BAB II

LANDASAN TEORI

2.1 Karate

Karate adalah seni bela diri yang berasal dari Jepang. *Karate* sendiri memiliki arti tangan kosong. Menurut *Zen-Nippon Karatedo Renmei/Japan Karatedo Federation* (JKF) dan *World Karatedo Federation* (WKF), ada beberapa aliran karate yang utama yaitu *Shotokan*, *Goju-Ryu*, *Shito-Ryu*, dan *Wado-Ryu*

Selain aliran-aliran tersebut terdapat aliran lain yang memiliki karakteristik tersendiri, seperti *Kyokushin*, *Shorin-Ryu* dan *Uechi Ryu*. Masing-masing aliran memiliki karakteristik masing masing. Beberapa aliran tersebut memiliki penekanan berbeda-beda dalam teknik *karate*, ada yang menekankan *Kata*, penggunaan senjata lain selain tangan kosong, dan *full body contact system*.

Dalam mempelajari *Karate*, terdapat beberapa tingkatan yang menandakan seberapa tinggi kemampuan bela diri *Karate*. Dimulai dari sabuk putih dimana semua orang yang ingin belajar *Karate* akan mendapat sabuk ini tanpa melalui tahap ujian. Tingkatan selanjutnya secara bertahap adalah kuning, hijau, biru coklat dan yang terakhir hitam.

Latihan dasar *Karate* terbagi tiga seperti berikut:

a. *Kihon*, yaitu latihan teknik-teknik dasar *Karate* seperti teknik memukul, menendang dan menangkis.
b. *Kata*, yaitu latihan jurus atau bunga *Karate*.
c. *Kumite*, yaitu latihan tanding atau *sparring*.

INSTITUT BISNIS & INFORMATIKA stikom SURABAYA

Kihon sendiri merupakan teknik dasar yang terdiri dari teknik menyerang dan bertahan. *Kihon* dilakukan secara berulang-ulang agar terbentuk respon gerakan yang cepat dan akurat. Latihan *Kihon* sendiri terdiri dari berbagai latihan yang berfungsi untuk melatih beberapa aspek dalam Karate, antara lain *Gueri*(tendangan), *Dachi* (posisi kaki), *Uke* (tangkisan), *Tsuki* (pukulan), *Uchi* (bentuk tangan).

Setiap aspek tersebut memiliki gerakan yang bervariasi dari tingkat kesulitan rendah ke tinggi. Terdapat beberapa posisi juga yang memiliki kemiripan yang hampir sama.

Gambar 2.1 Posisi *Oi-tsuki* (Santa-María, 2001)

Gambar 2.2 Posisi *Kizami-tsuki* (Santa-María, 2001)

Seperti *oi-tsuki* pada Gambar 1 dan *kizami-tsuki* pada Gambar 2, kedua posisi badan dan kaki memiliki posisi yang sama namun yang membedakan antara dua posisi karate ini adalah derajat kemiringan tangan dimana *Oi-tsuki* menarget dada lawan dan *Kizami-tsuki* mengarahkan pukulan ke mata lawan.

2.2 Metode Pembelajaran untuk Keterampilan Motorik

Meningkatkan keterampilan motorik manusia melibatkan berbagai faktor penting dalam masalah kemampuan kognitif. Salah satu masalah utama yang dihadapi adalah bagaimana mengambil informasi yang relevan dari lingkungan belajar. Terdapat tiga lingkungan belajar yang digunakan dalam mempelajari keterampilan motorik, yaitu *traditional group learning environment, video learning environment, dan Virtual Reality(VR) learning environment. Traditional group learning environment* adalah lingkungan belajar yang umum dimana pengajar dan murid bertemu dalam satu tempat dan melakukan kegiatan pembelajaran yang dipimpin oleh guru. Sedangkan video dan VR *learning environment* memanfaatkan teknologi dalam proses pembelajaran. Video *learning environment* dilakukan dengan melihat tayangan video yang berisi gerakan yang ingin dipelajari melalui layar maupun teknologi tingkat tinggi seperti *head mounted display*. Sedangkan VR membutuhkan alat bantu seperti *motion tracking sensor* dan bantuan visualisasi *virtual humans* yang dihasilkan dari proses rendering gerakan.

Masing-masing lingkungan ini memiliki kelebihan dan kekurangan masing-masing. *Traditional learning environment* memiliki kelebihan dimana pengajar dapat melakukan proses penyampaian umpan balik ke murid secara langsung. Namun kekurangan yang dimiliki dari *learning environment* ini adalah kebutuhan untuk bertatap muka secara langsung dan secara terjadwal. Jumlah rasio

dari pengajar dan murid juga sangat mempengaruhi efektifitas dari lingkungan belajar ini. *Video learning environment* sangat berguna bagi kalangan orang yang ingin belajar namun tidak terikat dalam proses tersebut dimana *learning environment* ini memiliki sifat yang fleksibel, baik waktu, tempat dan pengetahuan. *VR learning environment* memerlukan teknologi yang cukup mahal yang tidak dapat dijangkau kalangan awam. Dalam penelitiannya, Burns et al(2011) menyimpulkan bahwa *video learning environment* dan VR tidak memiliki pengaruh negatif dalam proses mengajar yang ditunjukkan dalam studi kasus Karate.

2.3 Kinect

Sensor *Kinect* merupakan salah satu proyek pengembangan alat *input* berdasarkan *motion-sensing* oleh Microsoft yang diumumkan dengan nama *Project Natal* pada bulan Juni 2009. Sensor *Kinect* merupakan salah satu teknologi yang dapat menangkap pergerakan manusia melalui *depth sensor,* kamera RGB dan didukung oleh sebuah *microphone array* (Catuhe, 2012). Pada kemunculan perdananya, sensor ini dijual dalam satu paket dengan konsol permainan Xbox 360. Dalam perkembangannya, Microsoft mengeluarkan SDK *Kinect* untuk penggunaan komersil yang memungkinkan para *developer* program mengembangkan program mengunakan sensor *Kinect*. Selain digunakan untuk konsol permainan Xbox 360, *Kinect* dikembangkan ke bidang lain. Beberapa contohnya adalah dalam bidang edukasi *Kinect* digunakan oleh staff pengajar untuk melakukan pembelajaran yang interaktif dengan siswanya melalui *Interactive White Board,* lalu dalam bidang kesehatan *Kinect* digunakan sebagai alat rehabilitasi pada beberapa penyakit seperti *dsylexia*.

Dari sisi elektronik, sensor *Kinect* terdiri dari beberapa komponen yang tampak pada gambar 3, yaitu :

1. Mikrofon
2. Pemancar sinar *infrared*
3. Penerima sinar *infrared*
4. Kamera RGB

Sensor *Kinect* ini dapat dihubungkan dengan komputer melalui *port* standar USB 2.0 yang memerlukan sebuah catu daya sebagai tenaga tambahan.

Gambar 2.3 Struktur Kinect

Dalam proses pendeteksian sensor *Kinect*, terdapat 3 macam aliran data yang disampaikan sensor *Kinect*. Yang pertama adalah aliran data gambar berwarna yang diperoleh dari kamera RGB. Kedua, aliran data *depth sensor* yang dihasilkan oleh sinar *infrared* yang dikeluarkan dan diterima oleh sensor *Kinect* yang dapat mengukur letak benda maupun orang yang ada di jangkauan sensor *Kinect*. Dan yang terakhir adalah aliran data suara yang ditangkap oleh *microphone array*. Sensor *Kinect* tidak memiliki *Central Processing Unit*, dimana alat ini berfungsi sebagai *digital signal processor* (DSP) yang digunakan untuk mengolah sinyal dari

microphone array. Melalui 3 data yang dihasilkan sensor *Kinect* ini, data akan diproses di komputer melalui pustaka pengendali perangkat keras *Kinect*.

Semenjak Microsoft mengeluarkan *Kinect* for *Windows Software Development Kit* pada Juni 2011, para *developer* di seluruh dunia dapat menggunakan sensor *Kinect* dalam pengembangan aplikasi komersil. SDK resmi yang dirilis oleh Microsoft yang terbaru sampai bulan Mei 2013 adalah SDK versi 1.7. SDK ini berisi berbagai kelas yang berguna bagi pengembang aplikasi yang ingin menggunakan sensor *Kinect.*

2.4 Bahasa Pemrograman C#

C# adalah sebuah bahasa pemrograman yang dikembangkan oleh Anders Hejlsberg seorang *program designer* Microsoft. C# sendiri merupakan pengembangan dari C++ dimana bahasa pemrograman ini diharapkan dapat menangani kekurangan yang dimiliki oleh C++. Bahasa pemrograman C# mendukung konsep pemrograman berorientasi objek, namun juga mendukung *generics,* dan *functional programming*.

C# memiliki kelebihan yang berasal dari *runtime* C# (*Common Language Runtime* - CLR) yang menyediakan beberapa servis antara lain *security sandboxing, runtime type checking, exception handling, thread management,* dan yang paling penting *automated memory management* (Griffiths, 2012). Selain kelebihan diatas, bahasa pemrograman C# didukung oleh .NET *framework* yang dipakai di Microsoft Windows. Dengan .NET, C# dapat menggunakan berbagai *library* yang ada di Microsoft Windows.

2.5 *Skeleton Tracking*

Salah satu dari tiga aliran data yang dihasilkan oleh sensor Kinect adalah *depth data*. Aliran data ini bersifat mentah dimana diperlukan sebuah metode yang dapat mendayagunakan *depth data* ini. *Skeleton tracking* adalah sebuah fitur dari sensor Kinect yang dikembangkan untuk mengolah *depth data* menjadi informasi yang berguna, yaitu *skeleton*/rangka manusia. *Skeleton tracking* menyediakan nilai koordinat X, Y, Z dalam bidang kartesian tiga dimensi untuk setiap *joint*/sendi dari rangka manusia yang terdeteksi.

Proses pendeteksian *skeleton* mencakup proses analisis *depth image* dengan menggunakan algoritma yang kompleks yang meliputi transformasi matrik, *machine learning*, dan hal lainnya yang berhubungan untuk mengkalkulasi *skeleton points. Skeleton data* didapat dari *skeleton stream* yang perlu diinisialisasi sehingga secara *realtime* menghasilkan koleksi dari *skeleton objects.* Tiap *skeleton object* memiliki data yang mendeskripsikan lokasi dari *skeleton* dan *skeleton joints.* Setiap skeleton joint memiliki identitas berupa nama (kepala, pundak, lengan, siku, dan lain-lain) dan vektor 3 dimensi.

Proses *skeleton tracking* dapat dibantu dengan penggunaan *class Skeleton* dari SDK *Kinect for Windows*. Melalui kelas *Skeleton*, *Skeleton stream* akan menghasilkan *SkeletonFrame objects*. Proses pendeteksian *skeleton* ini memiliki hasil yang perlu disesuaikan agar hasil pendeteksian menjadi lancar. Proses penyesuaian dapat dilakukan melalui penyesuaian parameter yang ada pada *skeleton stream.* Parameter yang dapat disesuaikan adalah Correction, JitterRadius, MaxDeviationRadius, Prediction dan Smoothing.

INSTITUT BISNIS & INFORMATIKA stikom SURABAYA

Skeleton tracking menghasilkan posisi dari seluruh *joint* yang dapat dideteksi menggunakan sensor Kinect. Dari data posisi ini dapat digambarkan sebuah visualisasi dari *skeleton* yang dideteksi. Masing-masing persendian dihubungkan membentuk "*stick-man*" yang merupakan representasi dari tubuh manusia yang dideteksi.

2.6 *Agile Software Development*

Perkembangan lingkungan pengembangan perangkat lunak membutuhkan waktu sesingkat mungkin dalam proses penyampaian dan pembuatan perangkat lunak. Para pengembang perangkat lunak menggunakan berbagai pendekatan dalam proses pembuatan perangkat lunak yang dapat menghasilkan produk dalam waktu yang singkat namun tanpa mengurangi kualitas produk yang dibuat. Salah satu pendekatan dalam pengembangan perangkat lunak yang dapat membantu mempercepat pembuatan perangkat lunak adalah *Agile software development.*

Agile memiliki beberapa karakteristik antara lain adalah *modularity, iterative, time-bound, parsimony, adaptive, incremental, convergent, people oriented,* dan *collaborative* (Miller, 2001). Karakteristik ini yang memungkinkan sebuah produk perangkat lunak dapat dikembangkan dalam waktu singkat namun tanpa mengurangi fitur sebuah produk.

Beberapa karakteristik ini memiliki peran penting dalam proses pengembangan perangkat lunak dalam tugas akhir ini. Karakteristik yang pertama yaitu *modularity* memungkinkan sebuah proses dibagi-bagi menjadi beberapa komponen yang disebut aktivitas. Proses pengembangan perangkat lunak membentuk beberapa aktifitas yang mengubah perencanaan terealisasi. Karakteristik yang kedua adalah *iterative* dimana karakteristik ini merupakan

INSTITUT BISNIS & INFORMATIKA stikom SURABAYA

proses pengulangan pembuatan sebuah aktifitas dimana *Agile* memiliki pandangan bahwa terkadang kala kita melakukan kesalahan sebelum kita melakukannya dengan benar. Proses iterasi ini dilakukan untuk memonitor proses pengembangan perangkat lunak ketika mengalami kesalahan maupun perubahan. *Time-bound*, karakteristik ini berhubungan dengan lamanya sebuah pengembangan perangkat lunak dilakukan. Tiap proses memiliki batas waktu yang jelas sehingga masing-masing aktifitas dapat dibangun dalam jangka waktu yang jelas. Karakteristik selanjutnya adalah *adaptive* yang memiliki arti bahwa dalam proses iterasi dapat timbul resiko yang membutuhkan tambahan aktifitas yang tak terencana. Namun hal ini bukan sebuah masalah dalam Agile mengingat karakteristik ke dua yaitu *iterative* yang memungkinkan adanya penyesuaian dalam proses ini. Incremental adalah karakteristik lainnya yang mendukung karakteristik *iterative* karena *Agile* tidak membuat seluruh sistem dalam sekali perencanaan, namun terbagi-bagi dalam sub aktifitas.

2.7 *Unified Modelling Language*

UML adalah singkatan dari *Unified Modelling Language* yang merupakan standarisasi pemodelan dalam *software engineering*. UML merupakan kumpulan notasi grafis yang disertai dengan *single meta-model* yang membantu mendeskripsi dan mendesain *software systems* yang mengunakan konsep *object oriented* (Fowler, 2004). UML merupakan metodologi kolaborasi antara metode Booch yang dikembangkan oleh Graddy Booch, OMT (*Object Modelling Technique*) yang dikembangkan oleh DR. James Rumbaugh, serta OOSE (*Object Oriented Software Engineering*) yang dikembangkan oleh Ivar Jacobson, dan beberapa metode lainnya, merupakan metodologi yang paling sering/tepat digunakan yang

berparadigma pemrograman berorientasi objek. Penggunaan UML sendiri dapat dianalogikan sebagai cetak biru dari sebuah bangunan dimana cetak biru ini merupakan rancangan dari sebuah desainer bangunan yang berisikan detil desain yang lengkap. Sama seperti UML, cetak biru ini diaplikasikan dalam *software engineering* dimana desainer perangkat lunak menuangkan hasil desain programnya ke dalam notasi-notasi berbentuk grafik yang memiliki standar dimana dengan standar ini diharapkan seluruh orang yang terlibat dalam pengembangan *software* mampu mengerti desain yang sudah dihasilkan oleh desainer perangkat lunak. Cetak biru ini berisikan berbagai detil informasi terkait dengan proses pembuatan perangkat lunak.

UML digunakan dalam ruang lingkup pemrograman berorientasi objek yang dapat membantu desainer perangkat lunak sebagai sarana analisis, pemahaman, visualisasi, dan komunikasi antar anggota tim pengembang (saat seorang analisis/perancang perangkat lunak bekerja dalam tim yang beranggotakan beberapa orang), serta sebagai sarana dokumentasi. Peran UML sendiri adalah sebagai kakas (*tools*) yang membantu penyampaian informasi dimana UML digunakan dengan menggunakan metodologi *Unified Software Development Process* (USDP).

Dalam UML terdapat berbagai view dan diagram yang digunakan. View yang dimaksud adalah sejumlah konstruksi pemodelan UML yang merepresentasikan suatu aspek tertentu dari perangkat lunak yang dikembangkan. View dapat diklasifikasikan menjadi 3 area utama, yaitu : klasifikasi struktural, perilaku dinamis, serta pengelolaan model. Keterangan mengenai *view* dan *diagram* dalam UML dapat dilihat pada Tabel 2.1.

INSTITUT BISNIS & INFORMATIKA stikom SURABAYA

Tabel 2.1 *View* dan *Diagram* dalam UML (Nugroho, 2010)

Major Area	*View*	*Diagrams*	*Main Concepts*
Structural	*Static view*	*Class diagram*	*Class, association,generalization, dependancy, realization, interface*
	Use case view	*Use case diagram*	*Use case, actor, association, extend,include, use case generalization*
	Implementation view	*Component diagram*	*Component, interface, dependancy, realization*
	Deployment view	*Deployment diagram*	*Node, component, dependancy, location*
Dynamic	*State machine view*	*Statechart diagram*	*Static, event, transition, action*
	Activity view	*Activity diagram,*	*State, activity, completion transition, fork, join*
	Interaction view	*Sequence diagram*	*Interaction, object, message, activation*
		Collaboration diagram	*Collaboration, interaction, collaboration role, message*
Model management	*Model management view*	*Class diagram*	*Package, subsystem, model*
Extensibility	*All*	*All*	*Constraint, streotype, tagged values*

2.8 *Unified Process*

USDP adalah metode yang memungkinkan UML digunakan sebagai sarana pengembangan sistem/perangkat lunak mengingat UML hanyalah sebuah kakas. USDP sesungguhnya merupakan salah satu metode rekayasa perangkat lunak berorientasi objek yang secara konsisten mencoba beradaptasi dengan semakin besar dan semakin kompleksnya sistem-sistem/perangkat lunak-perangkat lunak yang dikembangkan oleh para *vendor* perangkat lunak di seluruh dunia (Nugroho, 2010). Sekarang USDP lebih dikenal dengan nama *Unified Process.*

USDP memiliki karaterisitik *use-case driven, architecture centric, iterative & incremental*. *Use-case driven* memiliki arti bahwa perangkat lunak yang dihasilkan semestinya bersifat melayani para penggunanya dan sesuai dengan kebutuhan dan harapan pengguna. *Architecture centric* memiliki arti bahwa pada dasarnya arsitektur memuat aspek-aspek statis dan dinamis perangkat lunak yang bersangkutan sama halnya dengan arsitektur pada sistem konstruksi teknik sipil. Sedangkan *iterative dan incremental* dibagi ke dalam beberapa proyek-proyek kecil dimana masing-masing proyek yang lebih kecil dikerjakan secara iteratif sehingga pada akhirnya menghasilkan perangkat lunak yang terintegrasi berukuran besar yang terbentuk secara *incremental*. Iterasi berlangsung pada bagian-bagian kecil dari sistem yan dapat memperluas fungsionalitas sistem menjadi lebih besar.

Unified Process atau yang dikenal dengan *Unified Software Development Process* menggunakan diagram-diagram UML dalam proses perancangan perangkat lunak. *Use case* merupakan diagram yang bersifat sentral dimana sistem perangkat lunak yang dibangun berusaha memenuhi kebutuhan dan harapan pengguna serta pengembangan sistem/perangkat lunak yang dipandu oleh diagram *use case*. Namun juga terdapat beberapa model-model selain *use case* yang memiliki fungsi masing-masing.

Model analisis digunakan untuk memperhalus dan memperinci definisi masing-masing *use case*. Spesifikasi kebutuhan dan harapan pengguna dituangkan dalam model ini. Dapat berupa *user story* dimana dijelaskan sebuah kegiatan yang dilakukan oleh pengguna dengan tujuan yang spesifik sehingga memperoleh manfaat dari kegiatan tersebut.

INSTITUT BISNIS & INFORMATIKA stikom SURABAYA

Model perancangan mendefinisikan struktur statis sistem seperti subsistem, kelas, dan antarmuka beserta hubungannya dalam kerangka sistem yang dikembangkan.

Model implementasi memuat komponen-komponen yang merepresentasikan sistem sesuai dengan bahasa pemrograman yang dipilih dalam pengembangan perangkat lunak dan melakukan pemetaan kelas-kelas ke komponen-komponen.

Model *deployment* mendefinisikan simpul-simpul komputer secara fisik yang berfungsi sebagai rencana dan panduan dalam proses menerapkan perangkat lunak ketika digunakan dalam proses aktual sistem. Model ini mencakup hubungan antar komputer seperti bagaimana komputer dihubungkan melalui jaringan dan konfigurasinya. Model ini digunakan tidak hanya ketika perangkat lunak diimplementasikan dalam cakupan perangkat lunak yang terhubung satu sama lain, namun juga mencakup kebutuhan perangkat keras dari proses *deployment* tersebut.

Model pengujian atau *testing* mendeskripsikan kasus-kasus dan prosedur pengujian yang tujuannya adalah melakukan verifikasi terhadap perangkat lunak yang dihasilkan dengan cara melihat dan memastikan apakah yang diharapkan sesuai dengan hasil aktualisasi perangkat lunak melalui skenario pengujian.

2.9 Sudut Vektor

Vektor adalah obyek geometri yang memiliki besar dan arah yang dilambangkan dengan tanda panah (→) (Kurnianingsih & Kuntarti, 2007). Vektor dalam ruang euklidian tiga dimensi memiliki 3 komponen yaitu *i*,*j* dan *k*. Untuk mengetahui sudut antara dua buah vektor dalam sebuah bidang 3 dimensi

dibutuhkan beberapa operasi matematika yang berasal dari rumus dasar perkalian *dot product* (Wolfram, 2014). Rumus *dot product* ini dapat dilihat dalam rumus 2.1

$$\mathbf{X} \cdot \mathbf{Y} = |\mathbf{X}|\,|\mathbf{Y}| \cos\theta, \quad \text{2.1}$$

Dimana X.Y merupakan perkalian dot dari dua buah vektor [*x,y*] yang didapat dengan mencari panjang masing-masing vektor X dan Y. Panjang vektor dalam ruang euklidian tiga dimensi dilambangkan dengan notasi |vektor| yang didapat dari perhitungan $\sqrt{x_i{}^2 + x_j{}^2 + x_k{}^2}$. Dari rumus dasar di atas, dapat dikembangkan sebuah rumus untuk mencari nilai θ yang merupakan besar sudut antara dua vektor X dan Y.

Sudut antara dua vektor X dan Y dapat dicari dengan mencari nilai θ yang berasal dari nilai arc cos θ yang didapat. Rumus yang digunakan untuk mencari besar sudut dengan diketahui komponen vektor X dan Y dapat dilihat di rumus 2.2.

$$\cos\theta = \frac{X.Y}{|X||Y|} = \frac{x_i y_i + x_j y_j + x_k y_k}{\sqrt{x_i{}^2 + x_j{}^2 + x_k{}^2}.\sqrt{y_i{}^2 + y_j{}^2 + y_k{}^2}} \quad \text{2.2}$$

Nilai θ yang didapat merupakan sudut antara dua buah vektor dalam ruang tiga dimensi.

INSTITUT BISNIS & INFORMATIKA stikom SURABAYA