BAB V

HASIL DAN PEMBAHASAN

5.1. Penentuan Metode Deteksi Tabrakan

Untuk menentukan metode deteksi tabrakan yang lebih baik dibandingkan dengan AABB, penelitian ini melakukannya berdasarkan hasil penelitian tahun sebelumnya. Pada penelitian tahun sebelumnya terdapat dua batasan yang harus diperhitungkan dalam menentukan metode deteksi tabrakan objek.

Batasan I:

Penambahan perhitungan pada CVE sangat berpengaruh pada *runtime* dan *frame rate* dari aplikasi. Ini akan menyebabkan performa aplikasi CVE akan sangat buruk, bahkan tidak bisa digunakan.

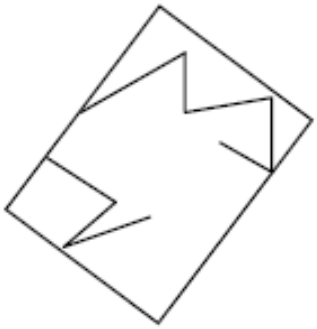
Tabel 5.1. Analisa Beberapa Metode Deteksi Tabrakan

Pengujian	**AABB**	**OBB**	**HPCCD**	**MCCD**
Keakuratan	Kurang akurat	Lebih akurat	Sangat akurat	Sangat akurat
Environment	CVE	CPU	Multicore	Multicore
Fleksibilitas	Sangat fleksibel	Sangat fleksibel	Kurang fleksibel	Kurang fleksibel
Penggunaan Memori	Sangat Kecil	Kecil	Besar	Besar

Batasan II:

Beberapa metode deteksi tabrakan objek yang lebih baik dari AABB seperti Oriented Bounding Box (OBB), HPCCD, dan MCCD belum digunakan pada CVE. Setiap metode memiliki kelebihan dan kekurangan. Karakteristik dari metode tersebut dapat dilihat pada Tabel 5.1.

Berdasarkan Batasan I dan Batasan II penggunaan metode deteksi tabrakan OBB lebih memungkinkan untuk digunakan pada CVE dibandingkan dengan metode yang lainnya. Hal ini didasarkan pada kompleksitas dan fleksibilitas dari metode tersebut yang lebih baik dibandingkan dengan yang lainnya.



Gambar 5.1. *Oriented Bounding Box*

5.2. Metode OBB

Oriented Bounding Box (OBB) adalah suatu kotak terkecil yang memiliki koordinat sembarang yang memuat suatu objek yang diberikan. Metode OBB lebih akurat dibandingkan dengan AABB. Misalkan bahwa pusat OBB adalah *c*. Tiga sumbu yang saling tegak lurus adalah v^1, v^2, v^3 dan "jari-jari" dalam arah tiga sumbu tersebut adalah r^1, r^2, r^3. Jadi, daerah definitif dari OBB adalah:

$$R = \{c + ar^1v^1 + br^2v^2 + cr^3v^3 | a, b, c \in (-1,1)\}$$

seperti ditunjukkan oleh Gambar 5.1.

Kelebihan dari OBB adalah arah dari koordinat yang berubah-ubah yang disesuaikan dengan posisi objek. Ini membuatnya dapat melingkupi objek seketat mungkin sesuai dengan bentuk objek tersebut. Tentunya ini berbeda dengan AABB yang memiliki koordinat yang tetap sehingga akurasi dari proses deteksi objek menjadi lemah.

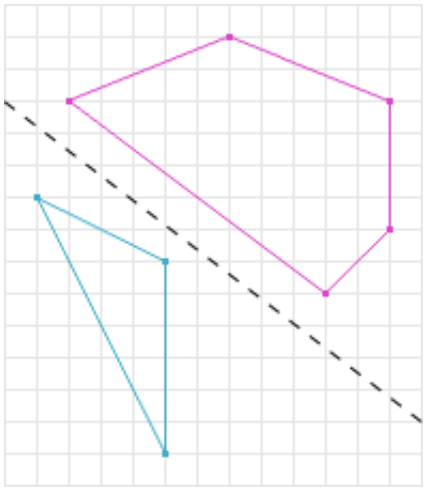
Namun sangat disayangkan bahwa perhitungan OBB adalah relatif lebih kompleks. Faktor kuncinya adalah untuk menemukan arah yang terbaik dan menentukan ukuran terkecil dari *bounding box* yang melingkupi objek dalam arah tersebut.

Salah satu cara yang digunakan untuk menentukan OBB adalah *Separated Axis Theorem* (SAT).

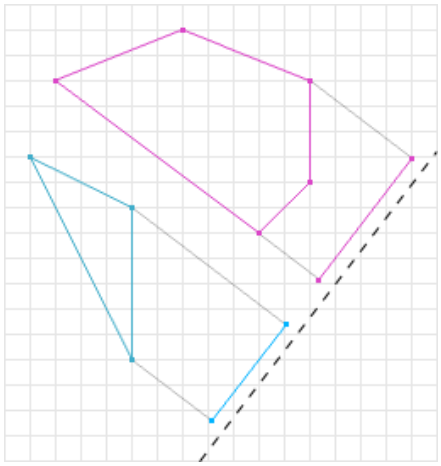
Teorema (SAT): Jika dua objek konveks tidak bertabrakan maka terdapat suatu sumbu dimana proyeksi dari kedua objek tersebut tidak akan tumpang tindih.

Gambar 5.2 mengilustrasikan dua bangun yang tidak saling berpotongan. Sebuah garis digambarkan diantara keduanya untuk menunjukkan hal ini. Jika dipilih garis yang tegak lurus terhadap garis tersebut, kemudian memproyeksikan bangun-bangun tersebut pada garis tadi, maka dapat dilihat bahwa tidak terdapat tumpang tindih pada hasil proyeksi tersebut. Suatu garis dimana proyeksi bangun-

bangun tersebut tidak tumpang tindih disebut sebagai sumbu pemisah (*separation axis*).



Gambar 5.2. Dua bangun konveks yang terpisah



Gambar 5.3. Proyeksi dari kedua bangun

Pada Gambar 5.3, garis putus-putus menyatakan suatu garis pemisah. Karena semua proyeksi yang didapatka tidak tumpang tindih maka menurut teorema SAT bangun-bangun tersebut tidak berpotongan.

5.3. Analisa Aplikasi CVE

Aplikasi CVE yang dikembangkan memungkinkan beberapa user secara bersama-sama berkolaborasi menggunakannya. Seorang user dapat menggunakan aplikasi tersebut dari sebuah komputer yang terpisah dengan user yang lainnya. Oleh sebab itu, setiap user yang mengggunakan CVE akan terhubung kedalam sebuah jaringan komputer.

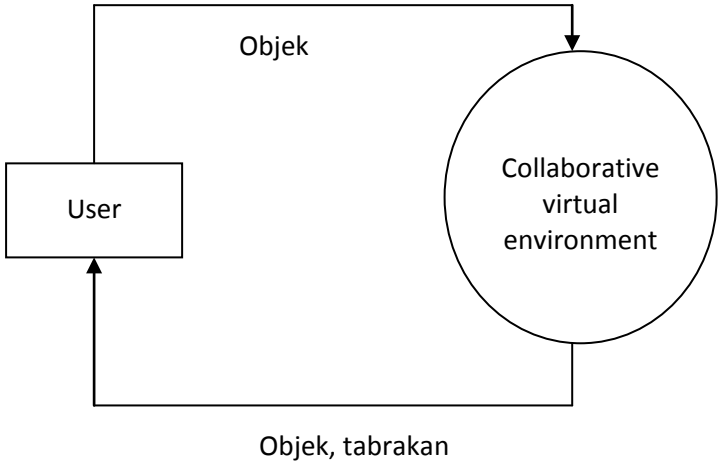
Menggunakan aplikasi CVE yang sama, seorang user dapat membuat satu atau lebih objek yang diinginkannya didalam *virtual environment* (VE). Objek yang dapat dibuat oleh seorang user adalah objek dinamis yang dapat berpindah posisi didalam VE. Objek yang dibuat oleh seorang user tersebut akan dapat dilihat pula oleh setiap user yang lainnya. Dalam hal ini setiap user akan memiliki tampilan antar muka aplikasi yang sama dalam suatu waktu, baik dalam hal apa yang ditampilkan maupun dalam hal berapa jumlah objek yang ditampilkan

Karena terdapat banyak objek didalam VE maka kemungkinan terjadinya tabrakan antar objek akan sangat besar. Oleh karena itu VE harus memiliki metoda yang dapat menangani tabrakan antar objek tersebut. Untuk ini digunakan Metode *Oriented Bounding Box* (OBB). Satu hal penting lainnya adalah bahwa VE yang dibuat harus merepresentasikan kondisi ril yang terjadi. Sebagai contoh, VE harus memiliki gaya gravitasi. Kemudian, ketika suatu objek bertabrakan

tentunya hasil dari tabrakan tersebut akan merubah posisi dan bentuk dari objek-objek itu sendiri.

5.4. Rancangan CVE

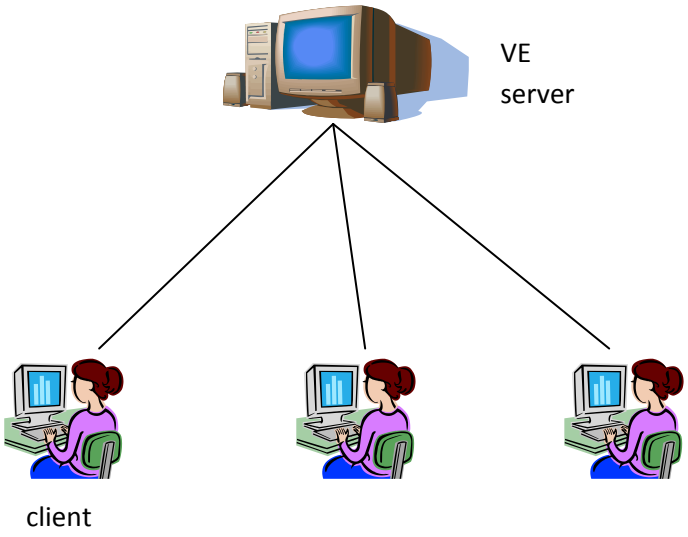
Interaksi antara user dengan aplikasi CVE dapat diilustrasikan pada Gambar 5.4.



Gambar 5.4. Interaksi user dan aplikasi CVE

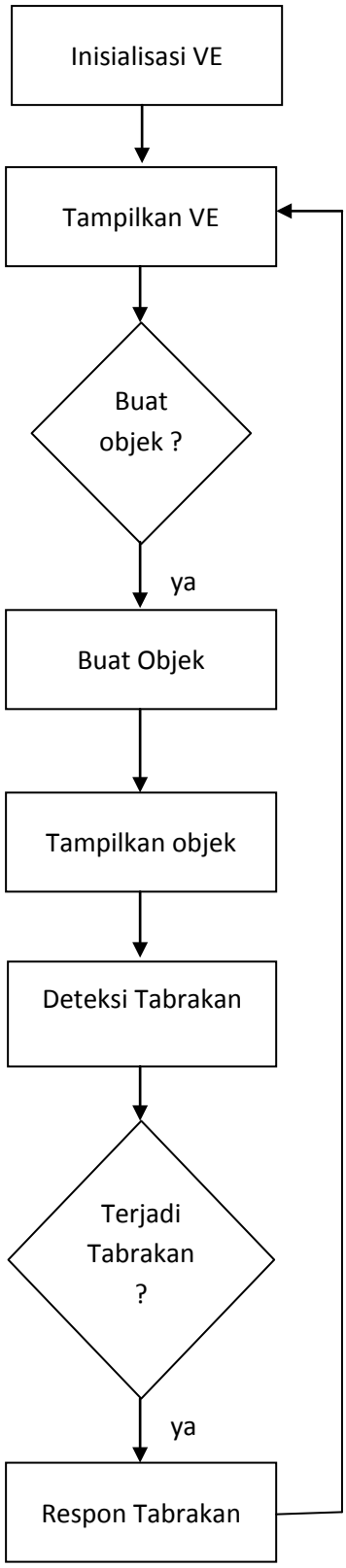
Seorang user atau *client* dapat membuat objek pada VE. Kemudian user dapat melihat objek-objek apa saja yang ada pada VE, dan sekaligus melihat tabrakan yang dapat terjadi pada objek-objek tersebut.

Oleh karena aplikasi ini dapat digunakan oleh banyak user, maka implementasinya dapat menggunakan topologi *client-server* seperti pada Gambar 5.5.



Gambar 5.5. Model jaringan yang digunakan

Selanjutnya, rancangan program dari aplikasi CVE ini dijelaskan pada Gambar 5.6. Tahap inisialisasi adalah tahap awal untuk membuat lingkungan virtual yang diinginkan. Pada tahap ini VE terdiri dari dua komponen yaitu langit dan *ground*. Setelah VE ditampilkan dalam bentuk tiga dimensi (3D), user dapat merubah sudut pandang terhadap VE dengan melakukan dua hal: *pan* and *tilt*, serta geser kiri-kanan dan atas-bawah.



Gambar 5.6. Rancangan program CVE

Tahap deteksi tabrakan adalah tahap dimana metoda OBB digunakan. Ketika terjadi tabrakan antar objek, hasil dari tabrakan tersebut tentunya akan merubah posisi dan bentuk objek yang bertabrakan. Penanganan pasca-tabrakan ini dilakukan oleh tahap respon tabrakan.

5.5. Kode Program Metode OBB

Kode program untuk metode OBB menggunakan SAT dapat dilihat pada Lampiran A. Dalam hal ini terdapat 15 sumbu pemisah (*separating axis*) yang mungkin untuk diperiksa. Cek apakah sumbu tersebut memisahkan kotak-kotak. Apabila terdapat satu sumbu saja yang memisahkan kotak-kotak tersebut maka sumbu pemisah yang lainnya tidak perlu diperiksa lagi. Artinya kotak-kotak itu tidak bertabrakan sesuai dengan teorema SAT.

5.6. Uji Program

Pengujian dari program dilakukan secara *black box*. Hasil ujinya dapat dilihat pada Tabel 5.2.

Tabel 5.2. Hasil Uji Program

No	Pengujian	Hasil Uji
1.	Tampilan awal *environment*	Sukses
2.	Titik koordinat penanda pada *ground*	Sukses
3.	Pembuatan objek	Sukses
4.	Gravitasi	Sukses
5.	Geser kiri	Sukses
6.	Geser kanan	Sukses

No	Pengujian	Hasil Uji
7.	*Pan-tilt*	Sukses
8.	Posisi *viewer*	Sukses
9.	Deteksi tabrakan	Sukses
10.	Reaksi tabrakan	Sukses
11.	Perubahan ukuran dan massa objek	Sukses

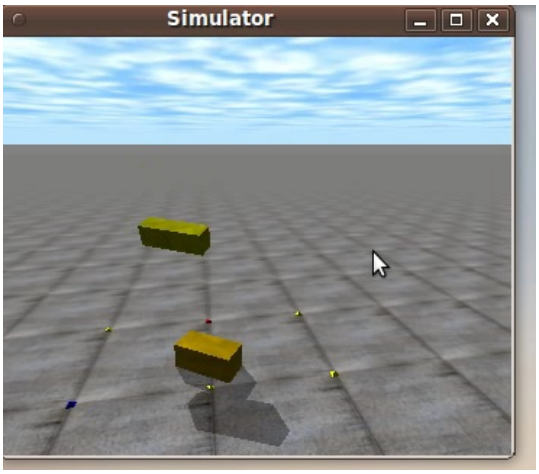
5.7. Implementasi OBB pada CVE

Implementasi OBB pada CVE dilakukan menggunakan dua skenario, yaitu:

1. OBB diterapkan pada CVE menggunakan arsitektur satu simulator
2. OBB diterapkan pada CVE menggunakan arsitektur simulator berbasis objek.

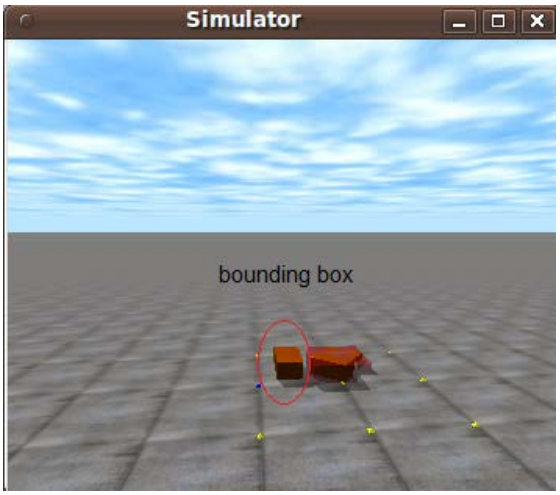
Gambar 5.7. Tampilan awal CVE

Tampilan awal dari DVE dapat dilihat pada Gambar 5.7. Seorang user dapat menggunakan antarmuka ini untuk melihat environment dan membuat objek. Gambar 5.8 menunjukkan beberapa objek yang dibuat oleh user. Objek yang dibuat berupa kotak dengan ukuran dan posisi berbeda yang jatuh dari ketinggian tertentu.



Gambar 5.8. Objek yang dibuat oleh user

Selanjutnya OBB digunakan oleh aplikasi untuk mendeteksi tabrakan antar objek. Gambar 5.9 menunjukkan *bounding box* yang dihasilkan oleh aplikasi. Dari gambar ini dapat dilihat bahwa *bounding box* dari objek (lingkaran merah) melingkupi objek dengan ketat sekali sehingga sulit untuk membedakan mana yang merupakan *bounding box* dan objek itu sendiri. Tidak ada ruang kosong antara keduanya.



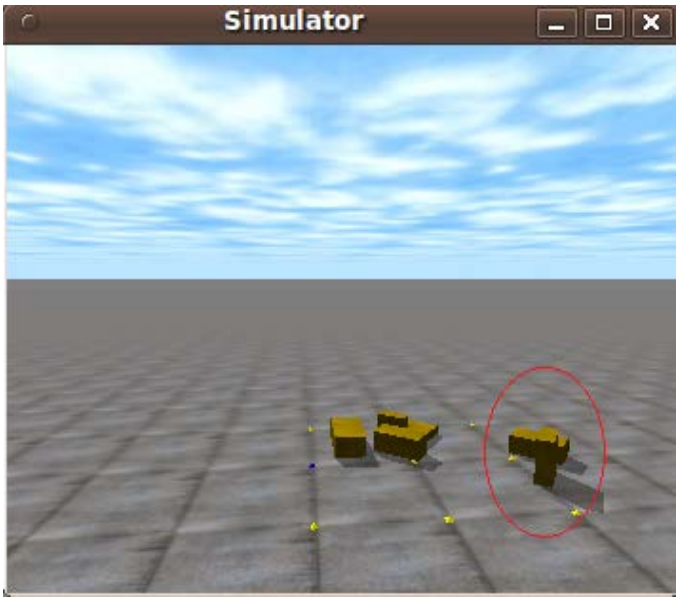
Gambar 5.9. *Bounding box* dari objek

5.8. Hasil Evaluasi

Untuk menentukan keakuratan dari metode deteksi tabrakan objek, pengukuran *missing* dilakukan terhadap metode AABB dan OBB. Tabel 5.3 menunjukkan hasil yang didapatkan.

Tabel 5.3. Hasil pengukuran *missing*

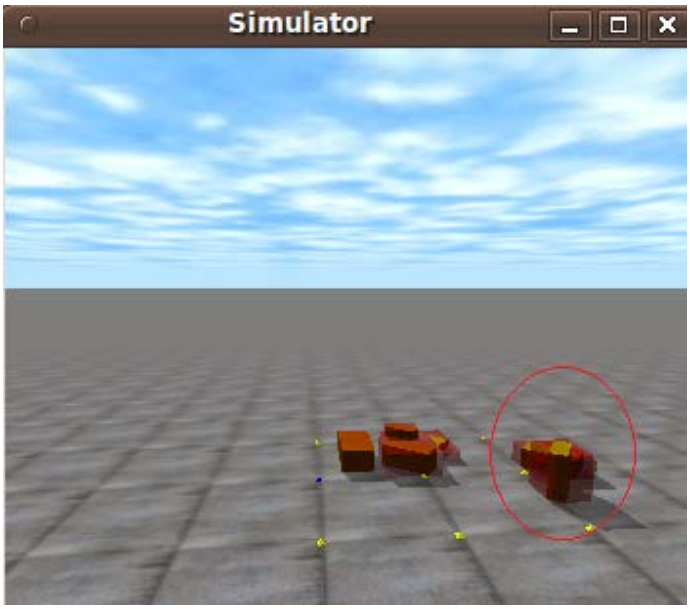
No	Jumlah Objek	Jumlah *missing*	
		AABB	OBB
1	10	0	0
2	20	2	0
3	30	3	0
4	40	5	0
5	50	8	0



Gambar 5.10. Dua objek yang tidak bertabrakan

Terjadinya *missing* dalam deteksi tabrakan objek pada Tabel 5.3 disebabkan karena *bounding box* yang dibentuk selalu sejajar dengan sumbu koordinat, padahal posisi atau bentuk objek yang dilingkupinya tidak demikian. Sebagai contoh, perhatikan Gambar 5.10 yang memiliki dua objek yang tidak bertabrakan (bagian kanan gambar yang dilingkari). Ketika untuk kedua objek ini dibuatkan *bounding box* yang sejajar dengan sumbu koordinat (metode AABB), maka hasil deteksi tabrakannya adalah bahwa kedua objek tersebut bertabrakan. Padahal, kalau menggunakan OBB kedua objek tersebut adalah tidak bertabrakan.

Dengan demikian, hasil pengukuran missing metode OBB adalah lebih kecil dibandingkan dengan AABB. Hal ini menunjukkan bahwa OBB lebih akurat dibandingkan dengan AABB.



Gambar 5.11. *Bounding box* pada AABB

Hasil *runtime* dari Skenario 1 dapat dilihat pada Tabel 5.4. Tabel ini menunjukkan hasil pengukuran *runtime* aplikasi untuk 20 kali perulangan simulasi (*simulation loop*) untuk banyaknya objek yang bervariasi mulai dari 10 sampai dengan 80 objek.

Tabel 5.4. *Runtime* aplikasi Skenario 1 (dalam detik)

No.	10 obj	20 obj	30 obj	40 obj	50 obj	60 obj	70 obj	80 obj
1	0.01	0.01	0	0.01	0.01	0.03	0.02	0.02
2	0	0	0.01	0	0.02	0.02	0.02	0.02
3	0	0	0	0.01	0.02	0.01	0.02	0.02
4	0	0.01	0.01	0.01	0.01	0.02	0.02	0.02
5	0.01	0	0	0.01	0.01	0.02	0.02	0.02
6	0	0.01	0.01	0.01	0.02	0.02	0.02	0.02

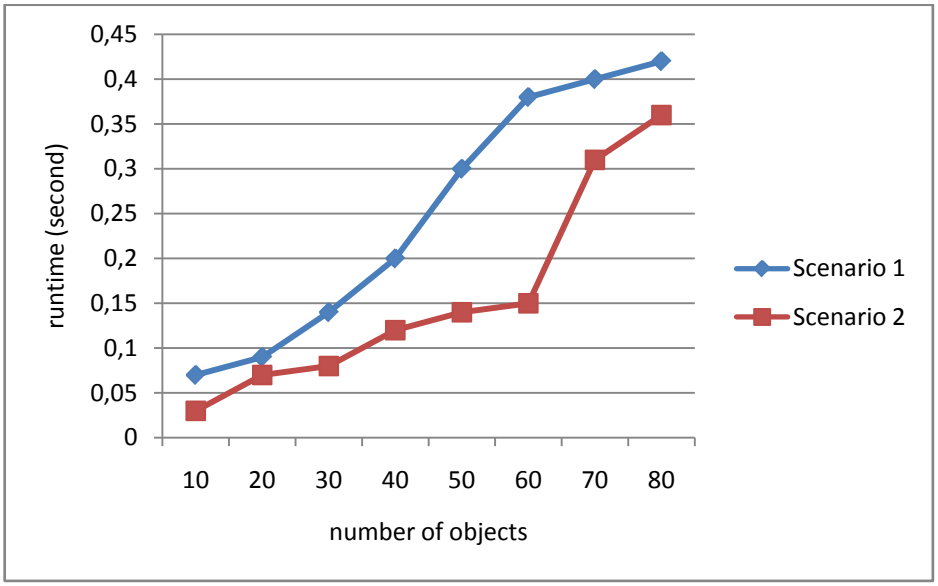
No.	10 obj	20 obj	30 obj	40 obj	50 obj	60 obj	70 obj	80 obj
7	0	0	0.01	0.01	0.01	0.01	0.02	0.02
8	0.01	0.01	0.01	0.01	0.02	0.02	0.02	0.02
9	0	0.01	0.01	0.01	0.01	0.02	0.02	0.02
10	0	0	0.01	0.01	0.02	0.02	0.02	0.02
11	0.01	0	0.01	0.01	0.02	0.02	0.02	0.02
12	0	0.01	0	0.01	0.01	0.02	0.02	0.02
13	0	0	0.01	0.02	0.02	0.01	0.02	0.02
14	0	0.01	0.01	0.01	0.01	0.02	0.02	0.03
15	0.01	0	0.01	0.01	0.01	0.02	0.02	0.02
16	0	0	0.01	0.01	0.02	0.02	0.02	0.02
17	0.01	0.01	0.01	0.01	0.02	0.02	0.02	0.02
18	0	0	0	0.01	0.01	0.02	0.02	0.03
19	0	0.01	0.01	0.01	0.02	0.02	0.02	0.02
20	0.01	0	0	0.01	0.01	0.02	0.02	0.02
Jlh	**0.07**	**0.09**	**0.14**	**0.2**	**0.3**	**0.38**	**0.4**	**0.42**

Selanjutnya, *runtime* aplikasi dari Skenario 2 dengan perlakuan yang sama dengan Skenario 1 dapat dilihat pada Tabel 5.5.

Tabel 5.5. *Runtime* aplikasi Skenario 2 (dalam detik)

No.	10 obj	20 obj	30 obj	40 obj	50 obj	60 obj	70 obj	80 obj
1	0	0.01	0.01	0.01	0.01	0.01	0.01	0.02
2	0	0	0	0	0	0	0.02	0.01
3	0	0	0	0.01	0.01	0	0.01	0.02
4	0	0	0	0	0	0.01	0.02	0.02
5	0	0	0.01	0.01	0	0.01	0.01	0.02
6	0	0.01	0	0	0.01	0.01	0.02	0.02
7	0	0	0	0.01	0.01	0.01	0.02	0.02
8	0	0	0	0.01	0.01	0.01	0.01	0.02
9	0.01	0.01	0.01	0	0.01	0.01	0.02	0.01
10	0	0	0	0.01	0.01	0	0.01	0.02
11	0	0	0.01	0	0.01	0.01	0.02	0.02
12	0	0.01	0	0.01	0.01	0.01	0.01	0.02
13	0	0	0	0	0	0.01	0.02	0.02
14	0	0.01	0.01	0.01	0	0	0.02	0.01
15	0.01	0	0	0.01	0.01	0.01	0.01	0.02
16	0	0	0.01	0	0.01	0.01	0.02	0.02
17	0	0.01	0	0.01	0.01	0.01	0.01	0.02
18	0	0	0.01	0	0.01	0.01	0.02	0.01
19	0.01	0.01	0.01	0.01	0	0	0.01	0.02
20	0	0	0	0.01	0.01	0.01	0.02	0.02
Jlh	**0.03**	**0.07**	**0.08**	**0.12**	**0.14**	**0.15**	**0.31**	**0.36**

Perbedaan dari kedua skenario tersebut dapat dilihat pada Gambar 5.12 dimana setiap titik menyatakan nilai rata-rata *runtime* untuk setiap banyaknya objek. Berdasarkan gambar ini dapat dikatakan bahwa *runtime* yang diperlukan oleh Skenario 2 adalah lebih kecil dibandingkan dengan Skenario 1. Ini terjadi pada setiap banyak objek yang berbeda dan terdapat perbedaan yang sangat signifikan antara kedua skenario tersebut.



Gambar 5.12. *Runtime* aplikasi kedua skenario

Hasil pengukuran *frame rate* aplikasi untuk Skenario 1 ditunjukkan oleh Tabel 5.6. Tabel ini berisikan jumlah *frame* untuk waktu lima detik, sehingga enam nilai yang ada pada tabel tersebut diambil dalam waktu 30 detik. Rata-rata dari *frame rate* aplikasi ini dapat dilihat pada baris terakhir pada tabel. Nilai *frame rate* ini diukur untuk banyak objek yang bervariasi mulai dari 10 sampai dengan 80.

Tabel 5.6. *Frame rate* aplikasi Skenario 1

No	10 obj	20 obj	30 obj	40 obj	50 obj	60 obj	70 obj	80 obj
1	1273	1223	1230	1035	933	689	760	676
2	1295	1229	1225	924	862	650	857	700
3	1302	1228	1083	758	774	789	665	770
4	1334	1260	968	715	740	900	701	674
5	1166	1082	855	723	605	829	900	826
6	1233	1011	963	715	640	769	823	671
Avg	1267.167	1172.167	1054	811.667	759	771	784.333	719.5

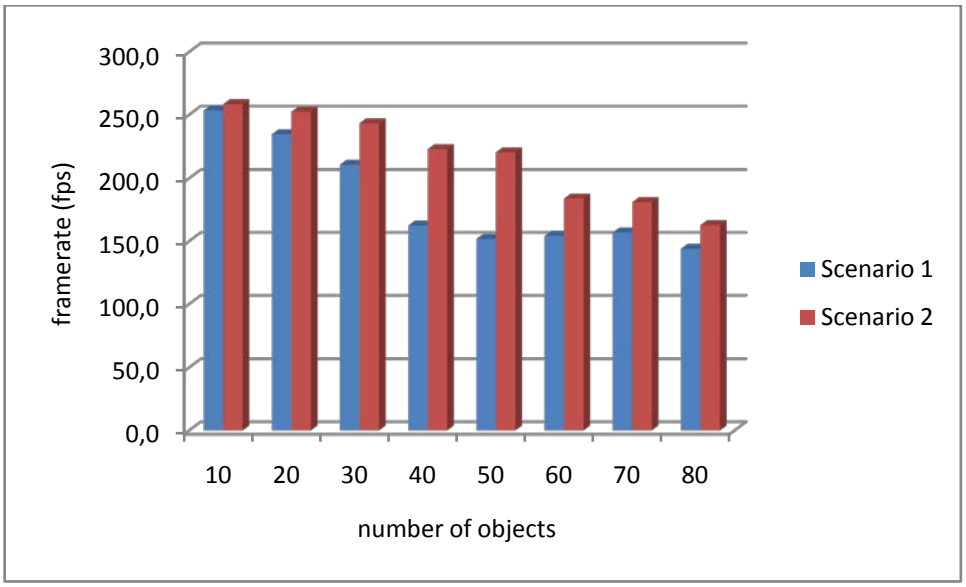
Selanjutnya Tabel 5.7 menunjukkan *frame rate* aplikasi Skenario 2 untuk jumlah *frame* dalam lima detik selama 30 detik. Nilai *frame rate* ini sama dengan Skenario 1 yaitu diukur untuk banyak objek yang berbeda.

Tabel 5.7. *Frame rate* aplikasi Skenario 2

No	10 obj	20 obj	30 obj	40 obj	50 obj	60 obj	70 obj	80 obj
1	1374	1337	1220	1156	1226	1168	1049	992
2	1322	1382	1259	1223	1228	1035	1093	906
3	1286	1269	1264	1192	1197	856	952	834
4	1221	1230	1305	1148	970	831	830	761
5	1305	1148	1106	969	995	794	711	728
6	1243	1205	1143	1006	1002	832	794	660
avg	**1291.833**	**1261.833**	**1216.167**	**1115.667**	**1103**	**919.333**	**904.833**	**813.5**

Perbedaan dari rata-rata *frame rate* untuk kedua skenario ini dalam satuan *frame per second* (fps) dapat dilihat pada Gambar 5.13. Berdasarkan gambar, *frame rate* dari skenario kedua lebih besar dari pada skenario pertama. Artinya, performa dari skenario kedua lebih baik bila dibandingkan dengan skenario pertama.

Berdasarkan Gambar 5.12 dan Gambar 5.13, terlihat bahwa skenario kedua memiliki *runtime* yang lebih kecil dan *frame rate* yang lebih besar bila dibandingkan dengan skenario pertama. Bahkan, kedua nilai parameter yang dimiliki oleh skenario kedua tersebut lebih baik dibandingkan dengan AABB. Dengan demikian, jelas terlihat bahwa faktor akselerasi dapat dipenuhi oleh skenario kedua.



Gambar 5.13. *Frame rate* kedua skenario