PERBANDINGAN ALGORITMA *HIDDEN SPACE REMOVAL: Z-BUFFER* DAN *SCANLINE* DILIHAT DARI PENGGUNAAN MEMORI DAN KECEPATAN

Djoni Haryadi Setiabudi
Fakultas Teknologi Industri, Jurusan Teknik Informatika - Universistas Kristen Petra
e-mail: djonihs@peter.petra.ac.id

Dody Irwin
Alumnus Fakultas Teknologi Industri, Jurusan Teknik Elektro - Universistas Kristen Petra

ABSTRAK: *Hidden surface removal* adalah suatu algoritma yang digunakan untuk menghilangkan penampilan bagian yang tertutup oleh objek yang didepannya. Apabila ada dua bidang yang berpotongan, apabila ditampilkan biasa tanpa menggunakan algoritma *Hidden surface removal* maka bagian yang berpotongan itu akan tidak kelihatan, oleh karena bidang yang satu ditutupi oleh bagian yang lain tanpa memotong. Oleh karena itu untuk menampilkan bidang perpotongan, diperlukan Algoritma *Hidden surface removal*.

Algoritma *Z buffer* melaksanakan proses *Hidden Surface Removal* dengan memasukkan warna dan kedalaman bidang permukaan yang tampak ke dalam buffer, dan kemudian setelah selesai hasilnya ditampilkan ke layar. Algoritma *Scan Line* melakukan *scanning* untuk setiap baris dari layar bidang gambar untuk setiap permukaan objek pada ruang tiga dimensi dan menampilkan hasilnya setelah melaksanakan proses setiap baris *scanning*-nya. Kedua algoritma ini dibandingkan berdasarkan besar memori dan waktu yang dipergunakan oleh masing-masing algoritma.

Dari hasil penelitian didapatkan bahwa algoritma Scanline menggunakan memori yang lebih sedikit dari algoritma Z-Buffer, sedangkan dari segi kecepatan algoritma *Scan Line* lebih unggul daripada algoritma *Z Buffer* bilamana objek yang ditampilkan pada bidang gambar mengumpul pada baris y, sedangkan *Z Buffer* lebih unggul dari *Scan Line* bila objek yang digambar menyebar dan menggunakan keseluruhan baris pada bidang gambar dengan bidang permukaan yang digambar semakin banyak.

Kata kunci: *Hidden Surface Removal, Z-Buffer, Scanline, computer graphics*

ABSTRACT : *Hidden surface removal is an algorithm used to hide part of the object which is blocked by the object in front of it. If there are two plane crossed each other displayed without Hidden surface removal algorithm, the crossing section is invisible, because one object will block another object without crossing. The crossing sections can be displayed using Hidden surface removal algorithm.*

Z buffer algorithm implements Hidden Surface Removal by entering color and depth of the visible plane into the buffer, then displays the result on the screen. Scan Line algorithm will scanning the screen row by row of each object surface in three dimension and then displays on the screen after each row scanning. Both of the algorithms will be compared based on the memory usage dan time needed to execute.

The experiment shows that Scanline algorithm uses less memory compared with Z-Buffer algoritm. Furthermore, based on the speed, the Scanline is better than the Z-Buffer if the object is collected on the y row, but the Z-Buffer is better than the Scanline if the object scattered and used all rows on the drawing plane and has more surface do displayed.

Keywords: Hidden Surface Removal, Z-Buffer, Scanline, computer graphics.

1. PENDAHULUAN

Penggambaran yang dilakukan dengan komputer memungkinkan untuk menghasilkan gambar objek benda tiga dimensi yang menyerupai dengan bentuk yang sebenarnya. Objek tiga dimensi ini mempunyai kedalaman arah x,y dan z. Untuk menggambarkan objek tersebut secara nyata maka bagian yang tidak tampak dari titik pandang harus dihilangkan, permukaan objek yang berada di belakang permukaan objek yang

lainnya harus disembunyikan. Bilamana ada dua permukaan bidang yang berpotongan maka bagian bidang yang tidak terlihat dari titik pandang juga harus disembunyikan.

Untuk melakukan penggambaran objek tiga dimensi untuk menampilkan bagian yang terlihat atau tampak dari titik pandang maka telah dikembangkan algoritma yang dikenal dengan Algoritma *Hidden Surface Removal.*

Dari bermacam bentuk algoritma yang ada maka dipilih dua algoritma untuk dibandingkan mana yang terbaik ditinjau dari ukuran memori yang dipergunakan dan kecepatan eksekusi, yaitu algoritma Z Buffer dan algoritma Scan Line.

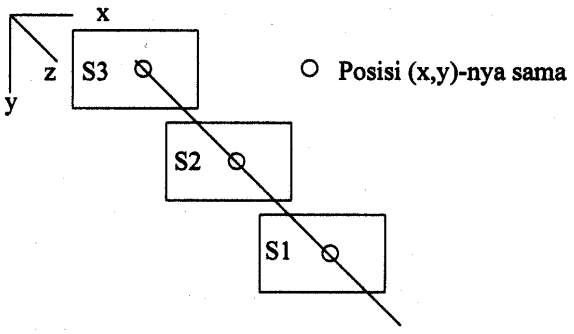
2. ALGORITMA

2.1. Algoritma Z Buffer

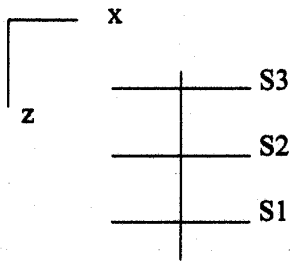
Algoritma Depth Buffer adalah salah satu dari algoritma *Hidden surface removal* yang mempergunakan image space sebagai dasar proses penghitungan tampak atau tidaknya permukaan suatu objek. Algoritma ini melakukan scaning satu kali untuk suatu permukaan objek sampai proses berakhir.

Algoritma ini menguji tampak atau tidaknya setiap pixel pada suatu permukaan objek yang satu terhadap permukaan objek yang lain dan harga permukaan yang paling dekat dengan bidang pandang yang akan tersimpan di dalam *Depth Buffer* dan selanjutnya harga intensitas warna dari permukaan pixel tersebut disimpan di dalam *Refresh Buffer* atau algoritma *Depth Buffer* ini akan menampilkan bagian permukaan objek berdasarkan posisi z yang paling dekat dengan bidang pandang dengan proyeksi orthogonal atau proyeksi tegak lurus. Permukaan untuk suatu titik pada (x,y) untuk setiap permukaan objek diuji mana yang paling dekat dengan bidang pandang. Untuk setiap titik posisi (x,y) pada bidang pandang, permukaan dengan z koordinat terbesar pada posisi itu akan terlihat.

Gambar 1 memperlihatkan tiga permukaan bidang pada berbagai kedalaman z dengan posisi (x,y) yang sama untuk setiap permukaan.



Gambar 1. Posisi (x,y) pada Berbagai Kedalaman Z



Gambar 2. Penampakan S1,S2,S3 pada Koordinat (x,z)

Pada Gambar 2 dapat dilihat bahwa permukaan S1 mempunyai harga z terkecil pada posisi (x,y) ini sehingga harga z disimpan pada Depth Buffer dan harga intensitas S1 pada (x,y) disimpan pada Refresh Buffer.

Algoritma ini membutuhkan dua buffer untuk implementasi, yaitu : *Buffer Depth* dan *Buffer Refresh*.

Depth Buffer digunakan untuk menyimpan harga kedalaman atau harga z masing-masing pixel untuk setiap permukaan objek pada posisi xy pada layar sebagai batasan daerah permukaan yang dibandingkan.

Refresh Buffer digunakan menyimpan harga intensitas(warna) yang dimiliki oleh masing-masing posisi pixel permukaan yang tersimpan pada *Depth Buffer*.

Langkah- langkah algoritma *Depth Buffer adalah* sebagai berikut :

1. Inisialisasi Depth Buffer dan Refresh Buffer sehingga untuk semua koordinat posisi(x,y) depth(x,y) =0 dan refresh(x,y) = *background*.
2. Untuk setiap posisi pada permukaan, bandingkan harga kedalaman terhadap harga yang tersimpan pada depth buffer untuk menentukan penampakan.

a. Hitung harga z untuk setiap posisi(x,y) pada permukaan.
b. Jika z>depth(x,y), kemudian masukan depth(x,y) = z dan refresh (x,y) = i, dimana i adalah harga dari intensitas pada posisi(x,y) di atas permukaan.

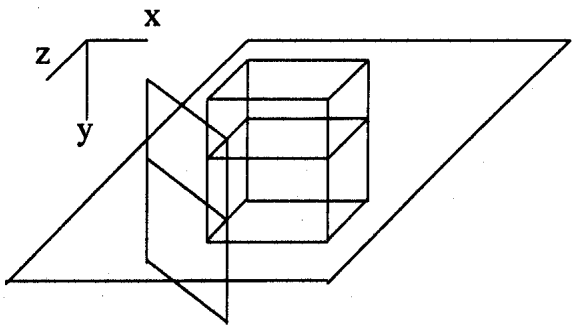
Pada langkah terakhir, jika z lebih kecil dari harga *Depth Buffer* untuk posisi tersebut, titik tidak tampak. Ketika proses ini selesai untuk semua permukaan, *Depth Buffer* berisi harga z untuk permukaan yang tampak dan *Refresh Buffer* berisi hanya harga intensitas.

Metode *Depth Buffer* tidak membutuhkan *sorting* dari permukaan sebuah gambar. T Sebagai contoh, sebuah sistem dengan resolusi 1024 x 1024 membutuhkan lebih dari 1 juta posisi dalam *Depth Buffer*, dengan setiap posisi berisi bit-bit yang cukup untuk menyatakan keperluan peningkatan dari koordinat z.

2.2 Algoritma Scan-Line

Algoritma *Scan Line* adalah salah satu dari algoritma *Hidden Surface Removal* yang digunakan untuk memecahkan masalah penggunaan memori yang besar dengan satu baris scan untuk memproses semua permukaan objek, biasanya *Scan Line* akan men-*sweeping* layar dari atas ke bawah. Dan sebuah baris scan horisontal bidang y di coba untuk semua permukaan dari objek. Perpotongan antara baris scan dan permukaan adalah berupa sebuah garis.

Algoritma melakukan scan dengan arah sumbu y sehingga memotong semua permukaan bidang dengan arah sumbu x dan z dan membuang garis-garis yang tersembunyi. Sebagai ganti menscan suatu permukaan satu kali dalam satu proses, maka akan berhubungan dengan menscan banyak permukaan dalam satu kali proses. Sebagaimana setiap baris scan diproses, semua permukaan polygon dipotong oleh baris scan untuk menentukan mana yang tampak. Pada setiap posisi sepanjang baris scan, perhitungan kedalaman dibuat untuk setiap permukaan untuk menentukan mana yang terdekat dari bidang pandang. Ketika permukaan yang tampak sudah ditentukan, harga intensity dimasukkan ke dalam buffer.



Gambar 3. Operasi Scanline, Baris Scan Memotong Objek

3. IMPLEMENTASI

3.1 Algoritma Z Buffer

Alur proses *Hidden Surface Removal* dengan menggunakan algoritma *Z Buffer* dapat dilihat pada Gambar 4. Proses yang dilakukan oleh *Z Buffer* adalah:

- Meng-inisialisasi isi Buffer
- Melakukan uji penampakan keseluruhan bagian permukaan setiap *link* mulai dari awal *link* hingga akhir *link* sebanyak 1 kali.
- Memindahkan/menampilkan seluruh isi *Z Buffer*

Jadi bilamana *Z Buffer* melakukan proses *Hidden Surface Removal* secara lengkap maka *Z Buffer* sudah melakukan:

a) Menginisialisasi *Depth Buffer* dan *Refresh Buffer* yang berukuran sebesar bidang gambar.
b) Berpindah dari *link* ke *link* berikutnya hanya satu kali dimulai dari awal *link* data permukaan hingga akhir *link* data permukaan untuk uji penampakan.
c) Memindahkan/menampilkan isi *Refresh Buffer* yang berukuran sebesar bidang gambar.

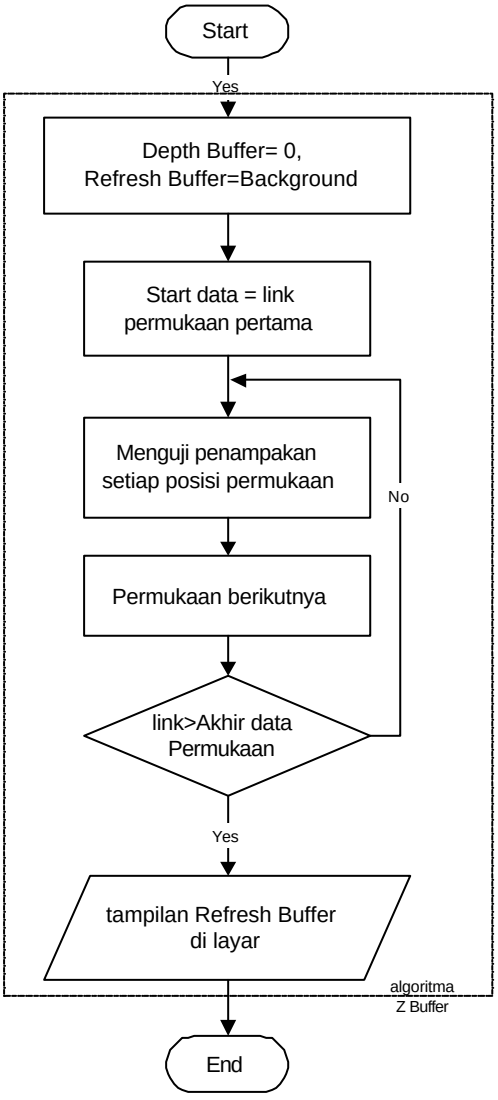
3.2 Algoritma Scan Line

Alur proses *Hidden Surface Removal* dengan menggunakan algoritma *Scan Line* dapat dilihat pada Gambar 5 sesudah ini. Yang dilakukan oleh Algoritma *Scan Line secara garis besar* adalah:

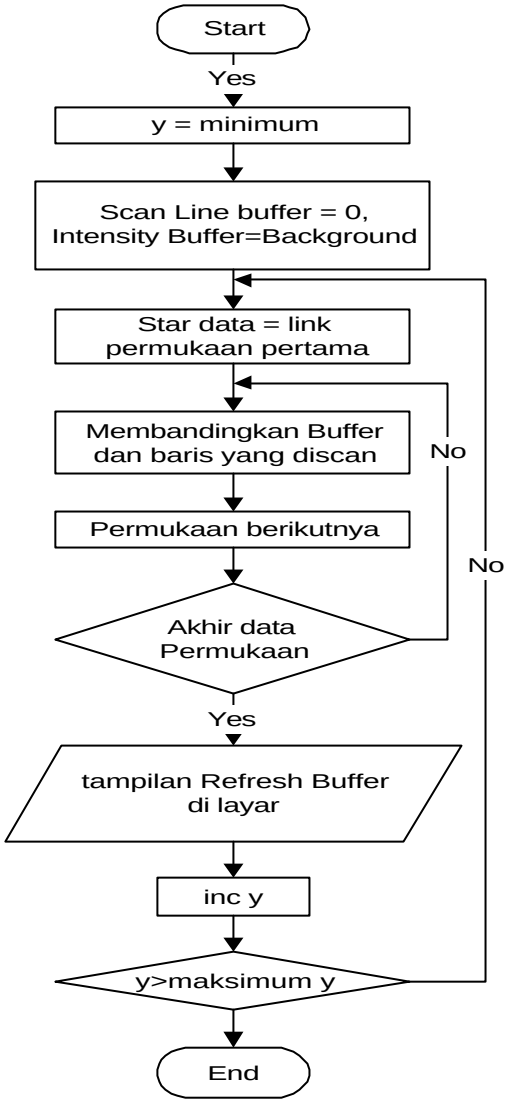
- Meng-inisialisasi Buffer secara berulang.
- Melakukan scan baris yang diperlukan.
- Memindahkan/menampilkan isi buffer satu baris secara berulang.

Dari Penjelasan di atas, proses yang sudah dilakukan oleh algoritma *Scan Line secara garis besar* adalah:

- Meng-inisialisasi *Scan Depth Buffer* dan *Scan Line Buffer* sebanyak baris yang di scan (y maksimum diantara permukaan objek pada image space dikurangi y minimum diantara permukaan objek pada image space).
- Berpindah dari awal *link* ke akhir *link* (pada point b) sebanyak $y_{maks} - y_{min}$.
- Memindahkan/menampilkan isi *Scan Line buffer* sebanyak baris yang di scan. (y maksimum diantara permukaan objek pada image space dikurangi y minimum diantara permukaan objek pada image space)



Gambar 4. Algoritma Z Buffer



Gambar 5. Algoritma Scan Line

4. ANALISIS

4.1 Analisis Program dari Segi Penggunaan Memori

a. Memori untuk Algoritma Z Buffer

Memori yang diperlukan adalah sebesar bidang layar yang akan di gambar dikali dengan besar variabel kedalaman z dan warna. Sebagai contoh, proses *Hidden Surface Removal* dilaksanakan pada layar dengan bidang yang akan digambar berukuran 640 x 480 pixel. Untuk *Z Buffer* diperlukan daerah pada memori dengan ukuran 640 x 480 * 6 byte (ukuran tipe real untuk variabel Z atau harga kedalaman) sama dengan 1843200 byte (1,8 MB). Dengan bidang gambar yang sama maka *Refresh Buffer* memerlukan memori sebesar

640 *480 x 4 byte (ukuran tipe integer untuk variabel warna) atau 640 x 480 x 4 = 1228800 byte atau (1,2288 MB). Jadi untuk keperluan Algoritma *Z Buffer* diperlukan daerah memori sebesar 3072000 byte(3 MB).

b. Memori untuk Scan Line Algoritma

Memori yang diperlukan adalah sebesar jumlah kolom bidang layar yang akan digambar dikalikan dengan besar variable kedalaman dan warna. Sebagai contoh Proses *Hidden Surface Removal* dilaksanakan pada layar dengan bidang yang akan digambar berukuran 640 x 480 pixel. Untuk Buffer *Scan Depth* hanya diperlukan daerah pada memori dengan ukuran 640 * 6 byte (ukuran tipe real untuk variabel Z atau harga kedalaman) atau 640 * 6 = 3840 byte(3,75 Kb). Dengan dimensi bidang gambar yang sama maka untuk buffer *Scan Line* diperlukan memori sebesar 640 * 4 byte (ukuran tipe integer untuk variabel warna) sama dengan 2560 byte(2,5 Kb). Jadi untuk keperluan Algoritma *Scan Line* diperlukan daerah memori sebesar 6400 byte (6,25 KB). Bilamana di bandingkan dengan daerah memori yang diperlukan untuk algoritma *Z Buffer* yaitu sebesar 3.072.000 (3 MB), sedangkan untuk algoritma *Scan Line* hanya 6400 byte maka Algoritma *Scan Line* sudah menghemat penggunaan memori sebesar 3.065.600 byte (2,99375 MB)

4.2 Analisis Algoritma dari segi Kecepatan

a. Analisis Kecepatan Berdasarkan Array yang Diinisialisasi.
- Semakin kecil ($y_{maks} - y_{min} + 1$) maka waktu untuk inisialisasi algoritma *Scan line* akan semakin kecil.
- Waktu yang diperlukan oleh algoritma *Z Buffer* untuk inisialisasi mempunyai waktu konstan yaitu waktu untuk inisialisasi keseluruhan array atau sebesar bidang gambar.
- Waktu untuk inisialisasi kedua algoritma akan sama bila $(y_{maks} - y_{min} + 1) = (maxy + 1)$

b. Analisis Kecepatan Berdasarkan jumlah suatu link yang dilalui
- Semakin kecil ($y_{maks} - y_{min} + 1$) maka jumlah suatu rangkaian *link* yang dilalui algoritma *Scan line* akan semakin sedikit.
- Jumlah suatu rangkaian *link* atau sejumlah *link* dari awal hingga akhir *link* yang dilalui oleh algoritma Z Buffer adalah sebanyak satu kali
- Jumlah rangkaian *link* yang dilalui kedua algoritma akan sama bila $(y_{maks} - y_{min} + 1) = 1$ *atau* $ymaks = ymin$

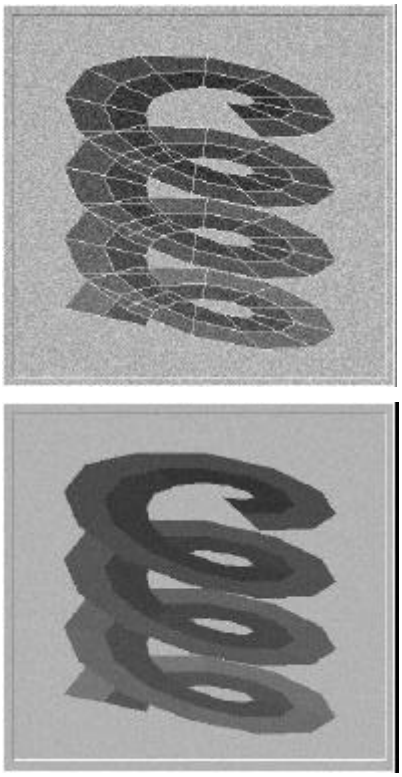
c. Analisis Kecepatan Berdasarkan isi Array Yang ditampilkan.
- Jika semakin kecil ($y_{maks} - y_{min} + 1$) maka waktu untuk menampilkan gambar dengan algoritma *Scan line* akan semakin kecil.
- Waktu yang diperlukan oleh algoritma *Z Buffer* untuk menampilkan isi buffer mempunyai waktu konstan yaitu waktu menampilkan keseluruhan array atau sebesar bidang gambar.
- Waktu untuk menampilkan kedua algoritma akan sama bila $(y_{maks} - y_{min} + 1) = (maxy + 1)$ *atau* $(y_{maks} - y_{min}) = (maxy)$.

Tabel 1. Hasil Analisis Source Code

PEMBANDING	ALGORITMA Z BUFFER	ALGORITMA SCAN LINE
Memori	Maxx*maxy* 2*(intensity + warna)	maxx*(y_{maks}-y_{min})* 2*(intensity+warna)
Inisialisasi Array	maxy + 1) * (maxx + 1) * 2	(y_{maks} - y_{min} + 1) * (maxx + 1) * 2
Jumlah link yang dilalui	1 * jumlah link total	(y_{maks} - y_{min}+1) * jumlah link total
Array yang ditampilkan	(maxy + 1)* (maxx + 1)	(y_{maks} - y_{min} +1)* (maxx + 1)

5. PENGUJIAN

Pada bagian ini ditampilkan hasil program Hidden Surface Removal dari a file data objek dengan extension dir.



Gambar 6. Wire Frame dan HSR (Hidden Surface Removal) untuk File Circular.dir

Tabel 2. Hasil Pengujian dengan Program Circular.dir

No	Nama File Data	Link	Baris	Kolom	Algoritma	Waktu (ms)
1	E:\array test\circular.dir	44	251	300	Scan Line	3,501.00
2	E:\array test\circular.dir	44	251	300	Scan Line	3,504.00
3	E:\array test\circular.dir	44	251	300	Scan Line	4,048.00
4	E:\array test\circular.dir	44	251	300	Scan Line	3,899.00
5	E:\array test\circular.dir	44	251	300	Scan Line	4,058.00
6	E:\array test\circular.dir	44	251	300	Scan Line	3,895.00
7	E:\array test\circular.dir	44	251	300	Scan Line	4,120.00
8	E:\array test\circular.dir	44	251	300	Scan Line	3,987.00
9	E:\array test\circular.dir	44	251	300	Scan Line	4,062.00
10	E:\array test\circular.dir	44	251	300	Scan Line	3,985.00
11	E:\array test\circular.dir	44	251	300	Scan Line	4,021.00
12	E:\array test\circular.dir	44	251	300	Scan Line	3,889.00
13	E:\array test\circular.dir	44	251	300	Scan Line	4,007.00
14	E:\array test\circular.dir	44	251	300	Scan Line	4,192.00
15	E:\array test\circular.dir	44	251	300	Scan Line	4,043.00
16	E:\array test\circular.dir	44	251	300	Scan Line	3,905.00
17	E:\array test\circular.dir	44	251	300	Scan Line	3,900.00
18	E:\array test\circular.dir	44	251	300	Scan Line	3,738.00
19	E:\array test\circular.dir	44	251	300	Scan Line	3,612.00
20	E:\array test\circular.dir	44	251	300	Scan Line	3,754.00
1	E:\array test\circular.dir	44	300	300	Z Buffer	3,844.00
2	E:\array test\circular.dir	44	300	300	Z Buffer	3,897.00
3	E:\array test\circular.dir	44	300	300	Z Buffer	4,480.00
4	E:\array test\circular.dir	44	300	300	Z Buffer	4,486.00
5	E:\array test\circular.dir	44	300	300	Z Buffer	4,372.00
6	E:\array test\circular.dir	44	300	300	Z Buffer	4,524.00
7	E:\array test\circular.dir	44	300	300	Z Buffer	4,329.00
8	E:\array test\circular.dir	44	300	300	Z Buffer	4,415.00
9	E:\array test\circular.dir	44	300	300	Z Buffer	4,463.00
10	E:\array test\circular.dir	44	300	300	Z Buffer	4,377.00
11	E:\array test\circular.dir	44	300	300	Z Buffer	4,353.00
12	E:\array test\circular.dir	44	300	300	Z Buffer	4,422.00
13	E:\array test\circular.dir	44	300	300	Z Buffer	4,379.00
14	E:\array test\circular.dir	44	300	300	Z Buffer	4,533.00
15	E:\array test\circular.dir	44	300	300	Z Buffer	4,376.00
16	E:\array test\circular.dir	44	300	300	Z Buffer	4,350.00
17	E:\array test\circular.dir	44	300	300	Z Buffer	4,353.00
18	E:\array test\circular.dir	44	300	300	Z Buffer	4,106.00
19	E:\array test\circular.dir	44	300	300	Z Buffer	4,064.00
20	E:\array test\circular.dir	44	300	300	Z Buffer	3,983.00

Dari Tabel 2 hasil percobaan, waktu rata-rata yang diperoleh dengan algoritma Scan Line adalah 3.906,00 ms sedangkan untuk Z buffer 4.305,30. Jumlah baris yang diproses oleh *Scan line* (Y_{maks}-Y_{min}) adalah 251 sedangkan algoritma Z Buffer adalah 300 (harga maxy). Jumlah Link yang diproses adalah 44 Link. Hal ini membuktikan bahwa bilamana gambar semakin menyebar dari suatu harga y, maka komponen waktu untuk jumlah *link* yang dilalui akan menjadi berpengaruh, sehingga waktu untuk *Scan line* akan menjadi mendekati waktu dari Z Buffer.

6. KESIMPULAN

1. Algoritma *Scan Line* lebih hemat dalam penggunaan memori dibandingkan dengan algoritma *Z Buffer*.
2. Algoritma *Scan Line* akan membutuhkan waktu lebih sedikit daripada algoritma *Z Buffer*, bila gambar yang dihasilkan mempunyai baris (y_{maks} - y_{min}+1) yang jauh lebih kecil dari maksimum tinggi bidang gambar atau gambar mengumpul untuk daerah y(pada suatu baris) dan hampir membentuk baris, karena hanya menginisialisasi dan menampilkan baris y_{min} *sampai* y_{maks} yang diperlukan saja dan karena (y_{maks} - y_{min}+1) yang kecil dan komponen waktu yang diperlukan untuk melalui suatu rangkaian *link* sebanyak (y_{maks} - y_{min}+1) akan menjadi kurang berpengaruh.
3. Waktu untuk kedua algoritma akan sama bila waktu inisialisasi (y_{maks} - y_{min} + 1) + *waktu tampil* (y_{maks} - y_{min} + 1) + *waktu untuk* ((y_{maks} - y_{min} + 1) * *suatu rangkaian link*) dari Scan Line sama dengan waktu *inisial bidang gambar + waktu menampilkan buffer + (1* suatu rangkaian link)* dari Z Buffer.
4. Bilamana gambar semakin menyebar dari suatu harga y, maka waktu yang diperlukan *Scan Line* akan semakin bertambah hingga pada akhirnya menjadi sama dengan algoritma Z Buffer sedangkan gambar yang semakin menyebar akan menyebabkan komponen waktu untuk jumlah *link* yang dilalui akan menjadi

bertambah besar, sehingga bilamana gambar tersebut mempunyai daerah yang besarnya sama dengan maksimum dari tinggi bidang gambar maka *Scan Line* akan menjadi lebih lama dari Z Buffer karena *Scan Line* akan melewati *link* berulang-ulang sedangkan Z Buffer hanya akan melewati suatu rangkaian *Link* sebanyak satu kali.

DAFTAR PUSTAKA

1. Ammeraal, Leendert, *Programming Principles in Computer Graphic*, Chichester: Wiley Professional Computing, 1992.
2. Sproull, Robert F., *Device Independent Graphics,* Singapore, Mc Graw Hill Book Co.,1989.
3. Rogers, David F., *Mathematical Elements for Computer Graphic*, NewYork: Mc-Graw Hill,1990.
4. Wolfram, Stephen, *Mathematica, A System for Doing Mathematics by Computer*, Redwood City, California: Addison Wesley Publishing Co, 1991.