Jurnal Ismetek *ISSN 2406-9841*
Vol. 16 No.2 Desember Tahun 2023 *e-IISN 2986-2973*

IMPLEMENTASI *BLOOM* SEBAGAI EFEK GAMBAR DALAM PERMAINAN VIDEO

Lola

Program Studi Teknik Informatika, FTI, Institut Teknologi Budi Utomo Jakarta,
lola.rezak@gmail.com

Abstrak
Efek gambar sering digunakan dalam permainan video sebagai perbaikan citra untuk menghasilkan gambar yang tampak seperti nyata. Salah satu efek gambar yang umum dipakai adalah *bloom* dengan mensimulasikan mekanisme fisik dari mata manusia sehingga menimbulkan scattering atau difraksi terhadap objek yang terang. Makalah ini membahas tentang cara implementasi *bloom* yang digunakan dalam permainan video.

Kata kunci : Image Effect, Video Games, Bloom.

1. PENDAHULUAN

Permainan video merupakan salah satu bentuk dari Computer Generated Imagery (CGI) yang dibuat secara *real-time*. Resolusi gambar dan frame rate dari permainan sangat mempengaruhi kinerja dari pemain sehingga berdampak pada *playability* dan *enjoyability* permainan. Pada permainan video shooter misalnya, setiap detik permainan membutuhkan sekitar 60 frames, biasa disebut sebagai 60 frames-per-second (FPS), untuk meningkatkan performa sebesar 7 kali lipat dari 3 FPS yang hampir tidak dapat dimainkan karena pemain tidak dapat mentargetkan musuh pada frame rate yang kecil. Selan itu, membuat gambar yang terlihat nyata dari CGI membutuhkan bantuan pemrosesan gambar untuk mensimulasikan efek pada dunia nyata menjadi efek digital. Namun, efek gambar tersebut harus dapat diimplementasikan secara real-time sehingga tidak mengurangi kesenangan dari pemain.

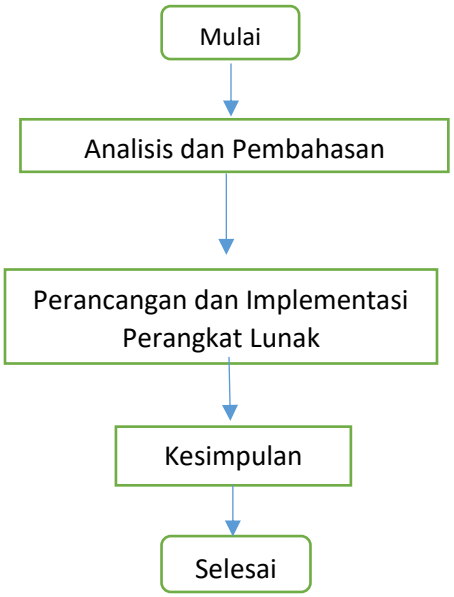
Dalam dunia nyata, kita menemukan objek-objek terang dengan cahaya yang terlihat menyebar di sekitarnya. Efek tersebut biasa disebut sebagai *bloom* atau *glow*. *Bloom* merupakan efek visual yang sering digunakan dalam permainan video dan film untuk menciptakan kesan cahaya yang menyebar tersebut. Efek ini dapat membuat objek tampak lebih terang dan menonjol, sehingga menambah suasana pada sebuah scene. Efek bloom dalam dunia nyata ditimbulkan dari efek yang dihasilkan oleh mata manusia berupa *scattering* pada kornea, lensa, dan retina, serta *diffraction* pada struktur sel koheren pada area radial di luar lensa (G. Spencer, 2021). Glow dan halo dari cahaya ini memberikan isyarat visual mengenai kecerahan gambar dan atmosfer dari scene. Dengan berkembangnya perangkat keras pemrosesan grafis atau *Graphics Processing Unit (GPU)*, kalkulasi efek ini dapat dilakukan hanya dengan beberapa operasi rendering sederhana, sehingga dapat dijalankan secara *real-time* dan memberikan tampilan yang lebih realistis terhadap objek-objek terang (G. James, 2020).

Terdapat beberapa metode yang dapat digunakan untuk memperbaiki citra, baik dalam domain frekuensi maupun spasial. Salah satu pendekatan yang umum digunakan untuk meningkatkan citra adalah dengan menggunakan penapis lolos rendah seperti gaussian blur. Filter ini dapat menciptakan efek visual yang mengesankan dan diaplikasikan untuk memberikan tampilan yang lebih halus pada gambar. Salah satu aplikasi efek ini adalah simulasi cahaya terang *(bloom*) pada objek-objek dalam suatu scene.

Pada makalah ini, akan diulas mengenai implementasi efek *bloom* yang telah ada dengan pengolahan citra pada permainan video baik dalam domain spasial maupun frekuensi. Pada domain spasial, akan digunakan penapis lolos rendah seperti Gaussian, sedangkan pada domain frekuensi akan menggunakan gambar kernel. Selain itu, akan dibahas efek gambar *bloom real-time* yang dapat digunakan dalam permainan video, serta implementasinya dalam *game engine popular*.

2. METODOLOGI

Metodologi penelitian digambarkan dalam bentuk diagram alir sebagai berikut :

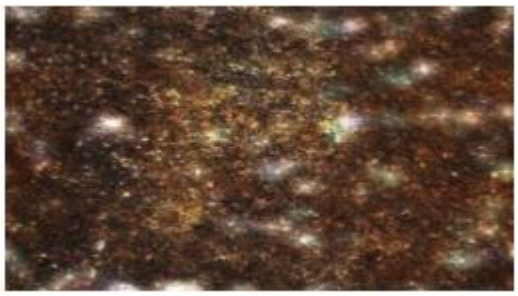


Gambar 1. Diagram Alir Metodologi Penelitian
Sumber: https://www.researchgate.net/publication/338235695_Metode-Metode_Penelitian_Dalam_Penulisan_Jurnal_Ilmiah_Elektronik

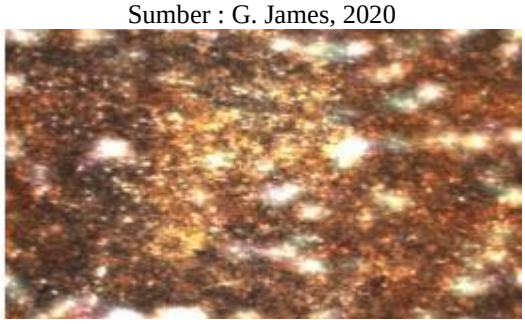
3. HASIL DAN PEMBAHASAN

3.1 HASIL

Hasil bloom menggunakan penapis lolos rendah Gaussian:



Gambar 2. Citra Sebelum Konvolusi.
Sumber : G. James, 2020



Gambar 3. Citra Setelah Konvolusi.
Sumber : G. James, 2020

Dapat dilihat bahwa citra memiliki kecerahan yang lebih tinggi, terutama pada objek yang terang. Kecerahan lebih tersebar secara merata.

3.2 PEMBAHASAN

Pada makalah ini, akan diulas dua cara dalam menerapkan efek gambar bloom yang telah ada. Pertama adalah yang paling umum digunakan dalam permainan video, yaitu blur atau pelembutan dengan penapis lolos rendah (Gaussian). Kedua adalah pendekatan lain yang digunakan oleh Unreal Engine, yaitu kakas untuk membuat permainan video atau sering disebut sebagai game engine.

A. Pelembutan dengan Penapis Lolos Rendah Gaussian

Untuk membuat efek citra berupa *bloom*, cara umum yang sering digunakan adalah menggunakan penapis lolos rendah utnuk melembutkan citra atau blur. Berikut adalah tahapan dalam melakukan implementasi bloom:

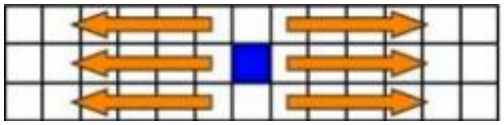
1. Tentukan asal citra yang akan dilakukan *bloom*.
2. Lakukan *downsampling* terhadap tekstur untuk performa sehingga pemrosesan akan dilakukan lebih cepat.

3. Terapkan blur menggunakan penapis lolos rendah pada citra.
4. Hasil blur kemudian ditambahkan ke citra asli.

Lebih lanjut, implementasi blur dalam menciptakan efek *bloom* melibatkan penggunaan penapisan untuk pemrosesan citra dua dimensi menggunakan penapis lolos rendah seperti Gaussian. Proses blur atau pelembutan ini sangat penting untuk mendapatkan efek bloom yang lembut dan alami. Efisiensi operasi blur secara signifikan mempengaruhi kecepatan dalam menghasilkan efek bloom. Waktu yang diperlukan untuk melakukan blur bergantung pada ukuran kernel penapis lolos rendah.

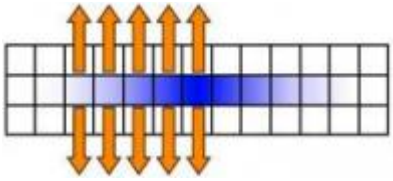
Untuk mengatasi pemrosesan citra dengan ukuran blur yang lebih besar, digunakan pendekatan dua langkah yang disebut dengan *separable convolution*. Pelembutan menggunakan Gaussian dapat memanfaatkan sifat separable dari fungsi Gaussian. Pendekatan ini akan mengurangi perhitungan dari pangkat dua ukuran kernel Gaussian k 2 menjadi hanya dua kali ukuran kernel Gaussian $2k$, sehingga beban untuk membuat *bloom* dengan ukuran yang lebih besar akan menjadi ringan. Berikut adalah tahapan dalam melakukan *separable convolution.*

1. Lakukan pelembutan citra asli secara horizontal.



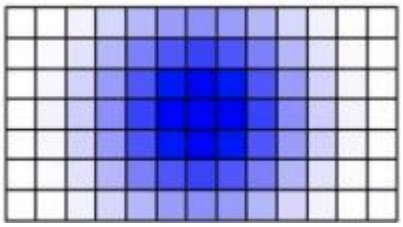
Gambar 4. Pelembutan Citra Asli Secara Horizontal.
Sumber : G. Spencer, 2021

2. Hasil dari pelembutan secara horizontal, kemudian dilembutkan secara vertikal.



Gambar 5. Pelembutan Citra Asli Secara Horizontal, Kemudian Dilembutkan Secara Vertikal.
Sumber : G. Spencer, 2021

3. Hasil blur akan terlihat.



Gambar 6. Hasil Blur Terlihat
Sumber : G. Spencer, 2021

B. Konvolusi pada Domain Frekuensi.

Alternatif lain dalam melakukan implementasi efek gambar bloom adalah melakukan konvolusi dengan kernel pada domain frekuensi. Pendekatan ini dilakukan oleh Unreal Engine (Unreal Engine, 2023) dengan motivasi bahwa penapis Gaussian menyebabkan bloom yang tersebar secara simetris atau sirkular. Selain itu, bloom memiliki beban komputasi konvolusi dalam domain spasial yang tinggi, yakni $O(N\ 2\ )$. Oleh karena itu, Unreal Engine memanfaatkan konvolusi Fast Fourier Transform (FFT) untuk meningkatkan efisiensi komputasi. Berikut adalah langkah-langkah implementasi yang digunakan:

1. FFT pada Citra Awal (*Image_Frequencies*).

Citra yang akan diterapkan konvolusi diubah ke ranah frekuensi melalui FFT. Transformasi ini dilakukan untuk menciptakan representasi citra dalam domain frekuensi, memfasilitasi operasi konvolusi dengan *kernel*.

2. FFT pada Kernel (*Filter_Frequencies*).

Kernel yang akan digunakan untuk konvolusi juga diubah ke ranah frekuensi melalui FFT. Proses ini menghasilkan representasi *kernel* dalam domain frekuensi yang akan digunakan dalam langkah selanjutnya. Langkah ini dilakukan hanya sekali, lalu hasilnya di-*cache* sehingga tidak perlu melakukan transformasi *kernel* berulang

kali. Jika sudah ada cache, maka cukup gunakan hasil yang sudah di-*cache.*

3. Konvolusi dalam Domain Frekuensi.

Pada tahap ini, akan dilakukan proses konvolusi sebenarnya dalam domain frekuensi dengan mengalikan representasi citra pada domain frekuensi (*Image_Frequencies*) dengan representasi *kernel* pada domain frekuensi (*Filter_Frequencies*). Langkah ini memanfaatkan sifat perkalian dalam domain frekuensi untuk menghasilkan citra yang telah mengalami konvolusi.

4. Inverse FFT pada Hasil Konvolusi (*Convolved_Image*).

Setelah konvolusi dalam domain frekuensi selesai, hasilnya dikembalikan ke domain spasial melalui inverse FFT. Langkah ini menghasilkan citra yang telah mengalami konvolusi sesuai dengan kernel yang diaplikasikan sehingga menghasilkan citra dengan efek *bloom*.

Kernel yang digunakan untuk konvolusi merepresentasikan respons dari perangkat optik seperti kamera terhadap sumber titik tunggal di tengah *view field*. Setiap piksel pada sumber berkontribusi terhadap Sebagian dari kecerahannya kepada tetangga-tetangganya sesuai dengan kernel. Semakin terang piksel sumber, semakin terlihat kilau yang dihasilkannya. Kernel harus selalu ada di GPU dan tersedia dalam resolusi penuh. Jika tidak, kualitas citra hasil bloom akan sangat buruk. Contoh kernel yang digunakan adalah sebagai berikut:

Gambar 7. Kernel
Sumber : Unreal Engine, 2023

Kiri adalah gambar kernel asli dan kanan adalah gambar kernel yang diperbesar untuk memperjelas bentuk bloom.

Implementasi konvolusi FFT ini memanfaatkan kecepatan transformasi citra yang relatif lebih cepat dalam domain frekuensi, memiliki kompleksitas transformasi dan inverse transformasi sebesar $O(NlogN)$ dengan konvolusi sebesar $O(N)$, dibandingkan pada domain spasial dengan kompleksitas konvolusi $O(N\ 2\ )$. Dalam komputasi *shader*, implementasi FFT dilakukan dengan tiga tahap: pertama *forward horizontal transform*, kedua *forward vertical transform* dilanjutkan dengan konvolusi lalu *inverse vertical transform* pada tahap yang sama, dan tahap ketiga atau terakhir *inverse horizontal transform*. Tahap-tahap ini memastikan bahwa proses konvolusi FFT dilakukan dengan efisien, mengoptimalkan waktu komputasi untuk menghasilkan bloom yang lebih akurat Hasil dari bloom menggunakan teknik ini, sesuai yang ada pada dokumentasi Unreal Engine (Unreal Engine, 2023) adalah sebagai berikut:

Gambar 8. Sebelum Diterapkan Efek Bloom
Sumber : Unreal Engine, 2023

Gambar 9. Setelah Menerapkan Efek Bloom
Sumber : Unreal Engine, 2023

Terlihat bahwa efek bloom yang dihasilkan terasa lebih sinematik dan realistis. Ketidaksimetrian dari pendekatan ini membuat *bloom* bukan menjadi efek yang membosankan.

4. KESIMPULAN

Efek gambar bloom pada permainan video dapat diimplementasikan dengan menggunakan penapis lolos rendah seperti Gaussian sehingga menciptakan blur atau citra yang halus. Penapis ini mensimulasikan efek yang menciptakan cahaya terang atau halo di sekitar objek terang.

Selain itu, terdapat pendekatan menggunakan gambar kernel yang melakukan konvolusi pada domain frekuensi. Konvolusi dengan gambar kernel pada domain frekuensi memberikan hasil yang lebih bagus daripada penapis Gaussian sederhana. Terakhir, untuk optimisasi sehingga dapat dijalankan secara *real-time*, dilakukan *downsampling* dan implementasi dalam bentuk program GPU

DAFTAR PUSTAKA

1. G. Spencer, P. Shirley, K. Zimmerman, D. P. Greenberg, 2021, Physically-based glare effects for digital images, Proceedings of the 22nd annual conference on Computer graphics and interactive techniques - SIGGRAPH ’95, New York, New York, USA: ACM Press, hlm. 325–334. doi: 10.1145/218380.218466.
2. G. James dan J. O’Rorke, Real-Time Glow, 2020, GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics, 1 ed., R. Fernando, Ed., Pearson Higher Education.
3. Unreal Engine, 2023, Bloom, https://docs.unrealengine.com/5.0/en-US/bloom-in-unreal-engine/