Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.202071159
Vol. 7, No.1, Februari 2020, hlm. 37-40 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 30/E/KPT/2018 e-ISSN: 2528-6579

PENGUKURAN PERFORMA ALGORITMA CULLING UNTUK OPTIMASI GRAFIS PADA GAME TIGA DIMENSI

Kholid Fathoni[*1], Nur Shabrina Shaliha[2], Halimatus Sa'dyah[3] , Dwi Kurnia Basuki[4] , Arif Basofi[5]

[1,2,3,4,5]Politeknik Elektronika Negeri Surabaya
E-mail: [1]kholid@pens.ac.id, [2]sabineswift@gmail.com, [3]halimatus@pens.ac.id, [4]dwiki@pens.ac.id, [5]ariv@pens.ac.id
*Penulis Korespondensi

(Naskah Masuk : 19 Oktober 2018, diterima untuk diterbitkan : 13 Januari 2020)

Abstrak

Saat ini, teknologi game berkembang pesat. Terdapat banyak pengembang game yang mengembangkan game menggunakan grafis 3D. Meskipun teknologi GPU berkembang dengan pesat, optimasi tetap dibutuhkan seiring dengan perkembangan teknologi visualisasi. Penelitian ini mengulas performa algoritma *Occlusion Culling* berbasis GPU serta algoritma *Occlusion Culling* klasik dalam melakukan rendering pada game tiga dimensi. Berdasarkan penelitian ini, kami ingin mengetahui apakah *Occlusion Culling* berbasis GPU relevan untuk digunakan pada proses *rendering* game tiga dimensi.

Kata kunci: *Algoritma Culling, Grafika Komputer, Game 3D*

CULLING ALGORITHM PERFORMANCE MEASUREMENT FOR GRAPHIC OPTIMIZATION IN THREE DIMENSIONAL GAMES

Abstract

Game technology is getting bigger. There are many games developed using 3D graphics. Although GPU technology is growing rapidly, we still need optimization technique because the complexity of visualization always increases day by day. In this paper, we want to investigate the performance of GPU driven occlusion culling algorithm and classic occlusion culling in 3D Game Rendering. According to this research, we want to find whether GPU driven occlusion culling relevant for 3D Game or not.

Keywords: *Culling Algorithm, Computer Graphics, 3D Game*

1. PENDAHULUAN

Kualitas visual game tiga dimensi terus bertambah dari tahun ke tahun. Hal ini tidak lepas dari perkembangan penelitian di bidang grafika komputer. Namun, tantangan untuk mengoptimalkan proses *rendering* pada game tiga dimensi tidak pernah habis meskipun perkembangan penelitian di bidang grafika komputer terus berkembang dari sisi perangkat lunak maupun sisi perangkat keras.

Di sisi perangkat keras, kecepatan GPU dan CPU semakin meningkat dalam memproses *rendering*. Hanya saja, hal tersebut tidak sebanding dengan kompleksitas gambar yang harus melalui proses *rendering.* Semakin hari, gambar 3D semakin mendekati gambar di dunia nyata sehingga kompleksitas objekpun terus meningkat.

Adapun dari sisi perangkat lunak, API dari CUDA dan OpenGL juga mengalami perkembangan (Lauritzen, 2017). Namun kedua API tersebut tidak khusus dirancang untuk mengembangkan game tiga dimensi sehingga belum ada fitur yang dapat mengoptimalisasi rendering untu *scene* yang dinamis dan realtime (Lauritzen, 2017).

Proses *rendering* pada game tiga dimensi berbeda dengan proses *rendering* dari film tiga dimensi. Pada film tiga dimensi, urutan objek yang harus melalui proses *rendering* sudah didefinisikan sejak awal dan tidak akan berubah. Sedangkan pada video game, urutan objek yang harus melalui proses *rendering* ditentukan berdasarkan aktivitas pemain game. Oleh karena itu, optimalisasi untuk proses *rendering* memerlukan usaha yang lebih banyak.

Saat ini, riset di bidang optimasi *rendering* untuk game memiliki beberapa cabang di antaranya tentang *Global Iluminating, Occlusion Culling,* dan *Deep Learning* untuk *Real-Time Rendering.* Barre-Brisebois membahas dengan detail tentang perkembangan penelitian mengenai *Global*

Iluminating (Barré-Brisebois, 2017). Sedangkan Salvi membahas dengan detail tentang perkembangan penelitian mengenai penerapan *Deep Learning* pada proses *rendering* (Salvi, 2017).

Adapun penelitian tentang *Occlusion Culling* sempat stagnan dan tidak mengalami banyak pengembangan pada dekade 2000-an. Jika kita mencari penelitian tentang *Occlusion Culling,* makalah yang kita temukan banyak diterbitkan di antara tahun 1990 hingga 2000. Perkembangan penelitian tentang *occlusion culling* di antara tahun tersebut telah dibahas oleh Pantazopoulos dan Tzafestas (Pantazopoulos & Tzafestas, 2002).

Penelitian tentang *Occlusion Culling* mulai kembali ramai seiring dengan perkembangan GPU yang bertransformasi menjadi *programmable* GPU. Salah satu penelitian tentang *Occlusion Culling* berbasis *programmable* GPU dapat dilihat pada algoritma yang dikembangkan oleh Haar dan Aaltonen (Haar & Aaltonen, 2015). Dalam algoritma yang dikembangkan Haar dan Aaltonen, proses *Occlusion Query* serta proses eksekusi perintah *rendering* dijalankan secara keseluruhan di GPU.

Dalam makalah ini, kami ingin menguji performa algoritma *Occlusion Culling* berbasis GPU jika dibandingkan dengan *Occlusion Culling* klasik pada *kasus rendering* game tiga dimensi. Tujuan dari pengujian ini adalah untuk mengetahui arah pengembangan algoritma *Occlusion Culiing* pada kasus *rendering* game tiga dimensi. Makalah ini dibagi menjadi tiga bab. Bab pertama membahas tentang pendahuluan, bab kedua membahas tentang algoritma *Occlusion Culling,* sedangkan bab ketiga membahas tentang pengujian algoritma *Occlusion Culling* untuk kasus game tiga dimensi.

2. TEORI ALGORITMA *OCCLUSION CULLING*

Occlusion Culling adalah algoritma yang digunakan untuk mengurangi beban *rendering* dengan menghilangkan benda-benda yang tidak ditampilkan di layar karena tertutup oleh benda lainnya. Algoritma ini bekerja sebelum proses *rendering* dilakukan (Boreskov & Shikin, 2014).

Pada umumnya, *Occlusion Culling* dikembangkan dengan menggabungkan algoritma *Viewing Frustum* dan *Occluders.* Frustum Culling bertugas untuk menonaktifkan proses *rendering* untuk objek yang berada di luar area tampilan kamera. Sedangkan *Occluders* dimanfaatkan untuk menonaktifkan sesuatu yang tersembunyi dari pandangan oleh benda penghalang (Boreskov & Shikin, 2014).

Implementasi algoritma ini bisa dilakukan di tingkat aplikasi maupun geometri (application stage and geometry stage lihat gambar). Dioptimalkan dengan penerapan ke Z-Buffer atau dengan kata lain penggunaan algoritma *Occlusion Culling* pada *Z-Buffer*. Pendekatan ini bisa membuat Z-Buffer hanya melakukan *rendering* pada objek yang terlihat, sebelum pengiriman data ke pipeline.

```
OcclusionCullingAlgorithm (G)
1.    OR = empty
2.    For each object g in G
3.      If (isOccluded(g,OR))
4.            Skip(g)
5.      Else
6.            Render(g)
7.      Update(OR,g)
8.    End For
9.    End Function
```

Gambar 1. Pseudocode *Occlusion Culling*
Sumber: Computer Graphics from Pixels to Programmable Graphic Hardware (Boreskov & Shikin, 2014).

Pseudocode algoritma *Occlusion Culling* dapat dilihat pada Gambar 1. Pada algoritma klasik, proses seleksi objek (*hidden surface determination)* yang akan direnderdilakukan melalui CPU. Sedangkan rasterisasi dilakukan di GPU. Variasi pengembangan *Occlusion Culling* klasik ada pada optimalisasi seleksi objek (Pantazopoulos dan Tzafestas, 2002).

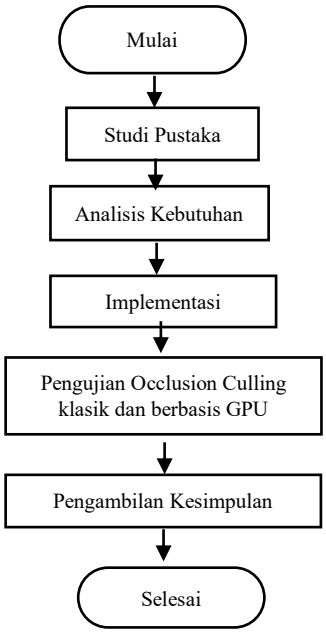
Kelemahan algoritma *Occlusion Culling* klasik terletak mekanisme seleksi objek yang dilakukan oleh CPU. Dalam *rendering pipeline* klasik, mekanisme seleksi objek di CPU dan proses *rendering* di GPU tidak dapat dilakukan secara bersamaan. Hal ini menyebabkan *bottleneck* problem dimana GPU yang seharusnya dapat memproses *rendering* dengan cepat harus menunggu proses seleksi objek di CPU. Hasilnya, proses *rendering* tetap tidak optimal.

Pada algoritma *Occlusion Culling* berbasi GPU, proses seleksi objek dilakukan di GPU. Selanjutnya, GPU mengirimkan data objek yang harus di-*render* untuk disimpan di CPU. Setelah itu, CPU mengirim perintah kepada GPU untuk melakukan proses *rendering*. Proses ini dapat memangkas waktu tunggu seleksi objek dari CPU ke GPU karena GPU dapat memproses seleksi dengan lebih cepat.

3. METODE PENELITIAN

Penelitian ini dimulai dengan melakukan studi pustaka terkait dengan optimasi grafis pada game 3 dimensi sehingga dapat menghemat sumber daya perangkat keras. Penulis mengambil satu metode optimasi grafis yaitu Occlusion Culling yang sudah banyak dimanfaatkan di berbagai *game engine*. Selanjutnya penulis mengimplementasikan kebutuhan untuk pengujian performa algoritma Occlusion Culling berbasis GPU dan Occlusion Culling klasik pada kasus rendering game tiga dimensi dengan beberapa skenario yang masing-masing skenario memiliki perbedaan jumlah verteks serta perbedaan jumlah objek dinamis. Performa yang dimaksud adalah pengaruh skenario dan algoritma terhadap lama waktu rendering. Kesimpulan yang didapat dari pengujian ini nantinya adalah kepastian

mengenai arah pengembangan algoritma Occlusion Culling pada kasus *rendering game* tiga dimensi. Alur metode penelitian ditunjukkan gambar 2.



Gambar 2. Alur Metode Penelitian

4. PENGUJIAN ALGORITMA *OCCLUSION CULLING*

Pembahasan utama dalam makalah ini adalah pengujian performa *Occlusion Culling*. Pengujian kali ini dilakukan dilakukan pada komputer dengan prosesor Intel Core i7 dengan GPU Nvidia GeForce 850M 4GB. Adapun algoritma yang digunakan adalah algoritma *Occlusion Culling* klasik serta algoritma *Occlusion Culling* berbasis GPU yang diusulkan oleh Haar dan Aaltonen.

Tabel 1. Skenario Pengujian

Skenario	Jumlah Vertex	Jumlah Objek Dinamis (%)
Skenario 1	500.000	25
Skenario 2	500.000	50
Skenario 3	500.000	75
Skenario 4	3000.000	25
Skenario 5	3000.000	50
Skenario 6	3000.000	75

Pada pengujian ini, kedua algoritma tersebut dimodifikasi pada bagian perintah *rendering* untuk objek yang bersifat statis. Dalam penelitian ini, kami memberi ID pada objek-objek yang bersifat statis dan menyimpannya ke sebuah array yang akan diproses oleh CPU. Tujuan pemberian ID ini adalah untuk meringankan beban komputer dalam melakukan seleksi objek yang akan di-*render*. Adapun objek-objek yang bersifat dinamis sepenuhnya diproses oleh kedua algoritma yang akan diuji. Pengujian ini dilakukan dalam enam skenario yang tertulis pada Tabel 1.

Tabel 2. Hasil Pengujian pada *Occlusion Culling* Klasik

Skenario	Waktu Rendering (time)
Skenario 1	Time < 10 ms
Skenario 2	14 ms < Time < 20 ms
Skenario 3	16 ms < Time < 25 ms
Skenario 4	Time < 10 ms
Skenario 5	14 ms < Time < 20 ms
Skenario 6	16 ms < Time < 25 ms

Tabel 3. Hasil Pengujian pada *Occlusion Culling* Berbasis GPU

Skenario	Waktu Rendering (time)
Skenario 1	Time < 5 ms
Skenario 2	Time < 5 ms
Skenario 3	Time < 5 ms
Skenario 4	Time < 5 ms
Skenario 5	Time < 5 ms
Skenario 6	Time < 5 ms

Pada pengujian ini, objek yang dirender adalah asset dari game tiga dimensi yang tidak diberi tekstur dan pencahayaan seperti pada Gambar 2. Objek yang akan diuji diatur sedemikian rupa karena pengujian berfokus pada algoritma *Occlusion Culling* untuk objek statis dan dinamis sesuai dengan game tiga dimensi pada umumnya,

Adapun hasil pengujian dapat kita baca pada Tabel 2 dan Tabel 3. Dalam pengujian ini, setiap skenario dijalankan sebanyak sepuluh kali. Data hasil pengujian disajikan dalam *range* (bukan dalam bentuk rata-rata) agar kita mengetahui rentang waktu yang dibutuhkan oleh komputer untuk memproses *rendering*.

Gambar 2. Visualisasi Objek yang akan Di-render

5. KESIMPULAN

Dalam Tabel 2 terlihat jumlah objek statis dan dinamis cukup mempengaruhi hasil pengujian. Sedangkan hasil pengujian dalam Tabel 3 terlihat stagnan. Jumlah objek statis dan dinamis tidak berpengaruh pada kecepatan algoritma untuk memproses objek yang harus di-*render*.

Berdasarkan pengujian yang dilakukan, terlihat bahwa optimasi di sisi proses seleksi objek (*Hidden Surface Determination)* tidak banyak berpengaruh pada algoritma *Occlusion Culling* berbasis GPU. Hal ini menguatkan hipotesa bahwa penelitian tentang *Occlusion Culling* seharusnya dikembangkan ke arah manajemen perintah pada GPU. Jika memungkinkan,

parallel processing dapat dilakukan untuk mempercepat proses rendering.

UCAPAN TERIMA KASIH

Ucapan terima kasih penulis sampaikan kepada Direktorat Riset dan Pengabdian Masyarakat (DRPM), Direktorat Jenderal Penguatan Riset dan Pengembangan, (DIRJEN RISBANG) Kementrian Riset, Teknologi, dan Pendidikan Tinggi (KEMENRISTEKDIKTI) atas pembiayaan penelitian ini melalui hibah Penelitian Dasar Unggulan Perguruan Tinggi (PDUPT) tahun 2018.

DAFTAR PUSTAKA

BARRÉ-BRISEBOIS, C. 2017. A Certain Slant of Light: Past, Present and Future Challenges of Global Illumination in Games. *Lecturer on SIGGRAPH 2017*. Los Angeles, USA: SIGGRAPH.

BORESKOV, A., & SHIKIN, E. 2014. *Computer Graphics from Pixel to Programmable Graphic Hardware.* Florida: CRC Press.

HAAR, U., & AALTONEN, S. 2015. GPU-Driven Rendering Pipeline. *Lecturer on SIGGRAPH 2015*. Los Angeles, USA: SIGGRAPH.

LAURITZEN, A. 2017. Future Direction For Compute For Graphic. *Lecturer on Siggraph 2017*. Los Angeles, United States of America: SIGGRAPH.

PANTAZOPOULOS, I., & TZAFESTAS, S. 2002. Occlusion Culling Algorithms: A Comprehensive Survey. *Journal of Intelligent and Robotic Systems vol 35*, 123-156.

SALVI, M. 2017. Deep Learning: The Future of Realtime Rendering. *Lecturer on SIGGRAPH 2017*. Los Angeles, USA: SIGGRAPH.