BAB 2
LANDASAN TEORI

2.1. Fisika

Fisika merupakan salah satu pilar utama ilmu pengetahuan dan teknologi yang memberikan pemahaman mengenai fenomena alam serta kemungkinan aplikasinya dalam meningkatkan kesejahteraan hidup umat manusia [1].

2.1.1 Momentum Dan Impuls

A. Pengertian Momentum

Momentum didefinisikan sebagai ukuran kesukaran untuk memberhentikan suatu benda ,momentum diperoleh dari hasil kali besaran skalar massa dengan besaran vector kecepatan [2]. Secara matematis dapat dituliskan sebagai berikut :

$$p = m.v \qquad \text{(Persamaan 1)}$$

Keterangan

- p = momentum(kg.m/s)
- m = massa(kg)
- v =kecepatan(m/s)

B. Pengertian Impuls

Impuls adalah peristiwa gaya yang bekerja pada benda dalam waktu hanya sesaat. Atau Impuls adalah peristiwa bekerjanya gaya dalam selang waktu yang sangat singkat [2]. Contoh dari kejadian impuls adalah: peristiwa seperti bola ditendang, bola tenis dipukul karena pada saat tendangan dan pukulan, gaya yang bekerja sangat singkat.

$$I = F.\Delta t \qquad \text{(Persamaan 2)}$$

Keterangan

- I = impuls
- F =gaya(N)
- Δt = selang waktu(s)

C. Hubungan Impuls dan Momentum

Impuls yang dikerjakan pada suatu benda sama dengan perubahan momentum yang dialami benda itu, yaitu beda antara momentum akhir dengan momentum awal [2].

$$I = p2 - p1 = \Delta p$$

$$F.\ \Delta t = \Delta p$$

$$F = \Delta p / \Delta t \qquad \text{(Persamaan 3)}$$

Artinya, gaya (F) yang diberikan pada suatu benda sama dengan perubahan momentum (Δp) benda per satuan waktu Δt.

2.1.2 Tumbukan

Tumbukan adalah pertemuan dua benda yang relatif bergerak. Pada setiap jenis tumbukan berlaku hukum kekekalan momentum tetapi tidak selalu berlaku hukum kekekalan energi mekanik. Sebab disini sebagian energi mungkin diubah menjadi panas akibat tumbukan atau terjadi perubahan bentuk. Tumbukan terdapat perbedaan jenis berdasarkan nilai koefisien elastisitasnya (koefisien restitusi = e). Koefisien elastisitas (e) dari dua buah benda yang bertumbukan didefinisikan sebagai harga negatif dari perbandingan antara beda kecepatan kedua benda yang bertumbukan sesaat sesudah tumbukan dan sesaat sebelum tumbukan. Secara matematis ditulis :

$$e = \frac{-\Delta v'}{\Delta v} = \frac{-(v_2' - v_1')}{v_2 - v_1} \qquad \text{(Persamaan 4)}$$

Nilai koefisien elastisitas (e) terbatas, yaitu $0 < e < 1$

2.1.2.1. Tumbukan Lenting Sempurna

Seorang pemain biliar memukul bola biru secara perlahan tanpa memberi efek putaran menuju bola merah yang diam. Bola biru kemudian menumbuk bola merah. Sesaat sesudah tumbukan bola putih menjadi diam dan bola merah bergerak dengan kecepatan yang hampir sama dengan kecepatan datangnya bola biru.

Peristiwa tumbukan antara bola biru (1) dan bola merah (2) dapat dilukiskan pada diagram Gambar 2.1.



Gambar 2.1 Tumbukan antara bola biru dan biru

Asalkan gaya luar yang bekerja pada sistem diabaikan, maka kekekalan momentum berlaku pada tumbukan ini. Karena bola merah (bola 2) diam sebelum tumbukan dan bola biru (bola 1) diam sesudah tumbukan, sedangkan massa kedua bola sama, maka kecepatan bola 2 sesudah tumbukan pastilah sama dengan kecepatan bola 1 sebelum tumbukan, yaitu *v*. Dalam kasus tumbukan ini seakan-akan momentum bola 1 dialihkan seluruhnya ke momentum bola 2. Bagaimana energi kinetiknya? Energi kinetik sebelum tumbukan yaitu energi kinetik bola 1, $\frac{1}{2}mv^2$, ternyata juga sama dengan energi kinetik sesudah tumbukan, yaitu energi kinetik bola 2, $\frac{1}{2}mv^2$. Jadi, dalam kasus tumbukan ini seakan-akan energi kinetik bola 1 juga dialihkan seluruhnya ke energi kinetik bola 2.

Dalam peristiwa tumbukan dua bola biliar seperti ditunjukkan pada Gambar 1, selain momentum sistem tetap, energi kinetik sistem juga tetap. Jenis tumbukan dimana berlaku kekekalan momentum dan kekekalan energi kinetik, kita sebut t*umbukan lenting sempurna*.

Perhatikan dua benda bermassa m_1 dan m_2 yang sedang bergerak saling mendekat dengan kecepatan v_1 dan v_2 sepanjang suatu garis lurus, seperti ditunjukkan pada Gambar 2.2.



Gambar 2.2 Tumbukan lenting sempurna antara dua bola keras

Keduanya bertumbukan lenting sempurna dan kecepatan masing-masing sesudah tumbukan adalah v_1' dan v_2'. Pada tumbukan lenting sempurna berlaku hukum kekekalan momentum dan hukum kekekalan energi kinetik.

$$p_1 + p_2 = p_1' + p_2'$$

$$m_1 v_1 + m_2 v_2 = m_1 {v_1}' + m_2 {v_2}' \qquad \text{(Persamaan 5)}$$

dan

$$EK_1 + EK_2 = {EK_1}' + {EK_2}'$$

$$\tfrac{1}{2} m_1 {v_1}^2 + \tfrac{1}{2} m_2 {v_2}^2 = \tfrac{1}{2} m_1 {v_1}'^2 + \tfrac{1}{2} m_2 {v_2}'^2 \text{ (Persamaan 6)}$$

Dari kedua persamaan tersebut, dapat diketahui hubungan antara v_1, v_2, v_1', dan v_2' adalah sebagai berikut:

$$m_1(v_1 - {v_1}') = m_2({v_2}' - v_2)$$

Dan

$$m_1({v_1}^2 - {v_1}'^2) = m_2({v_2}'^2 - {v_2}^2) \qquad \text{(Persamaan 7)}$$

Dengan mengingat rumus aljabar $a^2 - b^2 = (a + b)(a - b)$, persamaan (3) dapat ditulis menjadi:

$$m_1(v_1 - {v_2}')(v_1 + {v_2}') = m_2({v_2}' - v_2)({v_2}' + {v_2}') \quad \text{(Persamaan 8)}$$

Jika persamaan (4) dibagi persamaan (3), diperoleh

$$v_1 + {v_1}' = {v_2}' + v_2$$

Atau

$$v_1 - v_2 = -({v_1}' - {v_2}') \qquad \text{(Persamaan 9)}$$

Pada persamaan (5) ini, $v_1 - v_2$ menyatakan kecepatan relatif dua benda sebelum tumbukan, sedangkan ${v_1}' - {v_2}'$ menyatakan kecepatan relatif dua benda setelah tumbukan. Jadi, untuk tumbukan lenting sempurna sepusat (seluruh gerakannya terletak pada satu garis lurus), kelajuan relatif kedua benda setelah tumbukan sama dengan kelajuan relative sebelum tumbukan, tidak terpengaruh massa benda yang bertumbukan.

2.1.2.2. Tumbukan Lenting Sebagian

Pada umumnya dua buah benda yang bertumbukan, sebagian besar mengalami tumbukan lenting sebagian. Sebagai contoh, bola tenis atau bola kasti yang dilepas dari ketinggian h_1 di atas lantai akan terpental setinggi h_2, dimana h_2 selalu lebih kecil dari h_1.

Pada pembahasan tumbukan lenting sempurna, pada persamaan 5 diperoleh

$$\Delta v' = -\Delta v\text{e}$$

atau

$$\frac{-\Delta v'}{\Delta v} = 1$$

Rasio $\frac{-\Delta v'}{\Delta v}$ inilah yang didefinisikan sebagai koefisien restitusi.

Koefisien restitusi (diberi lambang *e*) adalah negatif perbandingan antara kecepatan relatif sesaat sesudah tumbukan dengan kecepatan relatif sesaat sebelum tumbukan.

$$e = \frac{-\Delta v'}{\Delta v} = \frac{-(v_2{}' - v_1{}')}{v_2 - v_1}$$

Nilai koefisien restitusi adalah terbatas, yaitu antara nol dan satu ($0 \leq e \leq 1$).

Untuk tumbukan lenting sempurna:

$$e = \frac{-\Delta v'}{\Delta v} = 1$$

Untuk tumbukan tak lenting sama sekali:

$$e = \frac{-\Delta v'}{\Delta v} = \frac{-(v_2{}' - v_1{}')}{v_2 - v_1} = 0$$

sebab $v_2{}' = v_1{}'$

Seperti telah disebutkan bahwa sebagian besar tumbukan adalah tumbukan lenting sebagian, yaitu tumbukan yang berada di antara dua keadaan ekstrem tumbukan lenting sempurna dan tumbukan tak lenting sama sekali. Jelaslah bahwa pada tumbukan lenting sebagian, koefisien restitusi adalah $0 < e < 1$, misalnya $e = \frac{1}{2}$, $e = \frac{1}{3}$, dan $e = 0{,}6$.

2.1.2.3 Tumbukan Tidak Lenting Sama Sekali

Segumpal plastisin dilemparkan dalam arah mendatar menuju ke sebuah bola bilyar yang diam di atas lantai licin. Plastisin tersebut menumbuk sentral bola dan sesaat setelah tumbukan, plastisin menempel pada bola bilyar dan keduanya kemudian bergerak bersama dengan kecepatan sama. Ini adalah contoh tumbukan tidak lenting sama sekali. *Jadi, untuk tumbukan tidak lenting sama sekali, sesaat setelah tumbukan kedua benda bersatu dan bergerak bersama dengan kecepatan yang sama*. Contoh lain dari tumbukan tidak lenting sama sekali adalah pada ayunan balistik di mana peluru tertanam dalam sebuah balok, dan keduanya kemudian mengalami suatu gerak ayunan.

Karena pada tumbukan tak lenting sama sekali kedua benda bersatu sesudah tumbukan, maka berlaku hubungan kecepatan sesudah tumbukan yaitu:

$v_1' = v_2' = v'$ (Persamaan 10)

Persamaan (6) dimasukkan dalam persamaan hukum kekekalan momentum:

$$m_1 v_1 + m_2 v_2 = m_1 v_1' + m_2 v_2'$$

sehingga persamaannya menjadi:

$$m_1 v_1 + m_2 v_2 = (m_1 + m_2) v' \qquad \text{(Persamaan 11)}$$

Apabila benda bermassa m_1 mula-mula bergerak dengan kecepatan v_1 dan benda bermassa m_2 mula-mula diam, maka persamaannya menjadi:

$$m_1 v_1 + m_2 . 0 = (m_1 + m_2) v' \qquad \text{(Persamaan 12)}$$

atau

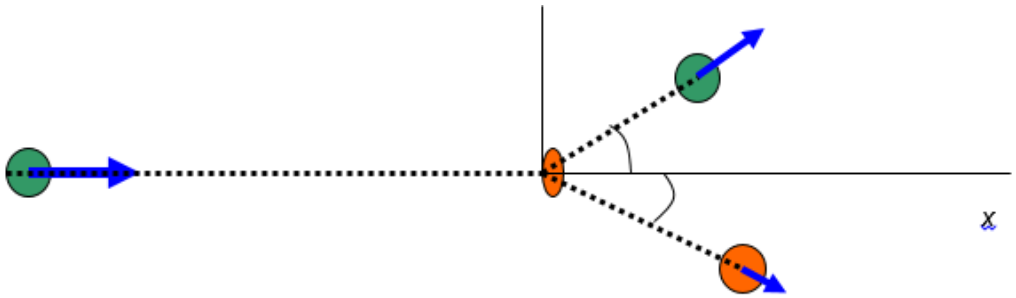
$$v' = \frac{m_1}{m_1 + m_2} v_1$$

Jadi, dengan mengetahui massa dan kecepatan mula-mula, dapat dihitung kecepatan benda setelah tumbukan.

2.1.3 Tumbukan Pada Dua Dimensi

Kekekalan momentum dan energi juga bisa diterapkan pada tumbukan dua atau tiga dimensi, dan sifat vektor momentum sangat penting. Satu tipe umum dari tumbukan yang tidak berhadapan adalah di mana sebuah partikel yang bergerak (disebut proyektil) menabrak partikel kedua yang diam (partikel "target"). Ini

merupakan situasi umum pada permainan seperti bilyar, dan untuk eksperimen pada fisika atom dan nuklir (proyektil, dari pancaran radioaktif atau akselerator energi-tinggi, menabrak inti target yang stasioner).



Gambar 2.3 Tumbukan 2D

Dari kekekalan momentum , untuk komponen gerak dalam arah x :

$$m_1 v_{1i} = m_1 v_{1f} \cos\theta + m_2 v_{2f} \cos\phi \qquad \text{(Persamaan 13)}$$

untuk komponen gerak dalam komponen y :

$$0 = m_1 v_{1f} \sin\theta - m_2 v_{2f} \sin\phi \qquad \text{(Persamaan 14)}$$

Bila dianggap tumbukannya lenting :

$$\frac{1}{2} m_1 {v_1}^2 + \frac{1}{2} m_2 {v_2}^2 = \frac{1}{2} m_1 {v'_1}^2 + \frac{1}{2} m_2 {v_2}'^2 \qquad \text{(Persamaan 15)}$$

Bila keadaan awal diketahui, masih ada 4 besaran yang tidak diketahui, tetapi persaamannya hanya 3, oleh karena itu salah satu besaran keadaan akhir harus diberikan [11].

2.2. Metode Simulasi

Simulasi adalah tiruan dari operasi proses dunia nyata atau sistem dari waktu ke waktu [3]. Simulasi digunakan dalam banyak konteks, seperti simulasi teknologi untuk optimasi kinerja, rekayasa keselamatan, pengujian, pelatihan, pendidikan, dan video game. Seringkali, percobaan komputer digunakan untuk mempelajari model simulasi. Simulasi juga digunakan dengan pemodelan ilmiah sistem alam atau sistem manusia untuk mendapatkan informasi tentang fungsi

mereka. Simulasi dapat digunakan untuk menunjukkan efek nyata akhirnya kondisi dan tindakan alternatif. Simulasi juga digunakan ketika sistem nyata tidak dapat bergerak, karena tidak dapat diakses, atau mungkin berbahaya atau tidak dapat diterima untuk terlibat, atau sedang dirancang tetapi belum dibangun, atau mungkin hanya tidak ada [7].

2.3. Algoritma Gilbert Jhonson Keerthi

Algoritma adalah urutan langkah-langkah logis pada penyelesaian masalah yang disusun secara sistematis. Masalah dapat berupa apa saja, dengan catatan untuk setiap masalah ada syarat kondisi awal yang harus dipenuhi sebelum menjalankan algoritma. Konsep algoritma sering kali disetarakan dengan sebuah resep. Sebuah resep biasanya memiliki daftar bahan atau bumbu yang akan *digunakan, urutan pengerjaan, dan bagaimana hasil dari urutan pengerjaan tersebut*. Apabila bahan yang digunakan tidak tertera (tidak tersedia), maka resep tersebut tidak akan dapat dikerjakan. Demikian juga jika urutan pengerjaannya tidak beraturan, maka hasil yang diharapkan tidak akan dapat diperoleh

Algoritma yang berbeda, dapat diterapkan pada suatu masalah dengan syarat yang sama. Tingkat kerumitan dari suatu algoritma merupakan ukuran seberapa banyak komputasi yang dibutuhkan algoritma tersebut untuk menyelesaikan masalah. Umumnya, algoritma yang dapat menyelesaikan suatu permasalahan dalam waktu yang singkat memiliki tingkat kerumitan yang rendah, sementara algoritma yang membutuhkan waktu lama untuk menyelesaikan suatu masalah membutuhkan tingkat kerumitan yang tinggi.

Algoritma *Gilbert Johnson Keerthi* (GJK) merupakan algoritma yang ditujukan untuk menentukan jarak terdekat antara dua *convex shapes* (bentuk cembung) [6], tetapi dapat pula digunakan untuk menentukan apakah *convex shapes* tersebut berpotongan atau tidak. GJK hanya dapat digunakan pada *convex shapes*, Tidak seperti banyak algoritma jarak lainnya, GJK tidak memerlukan data geometri yang disimpan dalam format tertentu, melainkan hanya mengandalkan *support function* untuk membangun sebuah simplex. Pada dasarnya GJK secara iteratif

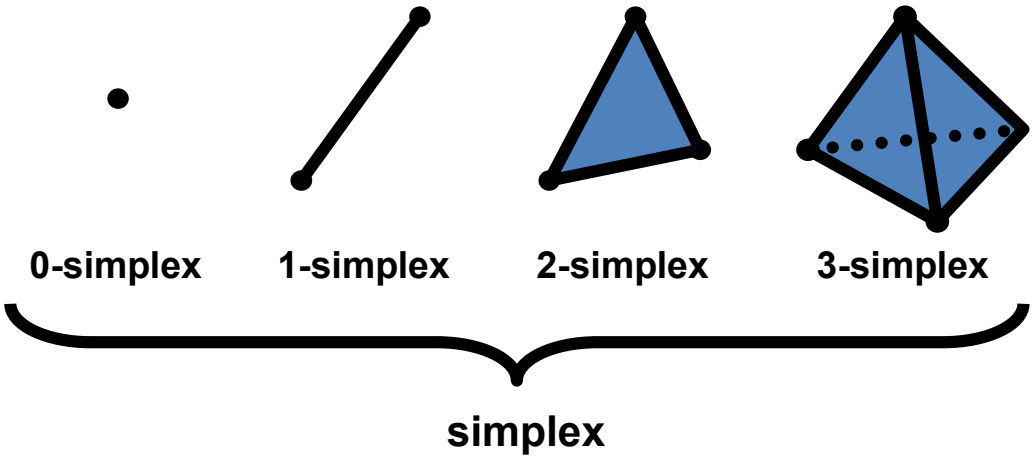
mencari *convex hull* (titik point) terdekat dengan bagian *origin* namun pada kasus pencarian jarak terdekat antara dua convex shapes maka GJK sangat bergantung pada sebuah konsep yang disebut minkowski sum [8].

Berikut algoritma GJK secara keseluruhan :

1. Simplex Initialze ke himpunan kosong (*simplex* -1D, secara teknis).
2. Gunakan arah awal untuk menemukan *support point* dari minkowski.
3. Menambahkan *support point* untuk *simplex*.
4. Mencari titik terdekat dari *simplex* ke *origin*
5. Jika titik terdekat adalah *origin*, maka minkowski berisi *origin* dan dua objek bertabrakan. (Akhir GJK).
6. Jika tidak, mengurangi simplex ke dimensi terendah yang masih mengandung titik terdekat dengan membuang simpul.
7. Gunakan arah dari titik terdekat ke *origin* untuk menemukan *support point* baru.
8. Jika *support point* baru tidak lebih dari sepanjang arah pencarian dari titik terdekat, dua benda tidak bertabrakan. (Akhir GJK).
9. Menambahkan *support point* untuk *simplex* sebagai titik baru. Ulangi ke langkah 4 [6].

2.3.1. Simplex

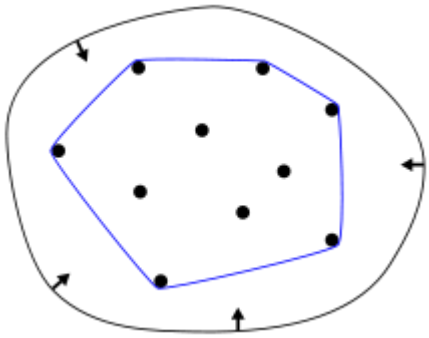
Untuk mendekati jarak yang sebenarnya suatu benda, GJK iteratif memilih sebuah simpleks (lihat Gambar II.10) dalam objek dan menghitung jarak ke simpleks ini. Sebuah n-simpleks adalah polytope dari n + 1 simpul dalam ruang n-dimensi. Untuk GJK, dua kasus yang paling menarik secara alami dua dimensi dan tiga dimensi ruang. Oleh karena itu, algoritma ini menggunakan 2-simpleks masing-masing segitiga untuk ruang dua dimensi dan 3-simpleks masing tetrahedron tiga dimensi ruang [9].



Gambar 2.4 Simplex

2.3.2. Convex Hull

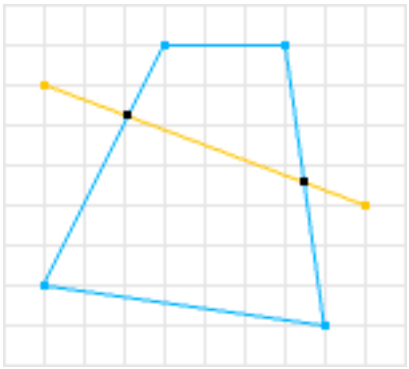
Dalam matematika, convex hull atau amplop cembung set X dari titik pada bidang Euclidean atau ruang Euclidean adalah himpunan cembung terkecil yang berisi X. Misalnya, ketika X adalah subset dibatasi bidang, convex hull dapat divisualisasikan sebagai bentuk dibentuk oleh karet gelang membentang sekitar X [10]. GJK bekerja dengan benda-benda cembung saja dan mendekati jarak objek yang sebenarnya dengan jarak komputasi untuk simplices dalam object. Terlebih, mungkin perlu untuk menggunakan convex hull dari simplex saat perhitungan, jika tidak bentuk cembung sendiri [9].



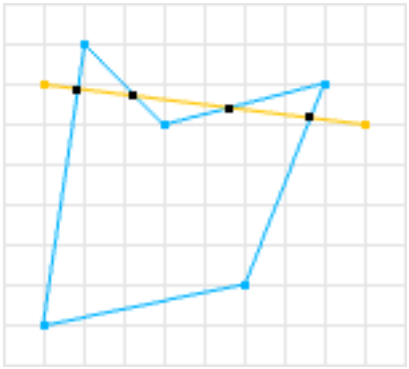
Gambar 2.5 Convex Hull dalam analogi karet gelang

2.3.3. Convex

Sebuah shapes dianggap convex jika, untuk setiap garis yang ditarik melalui shapes, garis yang melintasinya hanya dua kali. Jika garis dapat ditarik melalui bentuk dan menyeberang lebih dari dua kali maka shapes tersebut bukanlah convex (atau cekung). Contoh :



Gambar 2.6 Convex



Gambar 2.7 Non - Convex

Shape pertama dianggap convex karena tidak ada garis yang dapat ditarik melalui bentuk di mana ia akan menyeberanginya lebih dari dua kali. Bentuk kedua tidak convex karena ada garis yang melintasi lebih dari dua kali.

2.3.4. Minkowski Different

Algoritma GJK sangat bergantung pada konsep yang disebut Minkowski Sum. Konseptual Minkowski Sum sangat mudah dimengerti. Apabila kita memiliki

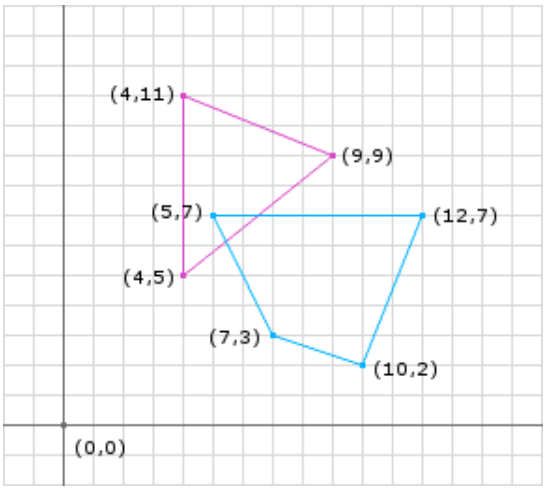
dua buah shapes, Minkowski Sum dari shape tersebut adalah semua poin di shape 1 ditambahkan ke semua poin di shape 2:

$$C = A + B = \{a + b \mid a \in A, b \in B\} \qquad \text{(Persamaan 16)}$$

Dengan kata lain, setiap titik B ditambahkan ke setiap titik A dan Hasil operasi membentuk objek baru C. Karena hasilnya adalah satu set juga, mungkin duplikat poin berkembang dari penambahan adalah mengeliminasi terkontaminasi. Dengan demikian, kardinalitas C selalu lebih kecil atau sama dengan jumlah yang dari kardinalitas A dan B [9]. Untuk mengetahui apakah kedua convex shapes tersebut bertumbukan atau berpotongan maka operator diubah menjadi dikurangi yang bernama Minkowski Different :

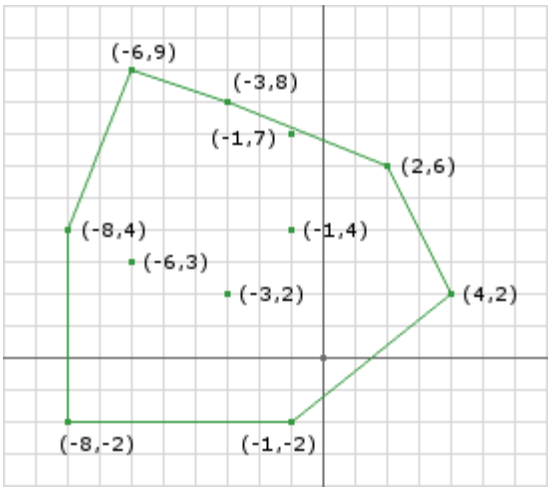
$$C = A - B = \{a - b \mid a \in A, b \in B\} \qquad \text{(Persamaan 17)}$$

Minkowski Different adalah kunci untuk ide algoritma mengurangi masalah kedekatan dua benda pada satu jarak object. Dua convex shapes A dan B adalah sama dengan jarak mereka. Jika dua convex shape tersebut bertumbukan / berpotongan maka Minkowski different akan berisi asal (origin).



Gambar 2.8 Dua convex shape berpotongan

Dari contoh gambar diatas dua convex shapes berada dalam kondisi berpotongan maka ketika dilakukan Minkowski Different maka akan menghasilkan shapes pada Gambar II.9



Gambar 2.9 Minkowski Different

Perhatikan bahwa shapes yang dihasilkan mengandung asal (0,0). Hal ini karena shapes yang berpotongan.

2.3.5 Support Function

Support function diciptakan untuk membangun simplex. Suport Function seharusnya mengembalikan poin kedalam Minkowski Different yang dihasilkan dua shapes. Telah diketahui bahwa kita dapat mengambil poin dari shape 1 dan titik dari shape 2 dan mengurangi mereka untuk mendapatkan poin dalam Perbedaan Minkowski.

Dapat dipastikan bahwa kita tidak mendapatkan poin yang sama setiap pemanggilan support function jika kita membuat fungsi dukungan tergantung pada arah. Dengan kata lain, jika kita mengembalikan support function pada titik terjauh dalam beberapa arah, kita bisa memilih arah yang berbeda kemudian untuk mendapatkan sudut pandang yang berbeda. Memilih titik terjauh dalam arah memiliki makna karena ia menciptakan simpleks yang mengandung luas maksimum sehingga meningkatkan kemungkinan bahwa algoritma selesai dengan cepat. Selain itu, kita dapat menggunakan fakta bahwa semua poin kembali dengan

cara ini berada di tepi minkowski different dan oleh karena itu jika kita tidak dapat menambahkan poin sebelumnya asal sepanjang beberapa arah kita tahu bahwa minkowski different tidak mengandung asal. Hal ini meningkatkan kemungkinan algoritma selesai dengan cepat dalam kasus-kasus non-convex.

2.4. Adobe Flash CS5

Flash merupakan program atau software yang ditujukan untuk membuat animasi, desain, web, aplikasi, game, dan sebagaianya. Flash lebih dikenal khalayak ramai sebagai software pembuat animasi. Seiring perkembangan zaman yang semakin maju dan pesat dalam bidang teknologi khususnya dalam bidang teknologi informasi, flash sampai sekarang telah mencapai versi 11 atau lebih dikenal dengan Adobe Flash CS5 Profesional yang dirilis oleh perusahaan software ternama di Amerika Serikat, yaitu Adobe System Incorporated.

Flash CS5 yang sekarang banyak dikenal dengan kemampuan yang andal dalam membuat animasi-animasi, tetapi tidak jarang yang mengenal sisi lain dari kelebihan Flash ini. Salah satunya adalah mempunyai bahasa pemrograman yang sangat ampuh dalam segala hal, yaitu Action Script yang sekarang sudah dirilis samapai ke versi 3.0 yang dimiliki oleh Flash CS3,CS4, dan CS5.

Adobe Flash CS5 menawarkan beberapa pembaruan yang komplet dengan penawaran penggunaan lebih menyenangkan. Fasilitas seperti 3D Effect atau transformations yang sangat menarik dapat digunakan untuk membuat efek-efek animasi 3 dimensi yang menarik. Berbagai fasilitas dan fitur terbaru telah disediakan untuk kemudahan dalam pengolahan para penggunanya. Adobe Flash CS5 merupakan penyempurnaan dari versi sebelumnya (CS4)[14].

2.4.1. Pengenalan Tool dalam Adobe Flash CS5

Beberapa komponen kerja program Adobe Flash CS5 telah ditampilkan sebagai tampilan standart. Masih banyak lagi komponen yang masih tersembunyi sehingga memerlupan perintah tertentu untuk menampilkannya. Berikut ini adalah beberapa komponen kerja dari Adobe Flash CS5[14]:

1. *Toolbox.*
2. *Timeline*

   Timeline berguna untuk menentukan durasi animasi, jumlah layer, frame, menempatkan script dan beberapa keperluan animasi lainnya. Semua bentuk animasi yang Anda buat akan diatur dan ditempatkan pada layer dalam timeline.

3. *Output.*
4. *Compiler errors.*
5. *Motion Editor*

   Motion editor berguna untuk melakukan control animasi yang telah dibuat seperti: mengatur motion, transformasi, pewarnaan, filter dan parameter animasi lainnya.

6. *Stage*

   Stage adalah lembar kerja yang digunakan untuk membuat atau mendesain objek yang akan dianimasikan. Objek yang dibuat dalam lembar kerja dapat berupa objek vector, movie clip, text, button dan lain-lain.

7. *Properties, library.*
8. *Color, swatches.*
9. *Align, info, transform.*

2.5. Pemrograman Berorientasi Objek

Metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkatl unak berorientasi objek didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas. Metode berorientasi objek meliputi rangkaian aktivias analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek, dan pengujian berorientasi objek.

Pada saat ini, metode berorientasi objek banyak dipilih karena metodologi lama banyak menimbulkan masalah seperti adanya kesulitan pada saat mentransformasi hasil dari suatu tahap pengembangan ke tahap berikutnya, misalnya pada metode pendekatan terstruktur, jenis aplikasi yang dikembangkan saat ini berbeda dengan masa lalu. Aplikasi yang dikembangkan pada saat ini sangat beragam (aplikasi bisnis, *real-time*, *utility*, dan sebagainya) dengan *platform* yang berbeda-beda, sehingga menimbulkan tuntutan kebutuhan metodologi pengembangan yang dapat mengakomodasi ke semua jenis aplikasi tersebut [12].

Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut [12]:

1. Meningkatkan produktivitas

   Karena kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai ulang untuk masalah lainnya yang melibatkan objek tersebut (*reusable*).
2. Kecepatan pengembangan

   Karena sistem yang dibangun dengan baik dan benar pada saat analisis dan perancangan akan menyebabkan berkurangnya kesalahan pada saat pengkodean.
3. Kemudahan pemeliharaan

   Karena dengan model objek, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering berubah-ubah.
4. Adanya konsistensi

   Karena sifat pewarisan dan penggunaan notasi yang sama pada saat analisis, perancangan maupun pengkodean.
5. Meningkatkan kualitas perangkat lunak

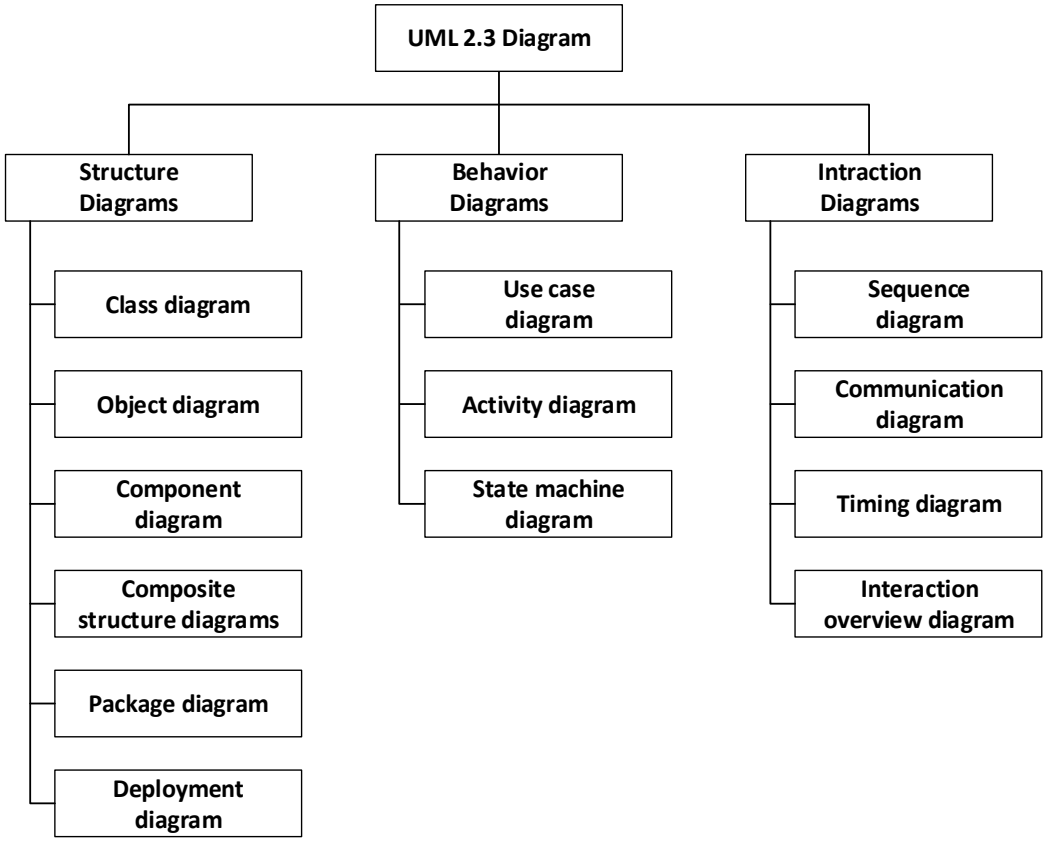
   Karena pendekatan pengembangan lebih dekat dengan dunia nyata dan adanya konsistensi pada saat pengembangannya, perangkat lunak yang dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

2.5.1 UML (*Unified Modeling Language*)

UML adalah bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek. UML diperlukan karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung [12].

2.5.2 Diagram UML

Pada UML 2.3 terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori [12]. Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada gambar 2.10



Gambar 2.10 Diagram UML [12]

Berikut ini adalah penjelasan singkat dari pembagian kategori tersebut:

a. *Structure diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
b. *Behavior diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
c. *Interaction diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem.

Penjelasan tentang masing-masing diagram adalah sebagai berikut:

1. *Class Diagram*

   Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.
2. *Object Diagram*

   Diagram objek menggambarkan struktur sistem dari segi penamaan objek dan jalannya objek dalam sistem. Diagram objek juga berfungsi untuk mendefinisikan contoh nilai atau isi dari atribut tiap kelas.
3. *Component Diagram*

   Diagram komponen atau *component diagram* dibuat untuk menunjukan organisasi dan ketergantungan diantara kumpulan komponen dalam sebuah sistem. Diagram komponen fokus pada komponen sistem yang dibutuhkan dan ada di dalam sistem
4. *Composite Structure Diagram*

   Diagram ini dapat digunakan untuk menggambarkan struktur dari bagian-bagian yang saling terhubung maupun mendeskripsikan struktur pada saat berjalan (*runtime*) dari *instance* yang saling terhubung.
5. *Package Diagram*

   Package diagram menyediakan cara mengumpulkan elemen-elemen yang saling terkait dalam diagram UML. Hampir semua diagram dalam UML dapat dikelompokkan menggunakan *package diagram*.

6. *Deployment Diagram*

   Deployment diagram menunjukkan konfigurasi komponen dalam proses eksekusi aplikasi.

7. *Use Case Diagram*

   Use case diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat.

8. *Activity Diagram*

   Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak.

9. *State Machine Diagram*

   State machine diagram atau *state chart diagram* digunakan untuk menggambarkan perubahan status atau transisi status dari sebuah mesin atau sistem atau objek.

10. *Sequence Diagram*

    Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek.

11. *Communication Diagram*

    Diagram komunikasi menggambarkan interaksi antar objek/bagian dalam bentuk urutan pengiriman pesan. Diagram komunikasi merepresentasikan informasi yang diperoleh dari diagram kelas, diagram sekuen, dan diagram *use case* untuk mendeskripsikan gabungan antara struktur statis dan tingkah laku dinamis dari suatu sistem.

12. *Timing Diagram*

    Timing diagram merupakan diagram yang fokus pada penggambaran terkait batasan waktu. *Timing diagram* digunakan untuk menggambarkan tingkah laku sistem dalam periode waktu tertentu.

13. *Interaction Overview Diagram*

    Interaction overview diagram berfungsi untuk menggambarkan sekumpulan urutan aktivitas. *Interaction overview diagram* adalah bentuk aktivitas diagram yang setiap titik merepresentasikan diagram interaksi.