University, Bandung

[1]nugraha.yogi99@gmail.com, [2]fitriyani.y@gmail.com, [3]nurul.ikhsan@yahoo.co.id

Abstrak

Render adalah proses perubahan sebuah *scene* objek yang berbentuk *vektor* menjadi *pixel* untuk ditampilkan menjadi *file* gambar atau *video*. Untuk me*render* objek yang memiliki kompleksitas yang tinggi akan memerlukan waktu proses yang lama. Oleh karena itu, *render* dapat dilakukan secara paralel untuk mempersingkat waktu proses *rendering*. Proses *rendering* secara paralel dilakukan pada lingkungan komputasi kinerja tinggi, yaitu pada *cluster* dan *Graphics Processing Unit* (GPU). Pada *cluster* terdapat banyak komputer yang terhubung pada jaringan yang dapat mengerjakan proses *render* secara bersamaan sehingga waktu prosesnya lebih singkat, dan pada GPU proses *render* dilakukan pada *framebuffer* GPU yang memiliki kecepatan lebih tinggi dibandingkan *default buffer* pada RAM yang dapat mempercepat proses render.

Pengujian pada *network render* dilakukan dengan blender menggunakan *file* blender *dolphin* dengan 1 dan 10 objek untuk di*render* dengan jumlah *frame* kelipatan 100 dari 100 hingga 1000 dan kelipatan 1000 dari 1000 hingga 10000. Hasil menunjukkan waktu *render* meningkat sesuai dengan bertambahnya jumlah *frame*, dan apabila render dilakukan dengan lebih banyak prosesor, maka waktu prosesnya membutuhkan waktu yang lebih singkat.

Pengujian GPU dilakukan dengan menggunakan OpenGL pada objek *dolphin* yang sama yang sudah diekspor menjadi *file* .obj yang didalamnya terdapat koordinat *vertex* dan *polygon* untuk kemudian dibaca dan di*render* menjadi sebuah objek dolphin. Hasil menunjukkan pada OpenGL GPU waktu proses *render* lebih singkat dan objek hasil *render*nya terlihat lebih halus dan memiliki proporsi warna yang lebih baik dibandingkan pada OpenGL standar, karena pada OpenGL GPU terdapat proses tambahan yaitu *shading* dan *texturing*.

Kata Kunci : *Render*, *cluster*, paralel, blender, OpenGL.

Abstract

Render is a transforming process from an object scene which is a vector into pixel so it can displayed as image or video files. To render object with high complexity, it will take a long time to process. Because of that, parallel rendering will reduce process time. Parallel rendering has been done on high performance computating environment, which is cluster and Graphics Processing Unit (*GPU). Cluster has many computers that connected each other in a network and will do the render process at the same time and make the process quicker, and on GPU, rendering process done on GPU's framebuffer that has better speed than RAM's default buffer and can increase processing speed.*

Experiment on network render with blender using dolphin blender file with 1 object and 10 object to be rendered with multiple of 100 from 100 to 1000 frame and multiple of 1000 from 1000 to 10000 frame. Experiment result showed that rendering time increase as frame increasing, and if render done with much more processor, then the process will be done in shorter time.

Experiment on GPU using OpenGL with the same dolphin object that has been exported to .obj file which contains vertexes and polygons coordinates to be read and rendered into a dolphin object. Experiment result showed that on OpenGL GPU render process time reduced and produced better color proportion and smoother object than standard OpenGL because in OpenGL GPU there are additional process, which are shading and texturing.

Keywords : Render, cluster, parallel, blender, OpenGL.

1. Pendahuluan

Perkembangan teknologi komputer grafis sangat pesat dengan ruang lingkup yang luas. Penggunaannya beragam, dari mulai untuk industri *entertainment* sampai akademis. Pada bidang industri *entertainment*, teknologi komputer grafis ini merupakan sesuatu yang penting dan harus diimplementasikan. Contoh pengaplikasian teknologi ini yang paling

kentara adalah *video* dan animasi. *Video* dan animasi terdiri dari gambar-gambar atau objek-objek diam yang disusun dan dimunculkan secara berurutan sehingga terlihat bergerak. Setiap gambar, atau *frame*, pada sebuah animasi harus berbaur dengan baik agar setiap gerakan terlihat halus. Pada film animasi 3D, terdapat gambar-gambar objek 3D yang sudah dibuat sebelumnya, lalu diurutkan dan kemudian digabung yang akan menghasilkan sebuah animasi 3D. Proses penggabungan ini merupakan proses *rendering*. Proses *rendering* sebuah film animasi akan memakan waktu yang sangat lama karena untuk membuat sebuah film animasi diperlukan ribuan *frame*. Waktu proses *rendering* sebuah *frame* pada animator profesional bisa sampai beberapa jam[2].

Pada penelitian ini penulis mencoba untuk mengimplementasikan proses *rendering* secara paralel pada lingkungan komputasi kinerja tinggi dengan menggunakan cluster dan GPU. Pada *cluster* akan dijlanakan proses *render* menggunakan blender, sedangkan pada GPU akan dijalankan proses *render* dengan OpenGL.

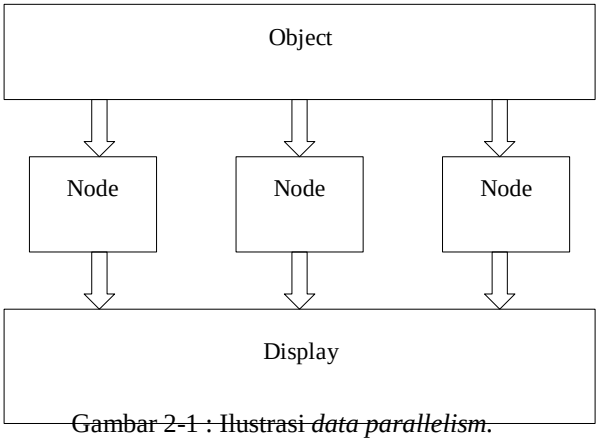
2. Tinjauan Pustaka

2.1 *Rendering*

Rendering adalah proses perubahan deskripsi abstrak sebuah *scene* menjadi sebuah gambar. *Scene* adalah kumpulan objek geometrikal dalam sebuah *object space* tiga dimensi, yang berhubungan dengan pencahayaan dan sudut pandang[3]. Proses rendering melalui beberapa tahapan. Proses pertama adalah *modelling transformation*, yaitu proses penempatan dan perubahan objek pada *object space*. Selanjutnya adalah proses *lighting*, yaitu proses penambahan cahaya agar objek terlihat lebih nyata yang memperhitungkan arah cahaya dan bayangan yang dihasilkan. Kemudian *viewing transformation*, yaitu proses pemetaan koordinat sudut pandang *object space* ke layar monitor. Selanjutnya proses *clipping*, yaitu proses pembuangan objek yang berada di luar sudut pandang *user*. Objek yang dibuang tidak akan diproses lebih lanjut. Proses terakhir adalah *rasterization*, yaitu proses konversi objek vektor menjadi gambar yang tersusun dari *pixel*.

2.2 *Parallel Rendering*

Untuk me*render* sebuah objek yang berkualitas tinggi, berdimensi besar, atau memiliki kompleksitas *scene* yang tinggi, dan membutuhkan waktu yang lama[3]. Maka dari itu, proses *rendering* ini diparalelkan agar proses *rendering* dapat dibagi-bagi ke *node-node* yang ada untuk diproses secara bersamaan.



Gambar 2-1 : Ilustrasi *data parallelism*.

Data parallelism adalah tipe *parallel rendering* yang membagi sebuah *scene* menjadi beberapa *stream* yang kemudian diproses secara bersamaan, yang kemudian akan digabungkan menjadi sebuah gambar. Konsep dasar dari *data parallelism* adalah *divide and conquer*[6]. Tipe ini memiliki dua cara dalam membagi *scene*, yaitu dengan memecah *scene* menjadi beberapa objek tunggal (*object decomposition*) dan dengan membagi *scene* menjadi potongan-potongan gambar (*image decomposition*).

2.2.1 *Object Decomposition* dan *Image Decomposition*

Object decomposition adalah proses memisahkan *data set* menjadi berbagai macam objek tunggal. *Image decomposition* adalah proses membagi-bagi *data set* menjadi beberapa potongan *data set*.

Gambar 2-2 menunjukkan bagaimana *data set* dapat dipisah berdasarkan dua proses ini. Pada *object decomposition*, setiap *workstation* akan me*render* satu objek tunggal dari *data set*. Pada *image decomposition*, setiap *workstation* me*render* setengah bagian *data set*.

2.2.2 Algoritma *Sort-First* dan *Sort-Last*

Algoritma *sort-first* adalah tahap preprocessing pada *parallel rendering* yang merupakan bagian dari *image decomposition*. Pada tahap ini, proses *sorting* dan *mapping* akan dilakukan sebelum *rendering* dilakukan. Jadi sebelum objek-objek yang terpisah

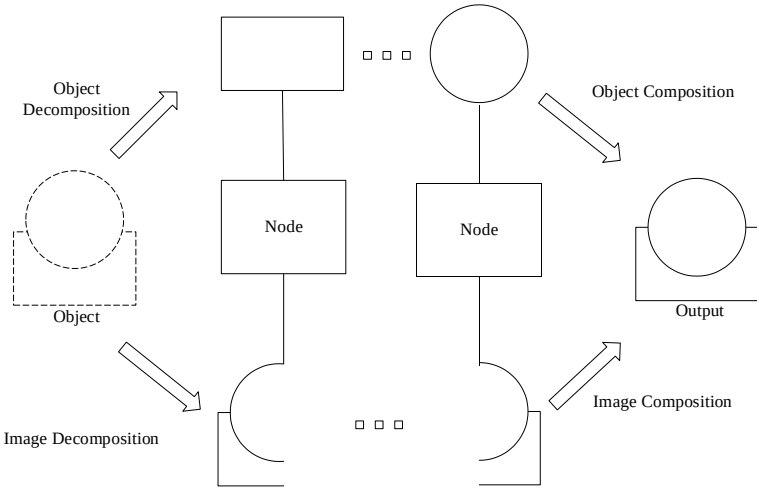
digabungkan, koordinat objek-objek tersebut untuk ditampilkan pada perangkat penampil sudah ditentukan sebelumnya.

Setiap *node* akan diberikan potongan-potongan *data set* untuk di*render*. Pertama, setiap *node* akan memeriksa primitif setiap potongan *data set* dan mengklasifikasi sesuai dengan posisi pada perangkat penampil. Pada saat ini akan ditentukan pada *node* mana primitif ini akan diproses. Ketika pengklasifikasian, *node-node* ini akan mendistribusikan ulang primitif sedemikian rupa sehingga semua *node* menerima semua primitif yang sesuai dengan posisi pada layar. Hasil distribusi ulang ini kemudian membentuk distribusi awal untuk *frame* selanjutnya.

Sedangkan algoritma *sort-last* adalah algoritma yang merupakan bagian dari *object decomposition*, dan merupakan tahap *post-processing* pada *parallel rendering*. Berbeda dengan *sort-first*, algoritma ini akan melakukan *sorting* dan *mapping* setelah proses *rendering* dilakukan. Setiap objek akan didistribusikan ke setiap *node*. *Node-node* ini akan melakukan seluruh proses rendering dan menghasilkan gambar *full-area* yang tidak lengkap dengan mentransformasi dan meraterisasi bagian dari primitifnya. Gambar-gambar sebagian ini kemudian dikomposisi, yang proses komposisinya memerlukan informasi *pixel* (warna dan kedalaman warna) dari setiap *node* yang dikirim ke *composition server.*

2.3 *Cluster*

Cluster adalah kumpulan komputer yang terhubung satu sama lain untuk menjalankan proses komputasi secara paralel pada suatu jaringan. *Cluster* terdiri dari satu *head* atau *master node* dan beberapa *node*. Dengan membagi *workload* ke setiap *node*, maka waktu proses pengerjaan akan lebih cepat. Manajemen dan perangkat lunak *programming* diperlukan untuk membagi



Gambar 2-2 : Proses *parallel rendering* dengan *object decomposition* dan *image decomposition.*

tugas menjadi bagian-bagian kecil yang kemudian membagikan bagian-bagian tersebut ke setiap *node*. *Node-node* tersebut akan mengerjakan tugas yang diberikan sebelumnya yang kemudian hasilnya akan dikirimkan kembali ke *head node.*

3. Rancangan Sistem

Pada bagian ini akan dijelaskan garis besar rancangan penelitian yang terdiri dari 3 proses utama.

a. *Input*

Input untuk *render cluster* adalah *file* blender *dolphin* yang terdiri dari 1 objek dan 10 objek. Sedangkan untuk OpenGL adalah *file* .obj yang berisi koordinat *vertex* dan *polygon dolphin.*

b. *Rendering*

Pada tahap ini objek di*render* yang dilakukan secara paralel pada cluster dengan menggunakan *plugin network render* dari blender dan pada OpenGL standar dan OpenGL GPU.

c. *Output*

Hasil dari render cluster ini adalah *file video* hasil render objek dan waktu *render* untuk dianalisis, sedangkan hasil dari OpenGL adalah file *executable* yang menampilkan objek *dolphin* dan waktu *render* untuk dianalisis.

3.1 Implementasi

3.1.1 Rencana Pengujian

Pengujian akan dilakukan dengan jumlah *frame* yang berbeda. Ini dimaksudkan untuk menganalisis sejauh mana pengaruh

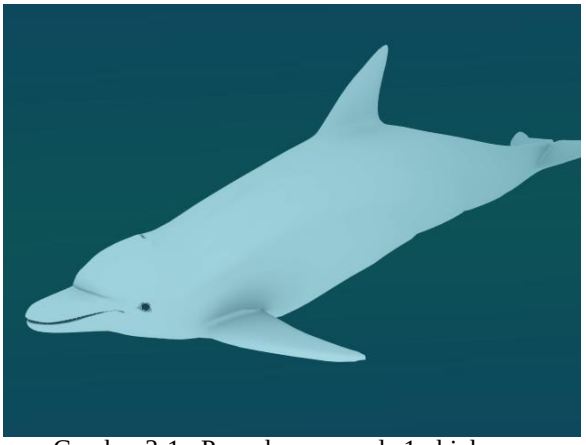
jumlah *frame* terhadap waktu proses *rendering.* Jumlah objek yang di*render* adalah 1 dan 10 objek. Alasan 10 objek yang dipilih adalah penulis ingin membandingkan sejauh mana perbedaan waktu proses *render* 1 objek dengan objek yang memenuhi layar, dan karena penulis tidak ingin merubah ukuran objek untuk konsistensi, maka jumlah yang didapat adalah 10 objek. Setelah *render* dilakukan, maka akan didapatkan waktu render yang kemudian diekstrapolasi dengan menggunakan persamaan linier sehingga didapatkan persamaan untuk memprediksi waktu *render* terhadap jumlah *frame* dan rasio *speedup* berdasarkan jumlah *thread*.

3.1.2 Objek Pengujian

Objek dolphin didapat dari https://github.com/WasserX/dist-2011-2/blob/master/test_files/blender/dolphin.blend. Pengujian dilakukan dengan dengan dua skenario, yaitu dengan 1 objek dan dengan 10 objek. Kedua objek ini memiliki kompleksitas yang berbeda. Berikut penjelasannya.

1) *Polygon* dan *Vertex*
   Polygon adalah bagian-bagian permukaan objek yang membentuk permukaan secara keseluruhan. *Vertex* adalah titik pertemuan setiap *polygon*. Semakin banyak jumlah *polygon* dan *vertex* maka semakin halus permukaan objek tersebut. Pada skenario 1 proses *render* hanya akan memproses *vertex* dan *polygon* dari 1 objek yang berjumlah 6464 *vertex* dan 6900 *polygon*, sedangkan pada skenario 2 seluruh *vertex* dan *polygon* dari 10 objek akan diproses, yang berjumlah 10 kali lebih banyak, yaitu 64640 *vertex* dan 69000 *polygon*.

2) Pencahayaan
   Pada skenario 1 seluruh permukaan objek menerima pencahayaan dengan jumlah yang sama, dan karena hanya ada 1 objek, maka tidak ada bayangan yang dihasilkan dari terhalangnya pencahayaan oleh objek lain.



Gambar 3-1 : Pencahayaan pada 1 objek.

Skenario 2 terdiri dari banyak objek maka pencahayaan akan terhalangi oleh objek lain dan akan menghasilkan bayangan. Bayangan ini akan membuat objek terlihat lebih nyata. Pada skenario ini yang diproses adalah objek, arah pencahayaan, dan posisi dimana bayangan akan dihasilkan dan muncul pada permukaan objek lain.

Gambar 3-2 : Pencahayaan pada 10 objek.

3) Animasi
   Objek *dolphin* ini memiliki gerakan lumba-lumba yang sedang berenang, dan terjadi berulang-ulang (*loop*). Tidak ada perbedaan animasi pada kedua skenario, namun pada skenario kedua jumlah objek yang bergerak lebih banyak.

4) *Rasterization*
   Pada proses ini skenario 2 memiliki lebih banyak objek untuk dikonversi dari vektor ke *pixel* untuk ditampilkan ke tampilan *video*.

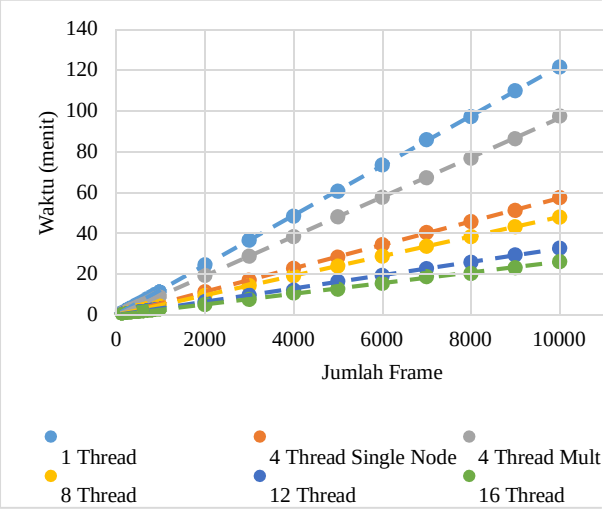
3.1.3 Konfigurasi *Parallel Rendering* pada Cluster

Proses *parallel rendering* dilakukan dengan menggunakan *plugin network render* yang sudah ter*install* pada blender. *Plugin* ini memungkinkan untuk melakukan proses rendering pada sebuah jaringan. Konfigurasi ini menggunakan 1 *node* untuk *master*, 1 *node* untuk *client*, dan sisanya untuk *slave*. Interaksi utama *user* berada pada *node client*.*Master* akan broadcast IP *address*nya dalam suatu jaringan, kemudian *client* dan *slave* akan menangkap *address* yang di*broadcast* tadi dan terhubung. Setelah itu *client* akan mengirim job ke *master*, lalu *master* akan membagi *frame-frame* ke setiap *slave*. Setelah *slave* selesai merender *frame-frame* tersebut, hasil *render* akan dikirimkan ke *master*, yang langsung diteruskan ke *client* untuk disusun menjadi sebuah *file video*.

4. Hasil Pengujian dan Analisis

4.1 *Render Cluster*

Pengujian *parallel rendering* pada cluster dilakukan dengan me*render* 100-1000 dan 1000-10000 *frame* untuk 1 objek dan 10 objek. Pengujian dilakukan pada *single-node* (1 *thread* dan 4 *thread*) dan *multi-node* (4 *thread*, 8 *thread*, 12 *thread*, dan 16 *thread*), dengan menggunakan *setting render default.*



Gambar 4-1 : Grafik waktu *render cluster* untuk 1 objek *dolphin* berdasarkan perubahan jumlah *frame* dan jumlah *thread*

Gambar 4-1 menunjukkan dengan bertambahnya jumlah thread, waktu proses *render* menjadi lebih singkat. Berdasarkan grafik di atas bisa didapatkan persamaan linier untuk memprediksi waktu *render* apabila jumlah *frame* ditambah menjadi lebih banyak. Berikut adalah persamaan linier yang dapat digunakan untuk memprediksi waktu *render* 1 objek *dolphin*.

Untuk 1 *thread* :

$$y = 0.0122x + 0.028 \quad \ldots\ldots (1)$$

dengan nilai $R^2 = 0.999956$

Untuk 4 *thread single node*:

$$y = 0.0057x - 0.1293 \quad \ldots\ldots (2)$$

dengan nilai $R^2 = 0.999864$

Untuk 4 *thread multi node* :

$$y = 0.0097x - 0.1861 \quad \ldots\ldots (3)$$

dengan nilai $R^2 = 0.999886$

Untuk 8 *thread* :

$$y = 0.0048x - 0.0758 \quad \ldots\ldots (4)$$

dengan nilai $R^2 = 0.999998$

Untuk 12 *thread* :

$$y = 0.0032x - 0.0673 \quad \ldots\ldots (5)$$

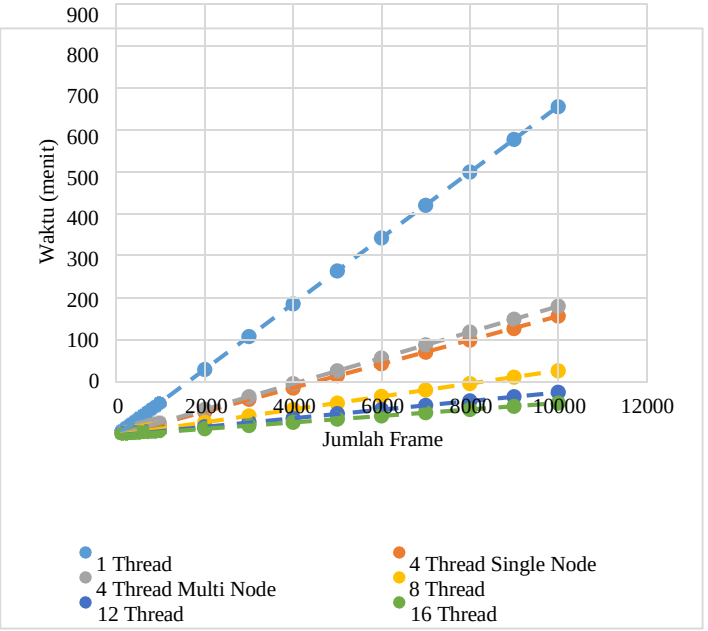
dengan nilai $R^2 = 0.999184$

Untuk 16 *thread* :

$$y = 0.0026x - 0.0501 \quad \ldots\ldots (6)$$

dengan nilai $R^2 = 0.999423$

dimana x adalah jumlah *frame* dan y adalah waktu *render* dalam menit.



Gambar 4-2 : Grafik waktu *render cluster* untuk 10 objek *dolphin* berdasarkan perubahan jumlah *frame* dan jumlah *thread*

Pada gambar 4-2, jika dibandingkan dengan gambar 4-1 memiliki perbedaan waktu proses *render* yang signifikan. Ini dikarenakan dengan menambahkan jumlah objek menjadi 10, kompleksitas tentu

bertambah, sehingga membutuhkan waktu proses yang lebih lama. Pada hal ini, proses *lighting* akan lebih banyak dilakukan karena dengan bertambahnya objek, maka perhitungan cahaya dan bayangan bertambah, dan pada proses *rasterization*, proses konversi vektor menjadi *pixel* menjadi bertambah banyak. Persamaan linier yang dapat digunakan untuk memprediksi waktu render 10 objek adalah sebagai berikut.

Untuk 1 *thread* :

$$y = 0.0782x + 1.1749 \quad \ldots\ldots (7)$$

dengan nilai $R^2 = 0.999998$

Untuk 4 *thread single node*:

$$y = 0.0284x - 0.0813 \quad \ldots\ldots (8)$$

dengan nilai $R^2 = 0.999984$

Untuk 4 *thread multi node* :

$$y = 0.0308x + 0.147 \quad \ldots\ldots (9)$$

dengan nilai $R^2 = 0.999994$

Untuk 8 *thread* :

$$y = 0.0154x + 0.4553 \quad \ldots\ldots (10)$$

dengan nilai $R^2 = 0.999967$

Untuk 12 *thread* :

$$y = 0.0103x + 0.183 \quad \ldots\ldots (11)$$

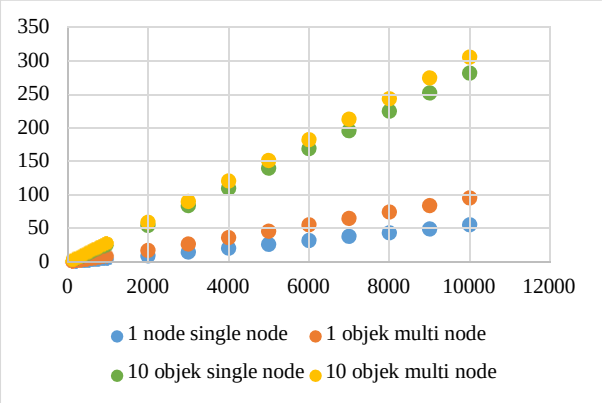
dengan nilai $R^2 = 0.99996$

Untuk 16 *thread* :

$$y = 0.0077x + 0.1775 \quad \ldots\ldots (12)$$

dengan nilai $R^2 = 0.999994$

dimana x adalah jumlah *frame* dan y adalah waktu render dalam menit.

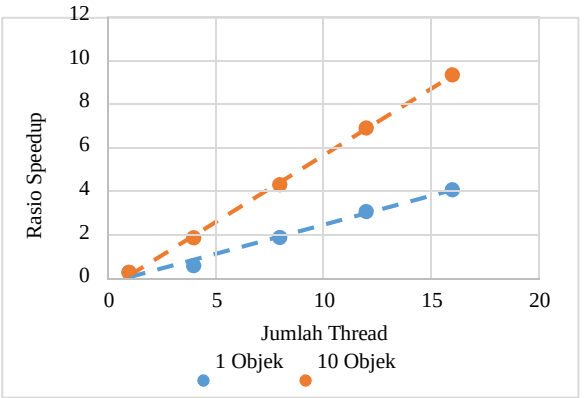


Gambar 4-3 : Perbandingan waktu *render cluster* pada 4 *thread single node* dengan 4 *thread multi node* untuk 1 objek *dolphin* dan 10 objek *dolphin*

Gambar 4-3 menunjukkan perbedaan waktu proses *render* pada *single node* lebih singkat dibandingkan pada *multi node* walaupun menggunakan jumlah *thread* yang sama. Ini dikarenakan pada *multi node* terdapat waktu transfer antara *client*, *master* dan *slave* yang menyebabkan penundaan.

Waktu *render* mengalami peningkatan ketika objek ditambah menjadi 10 objek, tetapi peningkatannya tidak sampai 10 kali, peningkatan rata-ratanya hanya sekitar 4.13 kali. Ini dikarenakan oleh proses *clipping* yang terjadi ketika proses *render* berjalan. Proses ini akan membuang objek yang tidak terlihat dari sudut pandang *user*, jadi ada beberapa bagian yang tidak diproses sampai selesai.

Speedup adalah peningkatan kecepatan waktu proses *render* terhadap jumlah *thread*. Rasio *speedup* didapat dari hasil bagi waktu *render* pada 1 *thread* dengan waktu render pada *thread* yang lebih banyak.



Gambar 4-4 : Grafik *speedup render cluster* untuk 1 objek *dolphin* dan 10 objek *dolphin* berdasarkan perubahan jumlah *thread*

Rasio *speedup* 10 objek lebih tinggi dibandingkan dengan rasio speedup 1 objek. Ini dikarenakan objek yang memiliki kompleksitas sederhana dan masih mampu dikerjakan secara serial apabila diproses secara paralel waktu prosesnya tidak akan mengalami penurunan yang signifikan. Sedangkan objek yang memiliki kompleksitas tinggi dan membebani apabila diproses secara serial dilakukan secara paralel, beban akan terbagi dengan merata sehingga waktu proses akan mengalami penurunan yang signifikan.

Berdasarkan grafik di atas maka persamaan linier untuk *speedup* dapat ditentukan yang dapat digunakan untuk

memprediksi berapa *speedup* yang didapat apabila proses *render* dilakukan dengan menggunakan *thread* yang lebih banyak. Berikut adalah persamaan linier untuk *speedup*.

Untuk 1 objek :

$$y = 0.2658x + 0.5117 \quad \ldots\ldots (13)$$

dengan nilai $R^2 = 0.988469$

Untuk 10 objek :

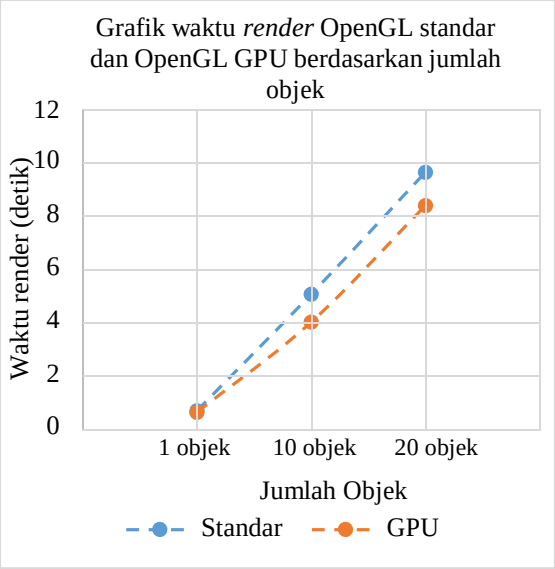
$$y = 0.6092x + 0.2487 \quad \ldots\ldots (14)$$

dengan nilai $R^2 = 0.999058$

dimana x adalah jumlah *thread* dan y adalah rasio *speedup*.

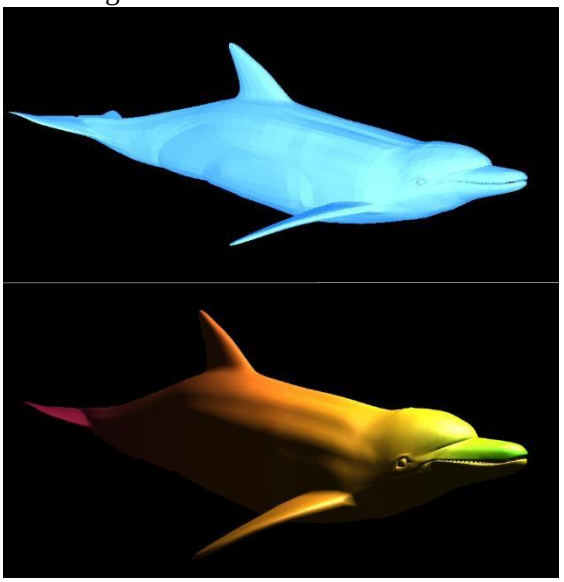
4.2 OpenGL

Pengujian dilakukan dengan membandingkan waktu eksekusi dan hasil *render* dari OpenGL standar dan OpenGL GPU. Setiap pengujian akan dilakukan sebanyak tiga kali dan waktu proses *rendering* yang dicatat merupakan rata-rata dari ketiga pengujian tersebut. Pada OpenGL standar hanya membaca *file* objek yang didalamnya terdapat koordinat vertex yang membentuk suatu objek. OpenGL GPU terdapat proses tambahan, yaitu *shading* dan *texturing* yang merupakan proses *default* pada OpenGL GPU. Objek yang diuji adalah objek *dolphin* yang berjumah 1 objek, 10 objek, dan 20 objek.



Gambar 4-5 : Grafik waktu *render* OpenGL standar dan OpenGL GPU berdasarkan jumlah objek *dolphin*.

Berdasarkan gambar 4-5, walaupun OpenGL GPU ditambah proses *shading* dan *texturing*, waktu prosesnya lebih singkat dibandingkan OpenGL standar. Ini dikarenakan pada OpenGL standar seluruh data objek disimpan pada *default buffer* pada RAM, dan ketika objek dimunculkan, data tersebut dikirim secara terus menerus ke GPU untuk ditampilkan pada layar monitor. Pada OpenGL GPU seluruh data objek disimpan *framebuffer* pada GPU yang memiliki kecepatan proses yang lebih tinggi dibandingkan dengan *default buffer* pada RAM, dan diproses oleh GPU yang menyebabkan waktu eksekusinya lebih singkat.



Gambar 4-6 : Perbandingan hasil *render* OpenGL standar (atas) dan OpenGL GPU (bawah).

Berdasarkan gambar di atas, hasil *render* OpenGL GPU memiliki objek yang lebih halus dan memiliki proporsi warna yang lebih baik dibandingkan hasil *render* OpenGL standar. Ini dikarenakan oleh proses tambahan yaitu proses *shading* dan *texturing*. Kedua proses ini adalah bagian *post-processing* yang secara *default* terdapat pada OpenGL GPU. Kedua proses ini yang menyebabkan objek terlihat lebih detail dan lebih halus. *Shading* adalah proses penempatan bayangan berdasarkan area objek, dengan menempatkan bayangan yang lebih gelap pada area yang jauh dari cahaya dan hanya sedikit bayangan pada bagian yang terkena sedikit cahaya. *Texturing* adalah proses untuk menambahkan detail permukaan atau menambahkan warna pada sebuah objek.

5. Penutup

5.1 Kesimpulan

Kesimpulan yang dapat diambil dari penelitian tugas akhir ini adalah sebagai berikut.

1) *Parallel rendering* berhasil diterapkan pada *cluster* dengan menggunakan *plugin network render* blender yang menggunakan *master*, *client*, dan *slave*. Hasil *rendering* berupa animasi 3D yang bisa ditampilkan pada komputer atau *video player*.
2) *Rendering* berhasil diterapkan pada OpenGL GPU dengan mengakses *framebuffer* GPU untuk menyimpan koordinat *vertex* dan *polygon* objek dan merender objek. Hasil rendering berupa *file executable* (.exe) yang akan menampilkan objek *dolphin*.
3) Proses *rendering* yang dilakukan secara paralel memerlukan waktu proses yang lebih singkat, karena proses *render* dilakukan oleh banyak *thread*. Ketika dijalankan secara paralel pada 16 *thread*, waktu *render* mengalami *speedup* sebanyak 4.7 kali untuk 1 objek dan 10 kali untu 10 objek.
4) Waktu eksekusi pada OpenGL GPU lebih singkat dibandingkan OpenGL standar, walaupun pada OpenGL GPU terdapat proses tambahan, yaitu *shading* dan *texturing*. Ini dikarenakan pada OpenGL GPU, data disimpan dan diproses di GPU sehingga waktu prosesnya lebih singkat.
5) Peningkatan waktu berbanding lurus sesuai dengan penambahan jumlah *frame*, sedangkan ketika objek ditambah menjadi 10 objek, peningkatan waktu *render* tidak sampai 10 kali. Ini dikarenakan adanya proses *clipping*, yaitu proses yang akan memotong atau membuang bagian objek yang tidak terlihat dari sudut pandang *user* ketika proses *render* berjalan, dan tidak diproses hingga selesai.

5.2 Saran

Saran yang dapat diperhatikan untuk mengembangkan penelitian tugas akhir ini adalah sebagai berikut.

1) Menggunakan objek yang lebih kompleks dengan jumlah *frame* yang lebih banyak.
2) Menggunakan tipe *parallel rendering* yang lebih baik dan menggunakan *setting render* yang lebih variatif.

Daftar Pustaka

[1] Blender Foundation. (2014, Mei). Blender Foundation – blender.org – Home of the Blender project – Free and Open 3d Creation Software. http://www.blender.org/foundation/

[2] Chong, A., Sourin, A., Levinski, K. (2006). Grid-based Computer Animation Rendering.

[3] Crockett, T. W. (1996). An Introduction to Parallel Rendering. NASA Langley Research Center, Hampton.

[4] Hypergraph. (2014, Mei). 2D Modelling Transformation : Introduction. http://www.siggraph.org/education/materials/HyperGraph/modeling/mod_tran/2dintr.htm

[5] Kaushik,K. (2008). Cluster Computing. Cochin University of Science and Technology.

[6] NVIDIA. The OpenGL FrameBuffer Object Extension. NVIDIA Corporation.

[7] Ou, L., Fang,Y. C., Celebioglu, O., Mashayekhi, V. (2007). Parallel Rendering Technologes for HPC Clusters. Dell.Inc.

[8] Smith, A. R. (1984). The Viewing Transformation. Pixar, San Rafael.

[9] Wulandari,L. Pencahayaan (Lighting). S1 Teknik Informatika Universitas Gunadarma