TUGAS AKHIR – KS141501

PENGEMBANGAN APLIKASI *BASIC TAEKWONDO TRAINING SYSTEM* BERBASIS TEKNOLOGI *REAL TIME MOTION CAPTURE* MENGGUNAKAN MICROSOFT KINECT

MUHAMMAD AFIF HENDRAWAN
NRP 5210 100 016

Dosen Pembimbing 1
Dr. Eng. Febriliyan Samopa, S.Kom., M.Kom.
Dosen Pembimbing 2
Nisfu Asrul Sani, S.Kom., M.Sc.

JURUSAN SISTEM INFORMASI
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2015

FINAL PROJECT – KS141501

DEVELOPMENT OF BASIC TAEKWONDO TRAINING SYSTEM USING REAL TIME MOTION CAPTURE TECHNOLOGI WITH MICROSOFT KINECT

MUHAMMAD AFIF HENDRAWAN
NRP 5210 100 016

Academic Promotor 1
Dr. Eng. Febriliyan Samopa, S.Kom, M.Kom
Academic Promotor 2
Nisfu Asrul Sani, S.Kom, M.Sc

DEPARTMENT OF INFORMATION SYSTEM
Faculty of Information Technology
Institute of Technology Sepuluh
Nopember Surabaya 2015

PENGEMBANGAN APLIKASI *BASIC TAEKWONDO TRAINING SYSTEM* BERBASIS TEKNOLOGI *REAL TIME MOTION CAPTURE* MENGGUNAKAN MICROSOFT KINECT

TUGAS AKHIR

Disusun untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Jurusan Sistem Informasi
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:

MUHAMMAD AFIF HENDRAWAN
5210 100 016

Surabaya, Februari 2015

KETUA
JURUSAN SISTEM INFORMASI

Dr. Eng. Febriliyan Samopa, S.Kom, M.Kom
NIP.197302191998021001

PENGEMBANGAN APLIKASI *BASIC TAEKWONDO TRAINING SYSTEM* BERBASIS TEKNOLOGI *REAL TIME MOTION CAPTURE* MENGGUNAKAN MICROSOFT KINECT

TUGAS AKHIR

Disusun untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Jurusan Sistem Informasi
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:

MUHAMMAD AFIF HENDRAWAN
5210 100 016

Disetujui Tim Penguji: Tanggal Ujian : Ferbuari 2015
Periode Wisuda : September 2015

Dr. Eng. Febriliyan Samopa, S.Kom., M.Kom. **(Pembimbing I)**

Nisfu Asrul Sani, S.Kom., M.Sc. **(PembimbingII)**

Faizal Johan Atletiko, S.Kom., M.T. **(Penguji I)**

Radityo Prasetianto W, S.Kom., M.Kom. **(Penguji II)**

PENGEMBANGAN APLIKASI *BASIC TAEKWONDO TRAINING SYSTEM* BERBASIS TEKNOLOGI *REAL TIME MOTION CAPTURE* MENGGUNAKAN MICROSOFT KINECT

Nama Mahasiswa	**:**	**Muhammad Afif Hendrawan**
NRP	**:**	**5210 100 016**
Jurusan	**:**	**Sistem Infomasi FTIf-ITS**
Dosen Pembimbing	**:**	**Dr.Eng. Febriliyan Samopa, S.Kom., M.Kom.**
		Nisfu Asrul Sani, S.Kom, M.Sc.

ABSTRAK

Tingginya tingkat kriminalitas yang terjadi di Indonesia mendorong masyarakat untuk belajar bela diri. Akan tetapi keterbatasan waktu dan tempat membuat sebagian masyarakat tidak dapat berlatih bela diri ditempat latihan yang sudah ada. Namun dengan teknologi saat ini, dimungkinkan untuk masyarakat berlatih bela diri dengan pelatih virtual di rumah. Selain kepraktisan yang ditawarkan, penggunaan teknologi juga dapat mengukur ketepatan gerakan yang di latih.

Memanfaatkan fitur motion capture pada sensor Microsoft Kinect, penulis akan membangun aplikasi "basic taekwondo training system" untuk menciptakan pelatih bela diri virtual yang dapat dimanfaatkan untuk berlatih taekwondo didalam rumah.

Dengan "basic taekwondo training system" penulis berharap dapat memberikan solusi bagi masyarakat yang ingin membekali diri dengan bela diri guna memenuhi

kebutuhan berolahraga dan mengantisipasi tindakan yang tidak dinginkan.

Kata kunci: motion capture, taekwondo, Microsoft Kinect

DEVELOPMENT OF BASIC TAEKWONDO TRAINING SYSTEM USING REAL TIME MOTION CAPTURE TECHNOLOGI WITH MICROSOFT KINECT

Nama Mahasiswa	**:**	**Muhammad Afif Hendrawan**
NRP	**:**	**5210 100 016**
Jurusan	**:**	**Sistem Infomasi FTIf-ITS**
Dosen Pembimbing	**:**	**Dr.Eng. Febriliyan Samopa, S.Kom., M.Kom.**
		Nisfu Asrul Sani, S.Kom., M.Sc.

ABSTRACT

The level of criminality in Indonesia encouraging people to learn how to defense their self , but the the limitation of time and place make half of them cant practice martial art on available place , with the technology today it is posible for online practice at home , besides of the simplicity that is offered the use of technology could measure the accuracy of the motion which is trained.

Using Microsoft Kinect, the author want to develop basic taekwondo traning system to create virtual couch. By using this application, user can train taekwondo everywhere and everytime.

With basic taekwondo training system, the author hope that it could give the solution for the people who wants to prepare them self with martial art to fulfill the needs of sports and anticipate the bad people.

Keyword: motion capture, taekwondo, Microsoft Kinect

KATA PENGANTAR

Segala puji dan syukur penulis tuturkan ke hadirat Allah SWT yang telah memberikan kekuatan dan kehidupan untuk penulis sehingga penulis dapat menyelesaikan laporan tugas akhir dengan judul:

PENGEMBANGAN APLIKASI *BASIC TAEKWONDO TRAINING SYSTEM* BERBASIS TEKNOLOGI *MOTION CAPTURE* MENGGUNAKAN MICROSOFT KINECT

Tugas akhir ini tidak akan terwujud tanpa dukungan dan bantuan dari berbagai pihak. Pada kesempatan ini penulis ingin mengucapkan rasa terima kasih yang sebesar-besarnya kepada pihak-pihak yang mendukung dan membantu proses pengerjaan tugas akhir hingga penyusunan laporan tugas akhir ini. Ucapan terima kasih penulis sampaikan pada,

1. Bapak dan ibu, Bapak Joko Supriyanto dan Ibu Jumala yang selalu mendukung, mendoakan, memberi semangat, dan menasehati penulis selama pengerjaan dan penyusunan laporan.
2. Bapak Febriliyan Samopa selaku dosen pembimbing I, dan Bapak Nisfu Asrul Sani selaku dosen pembimbing II yang bersedia meluangkan waktu dan tenaga untuk membimbing dan mengarahkan penulis dalam pengerjaan tugas akhir.
3. Dosen-dosen laboratorium e-Bisnis yang telah memberikan ilmu dan bimbingan mengenai riset laboratorium e-Bisnis.
4. Mas Bambang Yudoyono selaku pembimbing dalam tugas penulis sebagai asisten laboratorium e-Bisnis yang

selalu mengingatkan penulis pada saat pengerjaan tugas akhir.

5. Bapak, Ibu dosen, dan staf Jurusan Sistem Informasi ITS yang telah menjadi pendidik dan pengajar serta memberikan banyak ilmu yang bermanfaat sejak penulis melanjutkan jenjang pendidikan di Jurusan Sistem Informasi ITS.
6. Sahabat-sahabat penulis yang super, Imam, Rio, Aji, Ebi, Ilham, Adhika, Yogia, Lutfi "Blantik", Lutfi "Shading", Allen, Ijal, Dewa, Yan, Litasya, Amira, dan Inge yang selalu menjaga semangat penulis mulai dari pengerjaan hingga penyusunan laporan tugas akhir. Tidak lupa ucapan terima kasih penulis karena senantiasa menjaga kesehatan penulis dengan makanan-makanan yang bergizi.
7. Tim asisten laboratorium e-Bisnis, Imam, Leo, dan Mas Bambang yang selalu menjadi pemicu penulis untuk mempelajari hal baru.
8. Teman-teman dan senior di laboratorium e-Bisnis yang telah terlebih dahulu sukses, Dilo, Fino, Winny, Eki, Yudha, Ivo, Ichan, Mas Hafiz, Mas Aha dan Mas Satrio yang memberikan semangat untuk tetap betah di lab sejak awal bergabung dengan laboratorium e-Bisnis.
9. Keluarga ke 2 yang penulis cintai, FOXIS, yang selalu menemani dan mendukung penulis sejak pertama menimba ilmu di Jurusan Sistem Informasi.

Juga terima kasih yang sebesar-besarnya kepada semua pihak yang tidak dapat penulis sebutkan satu per satu yang secara langsung maupun tidak langsung membantu penulis dalam pengerjaan dan penyusunan laporan tugas akhir ini. Semoga waktu, tenaga, dan ilmu yang diberikan mendapatkan balasan yang besar dari Allah SWT.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR POTONGAN KODE

BAB I
PENDAHULUAN

Pada bab ini, akan dijelaskan tentang Latar Belakang Masalah, Perumusan Masalah, Batasan Masalah, Tujuan Tugas Akhir, dan Relevansi atau Manfaat Kegiatan Tugas Akhir.

1.1 Latar Belakang Masalah

Tingkat kriminalitas di Indonesia hingga saat ini masih terbilang cukup besar. Pada tahun 2013 tercatat oleh kepolisian, terjadi 305.708 kasus. Meski jumlah ini menurun 10,26% dari tahun 2012 angka tersebut masih terbilang cukup besar. Perlu diingat, jumlah tersebut merupakan tindakan kriminal yang ditangani oleh kepolisian saja (Muhammad, 2013). Hal ini mendorong masyarakat menyadari pentingnya memiliki ilmu bela diri. Mulai dari bela diri praktis hingga bela diri untuk prestasi olahraga mulai digemari dari berbagai kalangan, usia, maupun gender.

Masalah muncul ketika masyarakat tidak mempunyai banyak waktu untuk melakukan latihan bela diri. Berlatih dengan menggunakan video merupakan salah satu cara yang populer. Namun penggunaan media video tidak dapat diukur sehingga sulit mengetahui apakah gerakan yang dilatih benar atau tidak. Diperlukan sistem yang dapat menjadi media pelatihan bela diri yang dapat memberikan contoh gerakan yang benar, sekaligus dapat memonitoring gerakan yang dilakukan.

Teknologi yang dapat digunakan untuk hal tersebut adalah *motion capture*. *Motion Capture* pertama kali digunakan untuk kebutuhan pengembangan *game* dan film animasi. Akan tetapi teknologi sensor yang digunakan dalam pengembangan game dan film animasi terlalu mahal untuk diterapkan dalam penggunaan perorangan. Kemudian muncul teknologi yang lebih murah berupa sebuah kamera yang dapat menangkap gerakan manusian secara *real time*.

Teknologi ini juga pertama kali digunakan dalam ranah pengembangan *game*. Microsoft Kinect merupakan salah satunya. Microsoft Kinect memanfaatkan sensor berupa kamera yang terdiri dari 3 buah kamera sensor, yaitu 2 sensor infra merah masing-masing sebagai *emitter* dan *receiver* dan 1 sensor warna (*RGB / Red, Green, Blue Sensor*) (Catuhe, 2012). Dengan menggunakan ketiga sensor tersebut, Microsoft Kinect dapat berfungsi sebagai *motion capture* untuk membaca gerakan manusia secara *real time*. Microsoft Kinect pada awalnya secara khusus dikembangkan untuk memberikan pengalaman baru dalam bermain *game*. Dengan menggunakan Microsoft Kinect pemain tidak perlu menggunakan *joystick* sebagai kontroler permainan. Pemain hanya perlu menggerakkan bagian tubuh tertentu di depan sensor kamera Microsoft Kinect untuk menggerakkan karakter dalam *game*.

Pada awal pengembangannya, Microsoft Kinect dikhususkan sebagai pelengkap konsol Microsoft XBOX 360. Menyadari bahwa potensi pemanfaatan *motion capture* pada Microsoft Kinectx tidak hanya sebatas memberikan pengalaman baru dalam bermain *game*, banyak tim peneliti dan pengembang melakukan penelitian pemanfaatan Microsoft Kinect pada bidang yang lain. Contohnya dalam bidang kedokteran, Microsoft Kinect dapat dimanfaatkan sebagai media pelatihan gerekan tubuh pasca kecelakaan. Pada bidang seni, Microsoft Kinect dapat dimanfaatkan sebagai trainer dalam mensimulasikan gerakan tari. Karena banyaknya manfaat yang bisa didapat dari *motion capture* Microsoft Kinect, Microsoft mengembangkan *Kinect for Windows software development kit* (SDK) untuk keperluan pengembangan aplikasi menggunakan Microsoft Kinect berbasiskan Windows. Keunggulan *Kinect for Windows SDK* ini adalah merupakan *developer kit* resmi dari Microsoft yang didukung secara penuh oleh Microsoft. Selain dari *Kinect for Windows SDK* juga terdapat beberapa *developer kit* berbasis *open*

source yang dikembangkan untuk memanfaatkan Microsoft Kinect pada platform selain Microsoft XBOX 360 (Windows, Machintos, Linux) yaitu NiTE dan OpenNI yang dikembangkan oleh PrimeSense. Kedua *developer kit* tersebut lebih dulu hadir sebelum *Kinect for Windows* akan tetapi karena dikembangkan oleh kalangan terbatas menjadikan dukukan terhadap *developer kit* tersebut sulit didapatkan dan sulit untuk diimplementasikan.

Memanfaatkan *Kinect for Windows SDK* penulis akan membangun aplikasi *basic taekwondo training system* berbasis Windows. Menurut pelatih kepala Taekwondo Indonesia pengurus cabang Pati, *sabeum nim* Anang Setiawan yang juga penyandang *dan* V Kukkiwon, gerakan dasar untuk taekwondo cukup mudah dilakukan. Pertama dikarenakan gerakan pukulan dan tangkisan seluruhnya mempunyai bentuk tangan menggenggam. Kedua bentuk kuda-kuda yang digunakan juga mudah. Selain itu, gerakan tendangan dasar pada taekwondo juga tidak terlalu rumit, seperti tendangan memutar (*round kick*) atau tendangan menyamping. Analisa yang dilakukan oleh *sabeum nim* Anang berdasarkan pada rangkaian gerakan dasar dalam taekwondo, yaitu poomsae 1, poomsae 2, poomsae 3, taegeuk 1, dan taegeuk 2. Rangkaian gerakan tersebut digunakan dalam melatih taekwondo untuk pemula dan sebagai materi ujian kenaikan tingkat dalam taekwondo pada *geup* 10 (sabuk putih) hingga *geup* 6 (sabuk hijau). Sedangkan gerakan tusukan dan tendangan memutar sudah termasuk dalam gerakan dengan tingkat menengah.

Memanfaatkan fitur *motion capture* dan *skeleton tracking* menggunakan perangkat keras Microsoft Kinect. Selain itu dalam pengembangan aplikasi, penulis juga memanfaatkan Microsoft Visual Studio dan XNA Game Studio sebagai *environment* pengkodean aplikasi serta memanfaatkan *Kinect*

for Windows Developer Toolkit untuk melakukan pengetesan terhadap kode yang telah dibuat.

Dengan memanfaatkan *motion capture* pada Microsoft Kinect penulis ini membangun aplikasi *basic taekwondo training system* dalam bentuk gesture regocniton. Pembangunan aplikasi ini diharapkan dapat membantu mempermudah masyarakat dalam belajar bela diri sehingga diharapkan tindak kekerasan dapat ditekan.

1.2 Rumusan Masalah

Permasalahan yang akan diselesaikan dalam tugas akhir ini adalah,

1. Bagiamana cara memanfaatkan data masukan dari sensor Kinect menggunakan *Kinect for Windows SDK* ?
2. Bagaimana cara merancang dan membangun aplikasi *basic taekwondo training system* memanfaatkan Microsoft Kinect ?
3. Bagaimana cara memonitoring dan memberikan informasi perkembangan pelatihan bela diri menggunakan aplikasi *basic taekwondo training system* ?

1.3 Batasan Masalah

Batasan permasalahan dalam tugas akhir ini adalah.

1. Aplikasi yang dibangun hanya menggunakan satu sensor.
2. Aplikasi yang dibangun hanya mencakup pada perekaman gerakan tubuh (*gesture*) yang dapat dilakukan oleh satu sensor.
3. Gerakan yang digunakan dalam aplikasi adalah gerakan dasar bela diri taekwondo dengan sub batasan,

   a. Berdasarkan pada rangkaian dasar gerakan taekwondo yaitu,
      i. Poomsae 1
      ii. Poomsae 2
      iii. Poomsae 3
      iv. Taegeuk 1
      v. Taegeuk 2
   b. Seluruh gerakan pukulan dan tangkisan diasumsikan dalam tangan posisi menggenggam.
   c. Posisi hadapan kaki tidak diperhitungkan.
4. Aplikasi yang dibangun hanya akan menangkap pergerakan satu orang pengguna saja dan tidak dikembangkan untuk menangkap gerakan banyak pengguna.
5. Aplikasi tidak dibangun untuk dapat mengenali ekspresi wajah maupun perintah suara.
6. Aplikasi yang dibangun tidak mencakup interaksi antar pengguna.
7. Aplikasi dibangun dengan menggunakan *framework* Kinect for Windows SDK versi 1.8, XNA Game Studio versi 4.0, dan .NET versi 4.0
8. Aplikasi dibangun menggunakan sensor Microsoft Kinect for XBOX 360.

1.4 Tujuan Tugas Akhir

Tujuan dari tugas akhir ini adalah memahami dan menerapkan teknologi berbasis *motion capture* menggunakan Microsoft Kinenct dalam penerapannya dalam aplikasi *basic taekwondo training system*. Aplikasi ini nantinya diharapkan mampu menjadi solusi dalam hal pelatihan bela diri secara mandiri dan terukur serta memungkinkan pengguna untuk berlatih bela diri secara *virtual* namun tetap dapat memonitoring keakurasian gerakan.

1.5 Manfaat Tugas Akhir

Dengan pengembangan aplikasi *basic taekwondo training system* diharapakan dapat membantu para perempuan pada khususnya untuk belajar bela diri secara mandiri. Sehingga tingkat kekerasan fisik pada perempuan diharapkan menurun. Selain itu, aplikasi ini dapat dimanfaatkan oleh trainer bela diri profesional sebagai metode pelatihan yang dapat diterapkan dalam tempat latihan, khususnya untuk melatih gerakan jurus seni beladiri.

1.6 Sistematika Penulisan

BAB I
PENDAHULUAN
Pada bab ini akan dibahas tentang latar belakang pengerjaan tugas akhir, perumusan masalah dari latar belakang pengerjaan, batasan masalah pengerjaan tugas akhir, tujuan dan manfaat tugas akhir.

BAB II
TINJAUAN PUSTAKA
Pada bab ini akan dibahas mengenai teori-teori dasar yang menjadi acuan dalam pengerjaan tugas akhir.

BAB III
METODOLOGI
Pada bab ini akan dibahas mengenai alur metodologi penelitian tugas akhir. Alur penelitian dirangkum dalam diagram alur yang akan dibahas tiap tahap.

BAB IV
ANALISIS DAN RANCANGAN APLIKASI
Pada bab ini akan dibahas mengenai analisis dan rancangan aplikasi yang meliputi analisis kebutuhan bisnis, analisis kebutuhan aplikasi, dan rancang bangun pembuatan aplikasi.

BAB V
IMPLEMENTASI DAN UJI COBA SISTEM

Pada bab ini akan dibahas mengenai implementasi dari rancangan yang telah dibahas pada bab analisis dan rancangan aplikasi. Pada bab ini juga akan dibahas mengenai uji coba sistem baik secara fungsional maupun non-fungsional.

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan dibahas mengenai kesimpulan tugas akhir berdasarkan proses uji coba yang telah dilakukan sebelumnya, serta pemberian saran untuk pengembangan selanjutnya.

BAB II
TINJAUAN PUSTAKA

Untuk memudahkan pemahaman tentang apa yang akan dilakukan pada tugas akhir ini, dalam bab ini akan di paparkan tentang konsep dan teknologi apa saja yang akan digunakan atau di terapkan dalam pembuatan tugas akhir.

2.1 Taekwondo

Taekwondo merupakan salah satu beladiri trandisional korea yang berkembang dengan pesat di dunia yang mengajarkan lebih dari sekedar ilmu bela diri. Secara teknik, taekwondo pengajarkan penggunaan *Tae* dan *Kwon* atau tangan dan kaki dalam bela diri. Kedua adalah menjarkan *Do* yang dalam arti kata berarti jalan. *Do* dalam taekwondo mengistilahkan bagaimana cara untuk mengkontrol diri dan selalu damai. Secara keseluruhan, taekwondo dapat diartikan secara filosifi sebagai, "Jalan yang benar dalam menggunakan seluruh tubuh untuk menghentikan pertarungan dan membantu menciptakan kedamaian" (World Taekwondo Federation, 2013).

2.1.1 Gerakan Dasar (*Poomsae*)

Poomsae merupakan set gerakan dasar yang terdapat dalam ilmu bela diri taekwondo. Gerakan-gerakan dasar yang terdapat *poomsae* bisa terdiri dari pukulan, tangkisan, tusukan, maupun tendangan yang mencakup kedalam gerakan serangan atau pertahanan. *Poomsae* dapat dipelajari secara mandiri maupun dengan instruktur. Satu set gerakan *poomsae* memiliki pola yang tetap sehingga mudah untuk dipelajari. *Poomsae* juga menjadi salah satu tolak ukur kenaikan tingkat (geup) pada taekwondo dan memiliki sususan hirarki sebagai berikut,

1. Taegeuk Il Jang (Taegeuk 1).
2. Taegeuk Ye Jang (Taegeuk 2).
3. Taegeuk Sam Jang (Taegeuk 3).
4. Taegeuk Sah Jang (Taegeuk 4).
5. Taegeuk O Jang (Taegeuk 5).
6. Taegeuk Yok Jang (Taegeuk 6).
7. Taegeuk Chil Jang (Taegeuk 7).
8. Taegeuk Pal Jang (Taegeuk 8).
9. Koryo.
10. Keumgang.
11. Taebaek.
12. Pyongwon.
13. Sipjin.
14. Jitae.
15. Chonkwon.
16. Hansu.
17. Ilyeo.

Pada pengembangan aplikasi *basic taekwondo training system* ini akan mengambil gerakan taekwondo yang mengacu pada set gerakan *poomsae taegeuk* 1 hingga 2.

2.2 Optical Motion Sensor

Dalam pengembangan sensor gerak dalam industri perfilman, animasi dan game terdapat 2 jenis sensor gerak yang lazim digunakan, yaitu *marker-based sensor* dan *markerless sensor*.

2.2.1 Marked-based Sensor

Motion capture (mocap) yang berbasis *marked-based sensor* menggunakan penanda yang diletakkan pada sendi-sendi atau bagian lain guna mendukung sensor dalam membaca gerakan yang dilakukan oleh obyek. Salah satu pengembang teknologi ini adalah Motion Analysis. Tugas dari penanda yang digunakan adalah untuk merefleksikan kembali cahaya yang dikirimkan

kepada kamera sensor. Cara ini menggunakan banyak kamera sebagai sensor gerak.

Karena memanfaatkan banyak kamera sebagai sensor dan penanda untuk melakukan *tracking* pada obyek, maka teknologi marked-based sensor ini cenderung lebih mahal digunakan (Wittenberg, 2013).

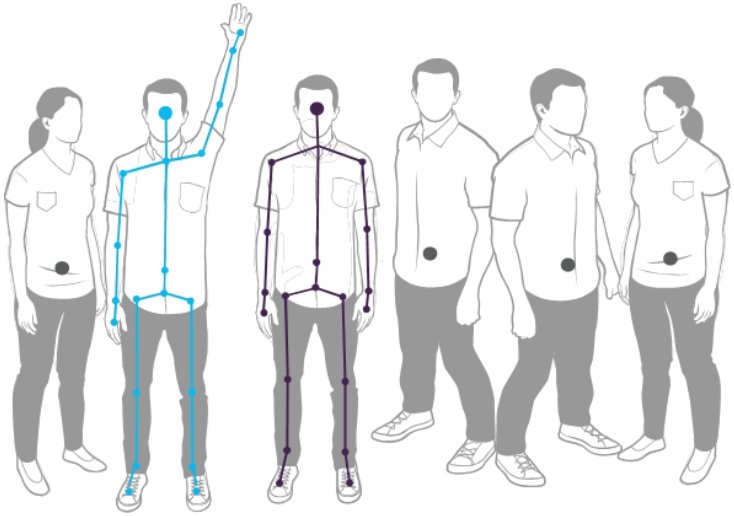
2.2.2 Markerless Sensor

Markerless sensor merupakan teknologi *motion capture* yang tidak membutuhkan penanda (marker) untuk membantu sensor dalam mendeteksi gerakan. *Markerless sensor* menggunakan algoritma tertentu yang tertanam dalam software sensor untuk mengenali sendi-sendi yang dimiliki manusia. Dengan menggunakan sensor ini, dimungkinkan untuk merekam gerakan dengan menggunakan satu sensor kamera ataupun lebih (Wittenberg, 2013). Salah satu sensor yang menerapkan metode *markerless* adalah Microsoft Kinect.

2.3 Microsoft Kinect

Microsoft Kinect merupakan alat input berbasis motion-sensing yang dikembangkan oleh Microsoft. Microsoft Kinect di-launching pertama kali pada bulan November tahun 2010. Microsoft Kinect memiliki sensor berupa 3 buah kamera yang berfungsi sebagai kamera 3 dimensi. Sensor tersebut menangkap pencitraan obyek tiap pixels warna dan membaca kedalaman (depth) tiap pixel-nya (Catuhe, 2012). Microsoft Kinect pada dasarnya menggunakan teknolgi range camera yang dikembangkan oleh PrimeSense. Teknologi tersebut menginterprestasikan obyek yang ditangkap secara 3D menggunakan sinar inframerah (IR) (Berg, Chattopadhyay, Schedel, & Vallier). Teknologi yang dikembangkan oleh PrimeSense juga dapat mengenali manusia berdasarkan struktur tulang yang dimiliki. Struktur tulang yang unik tiap

posisi inilah yang membuat sensor Microsoft Kinet dapat mengenali obyek sebagai manusia.



Gambar 2.1 *Skeleton Tracking* Sensor Kinect

(Sumber : http://i.msdn.microsoft.com/dynimg/IC584841.png)

Microsoft Kinect juga mengenali manusia dengan gerakan-gerakan yang diciptkaan dibandingkan dengan obyek-obyek yang tidak bergerak. Microsoft Kinect akan mengenali manusia dengan sebutan player. Dalam keadaan yang ramai pergerakan manusia yang keluar dan masuk pada area sensor, Microsoft Kinect akan tetap mengenali player dengan mengisolasi dan melakukan tracking pada struktur tulang serta mengabaikan obyek yang lain.

Keunggulan Microsoft Kinect terletak pada teknologi depth-sensing. Inframerah (IR) yang dipancarkan oleh Microsoft Kinect disebarkan keseluruh ruangan dan mengembalikan kembali kepada sensor kamera complementary metal-oxide

semiconductor (CMOS) yang kemudian dilakukan analisa data yang didapatkan oleh sinar Inframerah. Akurasi dari sinar inframerah yang dipancarkan berkisar antara 1,2 hingga 3,5 meter (optimal). Microsoft Kinect juga dibekali dengan RGB Camera yang berfungsi sebagai penangkap event secara real time pada resolusi 640x480 dengan kecepatan 30 frame per second (fps) (Jean, 2012).

Secara keseluruhan, sensor yang terdapat pada Microsoft Kinect adalah,

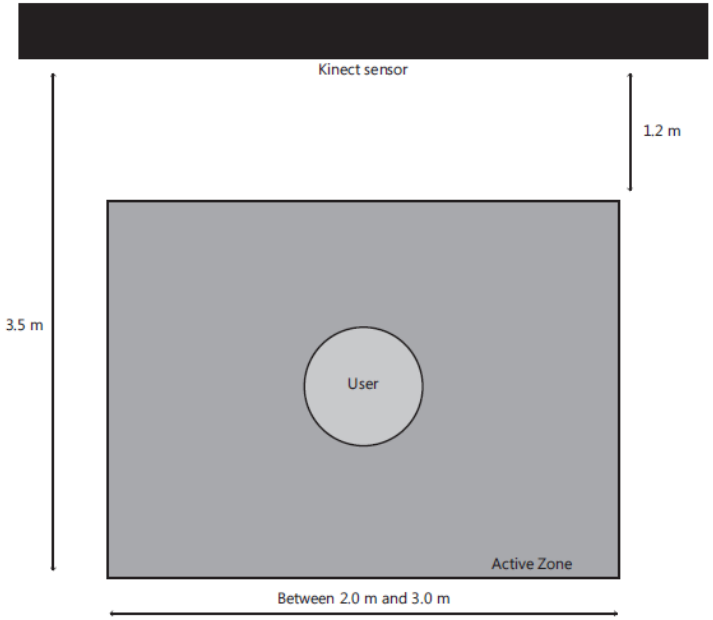
1. *Microphone array*
2. *Infrared emitter*
3. *Infrared receiver*
4. *Color camera (RGB Camera)*

Gambar 2.2 Sensor-sensor Pada Kinect

Sensor pada Microsoft Kinect juga memiliki batasan-batasan tertentu terkait dengan jarak yang dapat dijangkau oleh sensor. Batasan pada Microsoft Kinect adalah sebagai berikut,

1. Sudut pandang horizontal maksimal 57 derajat.
2. Sudut pandang vertikal maksimal 43 derajat.

3. Jarak obyek dengan sensor minimal 1,2 meter (penggunaan *near mode* hingga 0,4 meter) dan jarak terjauh adalah 4 meter (penggunaan *near mode* hingga 3 meter).
4. Rentang kedalaman untuk *depth image*, 400mm (untuk *near mode*) hingga 8000mm (untuk *standard mode*).
5. Temperatur 5 hingga 35 derajat celcius.



Gambar 2.3 Batasan Jarak Pengguna Sensor Kinect

(Sumber : Programming with the Kinect for Windows Development Kit, 2012)

2.4 Microsoft Kinect for Windows

Microsoft Kinect for Windows merupakan sensor berbasiskan Microsoft Kinect untuk konsol XBOX 360 yang dikhususkan untuk *platform* Windows dan dilisensikan dan didukung

untuk kepentingan komersial. Secara umum *Microsoft Kinect for Windows* memiliki kemampuan-kemampuan utama dari Microsoft Kinect untuk XBOX 360 seperti *infrared emitter*, *infrared receiver*, *RGB camera*, dan *multi-array microphone*. Perbedaannya selain dari sisi lisensi dan penggunaan komersial, *Microsoft Kinect for Windows* lebih diutamakan pada pemberian pengalaman baru dalam berinteraksi dengan *natural user interface* (NUI) bagi pengguna Windows dengan fitur khusus seperti *near mode* (minimun jarak obyek dengan sensor hanya 40 centimeter), *skeletal tracking control*, peningkatan *application user interface* (API), serta penggunaan USB sebagai konektor utama (Microsoft, 2013).

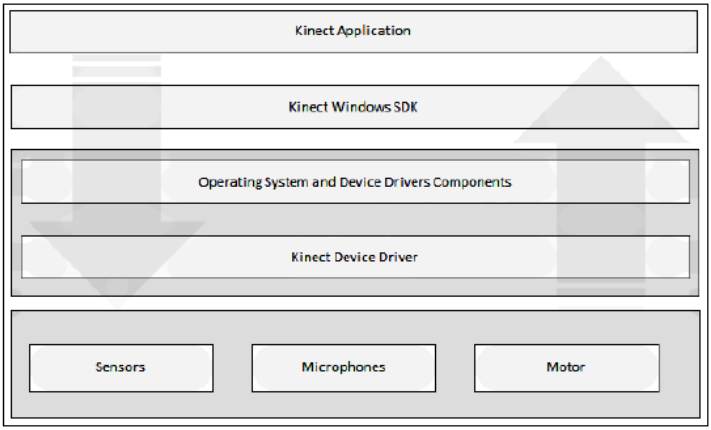
Gambar 2.4 Sensor Kinect for Windows

(Sumber : : http://images.pcworld.com/images/article/2012/01/kinect-for-windows-8667082.jpg)

Sedangkan Microsoft Kinect untuk XBOX 360 memang secara khusus dibuat dan dites hanya untuk konsol XBOX 360, bukan untuk plaform lain sehingga menjadikan alasan mengapa Microsoft Kinect untuk XBOX 360 tidak memiliki lisensi untuk penggunaan komeresial secara general.

2.4.1 Kinect for Windows Software Development Kit (Kinect for Windows SDK)

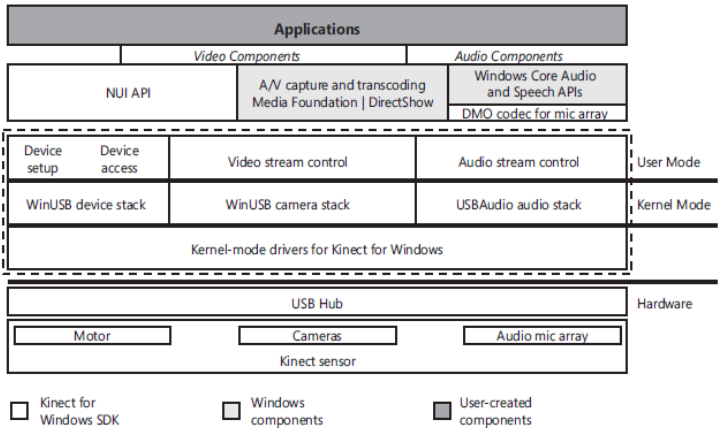
Untuk mengembangkan aplikasi berbasis Microsoft Kinect diperlukan *Kinect for Windows Software Development Kit* (Kinect for Windows SDK). *Kinect for Windows SDK* berfungsi sebagai penghubung (*interface*) antara Microsoft Kinect dengan aplikasi yang akan dibangun. SDK ini berisikan *application programming interface* (API) yang digunakan ketika mengakses fungsi sensor Microsoft Kinect. Kemudian API akan meneruskan perintah ke *driver* untuk menjalankan sensor. *Driver* Kinect berugas untuk mengakses data yang didapatkan oleh sensor (Jana, 2012). Alur *interface* dari *Microsoft Kinect* dapat digambarkan pada Gambar 2.5.



Gambar 2.5 Alur Interface Sensor Kinect

(Sumber : Kinect for Windows SDK Programming Guide, 2012)

Secara arsitektur, penggunaan *Kinect for Windows SDK* digambarkan pada Gambar 2.6.



Gambar 2.6 Arsitektur Kinect for Windows SDK

(Sumber : Programming with the Kinect for Windows Development Kit, 2012)

Pengembangan aplikasi menggunakan *Kinect for Windows SDK* dapat menggunakan perangkat keras *Microsoft Kinect for XBOX 360* maupun *Microsoft Kinect for Windows*. Akan tetapi penggunakan perangkat keras Microsoft Kinect for XBOX 360 akan memiliki ketebatasan dan pihak Microsoft tidak menjamin pengangkat keras akan bekerja dengan kompabilitas penuh (Catuhe, 2012).

Namun dalam pengembangan aplikasi *taekwondo art traning system* ini penulis akan menggunakan perangkat keras *Microsoft Kinect for XBOX 360* dikarenakan alasan produk yang dihasilkan bukan untuk diproduksi masal dan hanya dalam ranah penelitian serta kemudahan dalam mendapatkan perangkat keras.

Kinect for Windows SDK mendukung pengembangan aplikasi menggunakan 3 bahasa pemograman yaitu C#,C++, dan Visual Basic.

2.4.1.1 Video Stream

Video stream merupakan data pertama yang dihasilkan oleh sensor Kinect. Meskipun memiliki fungsi sebagai kamera 3 dimensi, video stream meruapakan data *basic* dari sensor Kinect. Video stream menggunakan resolusi dan *frame rates* sesuai dengan standard sebagai berikut,

1. 640 x 480 pada 30 *frames per second* (fps) menggunakan format *red, green, blue* (RGB).
2. 1280 x 480 pada 12 fps menggunakan format RGB.
3. 640 x 480 pada 15 fps menggunakan format YUV (atau raw YUV).

Format YUV merupakan format dengan kepadatan warna 16-*bit* yang lebih efektif dibandingkan dengan format RGB yang memiliki kepadatan warna 36-*bit* per *pixel*, sehingga menjadikan format YUV lebih efektif karena *driver* mengalokasikan penggunaan memori yang lebih sedikit (Catuhe, 2012).

2.4.1.2 Depth Stream

Depth stream merupakan data *pixel* yang dihasilkan oleh sensor Kinect yang berisikan jarak sensor dengan obyek terdekat dalam satuan milimeter (Catuhe, 2012). Aplikasi menggunakan *depth stream* untuk mendeteksi

gerakan manusia atau mengidentifikasi obyek yang diabaikan (Microsoft, 2014).

Depth stream menggabungkan dua tipe data yang berbeda yaitu,

1. Data depth dalam milimeter, dan
2. Data segmentasi *players* (*player segmentation data*). Tiap nilai *player* berupa data integer yang mengindikasikan index dari setiap *player* yang bersifat unik dalam satu *scene* (Microsoft, 2014).

Resolusi yang didukung dari fungsi *depth stream* adalah,

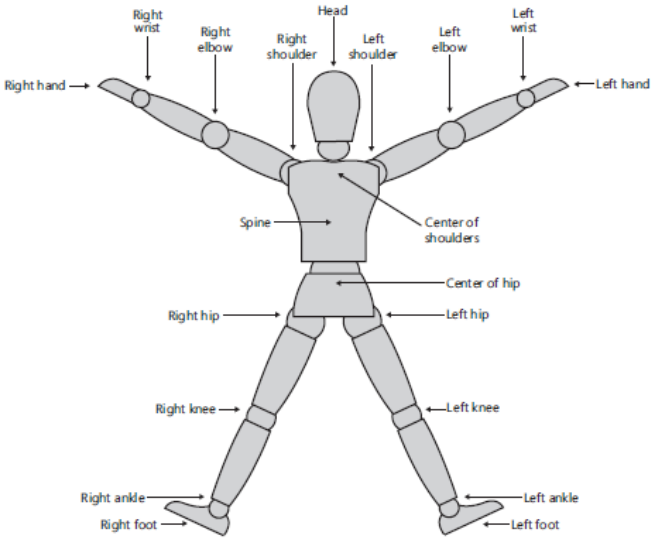
1. 640 x 480
2. 320 x 240
3. 80 x 60.

Ketiga resolusi yang didukung tersebut selurunya menggunakan *frames rate* sebesar 30 fps (Catuhe, 2012).

2.4.1.3 Skeleton Tracking

Skeletal tracking pada sensor kinect menggunakan *natural user interface* API (NUI API) pada *depth stream* untuk mendeteksi manusia yang berada di depan sensor. *Skeletal tracking* dioptimasikan untuk mengenali pengguna yang berada di depan sensor Kinect. Sensor Kinect mampu mengenali hingga 6 pengguna sekaligus dan melakukan *skeletal tracking* pada tiap pengguna.

Tiap pengguna yang tertangkap sensor, NUI akan memproduksi satu set *key point* yang saling berhubungan disebut *skeleton*. *Skeleton* terdiri dari 20 titik yang menggambarkan sendi-sendi dari tubuh manusia.

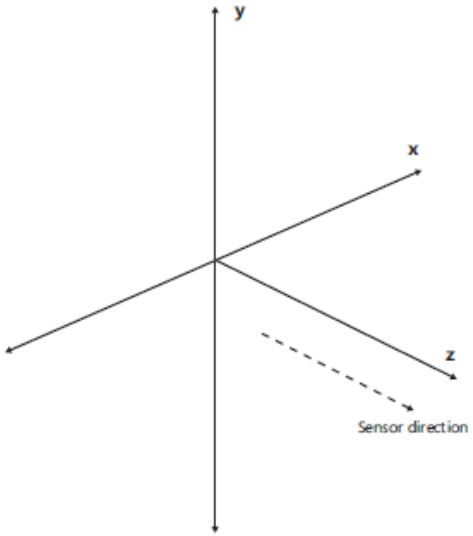


Gambar 2.7 20 Titik Sendi Sensor Kinect

(Sumber : Programming with the Kinect Software Development Kit, 2012)

Tiap point mendefinisikan posisi koordinat dalam prespektif 3 dimensi (x,y,z). Koordinat yang digunakan menggunakan satuan meter yang merujuk pada satuan milimeter yang dihasilkan oleh *depth stream*. Ruang lingkup *skeleton* didefinisakan sebagai arena seketar sensor yang memiliki koordinat dasar pada (0,0,0). Sumbu X dalam ruang lingkup *skeleton* menggambarkan gerakan ke kenan

dan ke kiri dari sudut pandang pengguna. Sumbu Y menggambarkan gerakan ke atas, sedangkan sumbu Z menggambarkan jarak pengguna dengan sensor (Catuhe, 2012).



Gambar 2.8 Sudut Pandang Sensor Kinect

(Sumber : Programming with the Kinect Software Development Kit, 2012)

2.4.2 Kinect for Windows Developer Toolkit

Kinect for Windows Developer Toolkit merupakan paket opsional dari *Kinect for Windows SDK* yang berisikan kode sampel dan penggunaan *Kinect for Windows SDK*. Dengan menggunakan *toolkit* ini pengembang aplikasi dapat dengan lebih cepat dan mudah dalam membangun *interface*. Contoh yang diberikan dari *toolkit* masing-masing memiliki masalah yang berbeda-beda yang dapat memberikan pengalaman baru dalam pemanfaatan sensor Kinect

pada platform Windows (Microsoft, 2013). Kode sampel yang diberikan dalam *toolkit* juga terdiri dari 3 bahasa pemograman yang didukung *Kinect for Windows SDK* yaitu C#,C++, dan Visual Basic.

2.4.2.1 Kinect Studio

Kinect studio merupakan *tool* dalam *bundling Kinect for Windows SDK* yang bermanfaat untuk melakukan perekaman dan pemutaran ulang pada data *depth* dan *color stream* yang di ambil oleh sensor Kinect. Kinect studio juga dapat digunakan untuk melakukan *debugging*, reka ulang skenario untuk testing, dan menganalisa performa dari data yang didapat oleh sensor Kinect (Microsoft, 2014).

2.5 Microsoft Visual Studio

Microsoft Visual Studio merupakan *integrated development environment* (IDE) yang digunakan untuk membangun aplikasi ASP.NET (web), *XML Web Services*, dekstop, dan *mobile*. Microsoft Visual Studio mendukung bahasa pemograman berbasis Visual Basic, C#, dan C++ (Microsoft, 2010). Penulis akan menggunakan IDE Visual Studio dalam mengembangkan aplikasi *basic taekwondo traning system* karena dukungannya terhadap bahasa C# dan C++ yang lazim di gunakan dalam pengembangan aplikasi menggunakan sensor Kinect.

2.6 XNA Game Studio

XNA Game Studio merupakan *framework* yang didesain untuk membuat game berbasis Microsoft Windows, XBOX 360, dan Windows Phone. XNA Game Studio sendiri juga berkaitan dengan Microsoft Visual Studio dalam hal dukungan *framework* dan *tools*. *Framework* dari XNA terdiri dari *library-library* yang dikhususkan dalam pengembangan *game*. Dalam paket instalasi XNA Game Studio juga terdapat

tools untuk mengolah grafis dan audio yang dibutuhkan dalam sebuah *game* (Microsoft, 2010).

XNA Game Studio *framework* didasarkan pada framework Microsoft .NET sehingga XNA Game Studio dapat memanfaatkan XNA Game Studio *framework* sendiri ataupun memanfaatkan *framework* yang dimiliki oleh .NET. Bahasa pemograman standard yang digunakan oleh XNA Game Studio adalah C#, namun secara teori, dapat juga menggunakan C++ ataupun Visual Basic (VB).

2.7 Nuclex Framework

Nuclex framework merupakan komponen tambahan dari pihak ketiga untuk memberikan fitur tambahan yang tidak dimiliki oleh XNA. Beberapa fitur yang tawarkan *Nuclex framework* bagi pengembang adalah,

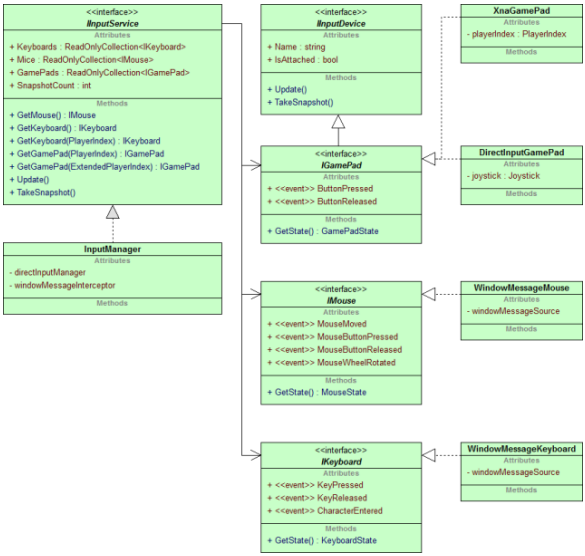
1. *Game State Management*
2. *GUI Management*
3. *Input Control*
4. *Lemper-Ziv-Makrov (LZMA) Algorithm*
5. *Particles*
6. *Vector Font*

Penulis memanfaatkan *Nuclex framework* untuk kebutuhan fitur GUI dan *keyboard input*. GUI yang dimaksudkan adalah penggunaan *form* dan *button* sedangkan *keyboard input* adalah menanggap *event typing* dari pengguna. Dengan memanfaatkan GUI dan *keyboard input* dari pihak ketiga ini, penulis dapat dengan lebih mudah memasukkan fitur tersebut dibandingkan dengan harus melakukan integrasi antara XNA dengan *windows presentation foundation* (WPF).

2.7.1 Input Control

Meskipun XNA sudah menyediakan *input handling* baik untuk *mouse*, *keyboard*, dan *joystick* namun masih ada satu hal yang belum dimiliki oleh XNA,

yaitu kemudahan dalam meng*handle* proses mengetik yang dilakukan oleh pengguna. XNA dapat melakukan hal tersebut dengan mamasukkan seluruh karakter pada *keyboard* kedalam sebuah *dictionary*. Tentu bukanlah hal yang efektif dan cepat dilakukan. Dengan menggunakan *input control* dari *Nuclex framwork* pengembang dapat dengan mudah mengatasi proses mengetik dari pengguna dapat membuat *dictionary* tiap karakter dari *keyboard*. Selain itu fitur ini dapat mempersingkat deklarasi untuk *event* menekan dan melepas tombol daripada cara deklarasi yang dimiliki oleh XNA.



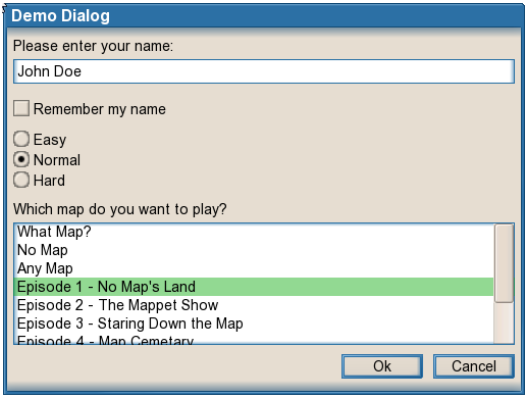
Gambar 2.9 Desain Class Library InputControl Nuclex Framework

(Sumber : http://blog.nuclex-games.com/2010/08/new-component-nuclex-input/)

Untuk menggunakan fitur ini, pengembang harus memasukan *library* Nuclex.Input pada aplikasi yang dikembangkan.

2.7.2 GUI Management

Fitur GUI *management* memungkinkan pengembang aplikasi yang menggunakan XNA untuk menambahkan fitur *form* dan *button* pada aplikasi yang dikembangkan. Sebelumnya, secara langsung XNA tidak menyediakan fitur untuk membuat *form*. Sedangkan fitur *button*, pengembang harus lebih kreatif menciptakan *button* dengan gambar. Hal ini dikarenakan pengembangan XNA yang lebih mengarah pada pengembangan aplikasi berbasis konsol XBOX. Dengan *Nuclex framework* penambahan fitur tersebut dapat dilakukan dengan lebih cepat sehingga memudahkan pengembang aplikasi XNA untuk ranah Windows yang seringkali membutuhkan *form* dan *button*.



Gambar 2.10 Contoh GUI Nuclex Framework

(Sumber : http://nuclexframework.codeplex.com/wikipage?title=Nuclex.User Interface&referringTitle=Home)

Untuk memanfaatkan fitur ini pengembang perlu menambakan *library* Nuclex.UserInterface.dll pada aplikasi yang dikembangkan.

2.8 Perangkat Lunak Penunjang

Dalam pengembangan aplikasi, penulis juga memerlukan beberapa perangkat lunak penunjang guna melengkapi kebututuhan pengembangan. Diantaranya adalah perangkat lunak pengolah model tiga dimensi dan perangkat lunak pengolah gambar.

2.8.1 3ds Max

Pengolahan model 3 dimensi yang akan digunakan dalam pengembangan aplikasi *basic taekwondo training system* menggunakan perankgat lunak 3ds MAX. 3ds Max merupakan perangkat lunak pengolah model 3 dimensi yang dapat digunakan untuk membuat model 3 dimensi untuk kebutuhan modeling, animasi, simulasi dan *rendering* (Autodesk, 2014). Penulis memanfaatkan 3ds Max untuk melakukan permodelan karakater dan *rigging* yang dibutuhkan dalam pengembangan aplikasi.

2.8.1.1 *Rigging*

Rigging merupakan salah satu teknik animasi 3 dimensi dengan menanamkan susunan tulang digital pada model yang akan di animasikan. Seperti halnya tulang pada makluk hidup, *teknik rigging* menciptakan tulang dan sendi pada model sehingga animator dapat menggerakan model dalam berbagai pose (Slick). Proses *rigging* sendiri secara umum diawali dengan penempatan tulang digital pada model sesuai dengan

kebutuhan, kemudian dilanjutkan dengan menyusun hirarki dari tulang-tulang yang dibuat. Kemudian dilakukan pembobotan pada tulang dan pengaturan efek pada *mesh* model.

2.8.1.2 *Skin Modifier*

Skin modifier merupakan tool untuk melakukan deformasi atau modifikasi pada *mesh* yang dimiliki model. *Skin modifier* sangat bermanfaat pada animasi model 3 dimensi karena dapat menyesuaikan bentuk *mesh* sesuai dengan pergerakan atau pose model . Perubahan pada *mesh* menggunakan *skin modifier* dipengaruhi oleh bobot struktur tulang dan sendi digital yang dibenamkan dalam model 3 dimensi (3ds Max Tutorial).

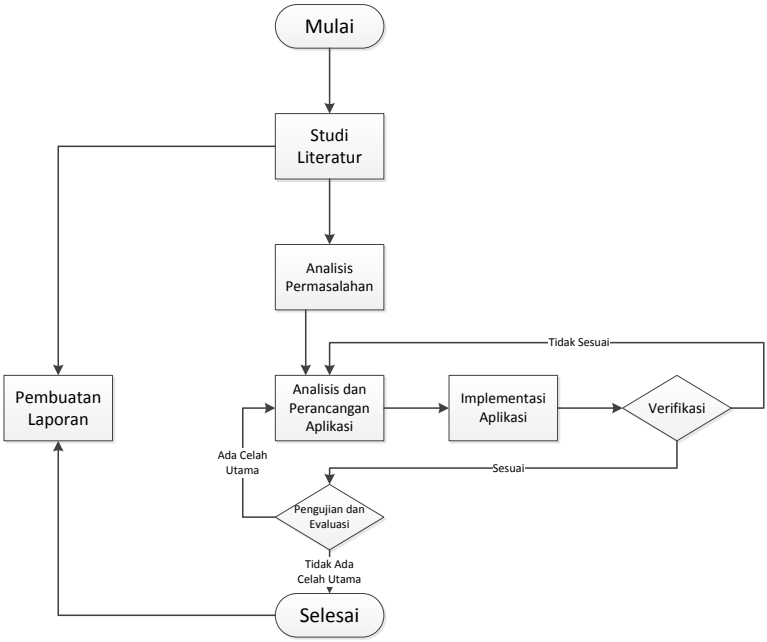
2.8.1.3 Format Model FBX

FBX merupakan format model 2 dimensi dan 3 dimensi yang dikembangkan oleh Autodesk. Keunggulan dari model FBX adalah dapat digunakan dalam berbagai macam platform. FBX merupakan salah satu format model 3 dimensi yang didukung oleh Microsoft Visual Studio selain X (directX).

Halaman sengaja dikosongkan.

BAB III
METODOLOGI

Pada bab ini dibahas mengenai obyek penelitian serta langkah-langkah penelitian yang dilakukan. Pengembangan aplikasi *basic taekwondo training system* diawali dengan studi literatur, kemudian dilanjutkan dengan analisis permasalahan, analisis dan perancangan aplikasi, implementasi aplikasi, verifikasi aplikasi, dan diakhiri dengan pengujian dan eveluasi. Diagram alur pengembangan aplikasi *basic taekwondo training system* dapat dilihat pada Gambar 3.1.



Gambar 3.1 Alur Pengembangan Aplikasi

Obyek penelitian yang digunakan sebagai tugas akhir ini adalah pemanfaatan teknologi Microsoft Kinect sebagai sensor *motion*

capture dengan studi kasus gerakan dasar pada ilmu bela diri taekwondo. Terdapat pula batasan-batasan penelitian yang menjadi perhatian utama dikarenakan adanya batasan yang dimiliki oleh Microsoft Kinect dan *software development kit* (SDK) yang digunakan dalam penelitian. Hal terpenting dalam penelitian tugas akhir ini adalah pengguna dapat melakukan latihan dasar ilmu bela diri taekwondo dengan memanfaatkan Microsoft Kinect.

3.1 Studi Literatur

Pada tahap studi literatur penulis melakukan studi mengenai obyek penelitian tugas akhir dan metode yang digunakan, yaitu,

1. Mempelajari gerakan taekwondo secara umum.
2. Pemahaman algoritma yang digunakan oleh Kinect for Windows SDK dalam pendeteksian kerangka pengguna.
3. Memahami proses pengintegrasian antara Kinect for Windows SDK dengan XNA Game Studio.
4. Perancangan pengintegrasian antara Kinect for Windows SDK dengan XNA Game Studio.
5. Pemahaman pemanfaatan kerangka kerja XNA Game Studio dengan IDE Visual Studio.
6. Pemahaman penggunaan model tiga dimensi dan pengaplikasiannya pada XNA Game Studio.
7. Mempelajari cara perekaman gerakan pengguna menggunakan sensor Microsoft Kinect.
8. Mempelajari gerakan taekwondo yang mampu dibaca oleh sensor Microsoft Kinect.
9. Mempelajari pemanfaatan hasil rekam gerakan untuk kebutuhan sinkronisasi gerakan (*motion regocnition*).

3.2 Analisis Permasalahan

Pada tahap ini akan dilakukan analisis terhadap masalah yang diangkat dalam studi tugas akhir ini seperti batasan gerakan

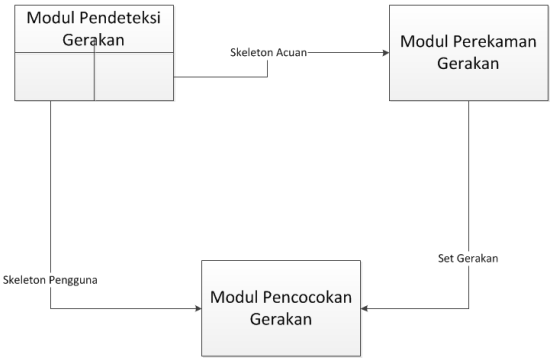
taekwondo yang mungkin dideteksi oleh sensor Kinect serta batasan yang dimiliki oleh Kinect sendiri.

3.3 Analisis dan Rancangan Aplikasi

Pada tahap ini akan dilakukan analisis awal dan pendefinisian kebutuhan sistem untuk mengetahui masalah yang ada dalam studi tugas akhir. Dilanjutkan dengan tahap perancangan aplikasi yang dilakukan dengan memanfaatkan sensor *Microsoft Kinect for XBOX 360*, Kinect for Windows SDK, Visual Studio, dan XNA Game Studio. Parancangan aplikasi meliputi 3 modul utama yaitu,

1. Modul pendeteksian gerakan pengguna secara *real time*.
2. Modul perekaman gerakan taekwondo.
3. Modul analisa gerakan pengguna dengan gerakan model.

Keterkaitan antar modul dalam pengembangan aplikasi ditunjukkan pada Gambar 3.2.



Gambar 3.2 Skema Keterkaitan Antar Modul

3.4 Implementasi dan Uji Coba Sistem

Pada tahap ini akan dilakukan pembangunan aplikasi secara menyeluruh berdasakan desain yang telah dilakukan pada

tahap analisis dan perencanaan. Bahasa pemograman yang digunakan adalah C# dengan menggunakan kerangka kerja Kinect for Windows SDK dan XNA. Implentasi aset model tiga dimensi menggunakan aplikasi pengolah model tiga dimensi. Pembangunan aplikasi dibagi menjadi 3 tahap utama sesuai dengan model yang direncanakan, yaitu,

1. Modul pendeteksi gerakan *realtime*.
    a. Pendeteksian gerakan pemain kemudian dirubah dalam bentuk informasi sendi-sendi kerangka (skeleton).
    b. Transformasi informasi sendi-sendi kerangka pemain menjadi model tiga dimensi dengan menghitung rotasi dan orientasinya.
    c. Hasil informasi koordinat sendi-sendi dari *stream* sensor Kinect dirubah dalam koordinat XNA Game Studio.
    d. Pemasangan koordinat sendi XNA kedalam model tiga dimensi.
2. Modul perakaman gerakan taekwondo.
    a. Mengatur informasi gerakan taekwondo.
    b. Melakukan perekaman gerakan.
    c. Penyimpanan detail set gerakan.
    d. Menyimpan seluruh gerakan.
3. Modul analisa gerakan pengguna dengan gerakan model tiga dimensi.
    a. Merancang sistem sinkronisasi antara gerakan model hasil rekam dengan gerakan realtime dari pengguna.
    b. Melakukan analisa ketepatan gerakan yang dilakukan oleh pengguna berdasarkan gerakan model.

3.5 Verifikasi

Pada tahap ini akan dilakukan pengecekan kembali apakah implementasi aplikasi sudah sesuai dengan analisis dan perancanaan aplikasi yang telah dibuat pada tahap sebelumnya.

3.6 Pengujian dan Evaluasi

Pengujian dilakukan pada sebelum integrasi dan setelah integrasi. Pengujian ini dilakukan untuk mencari celah yang ada pada aplikasi, bila terdapat celah pada aplikasi maka akan kembali ke tahap implementasi dan kemudian akan dilakukan pengujian kembali. Evaluasi keberhasilan aplikasi ini adalah hasil analisa gerakan yang dilakukan oleh pengguna dan ketepatan gerakan yang diperagakan oleh model dengan gerakan hasil perekaman gerakan taekwondo.

3.7 Pembuatan Laporan

Pembuatan laporan adalah tahap di mana membuat semua dokumtentasi dari pengembangan aplikasi, berupa langkah-langkah dan juga analisa. Laporan berguna untuk memberikan informasi berupa pemahaman proses pengembangan aplikasi *basic taekwondo training system* serta apakah pengembangan aplikasi mampu memecahkan permasalahan yang ada.

Halaman ini sengaja dikosongkan.

BAB IV
ANALISIS DAN RANCANGAN APLIKASI

Pada bab ini akan dijelaskan mengenai analisis dan rancangan sistem aplikasi yang dibangun pada tugas akhir ini. Secara keseluhuran sistem aplikasi terdiri dari 3 modul utama yaitu, pendeteksi gerakan, perekaman gerakan, dan pencocokan gerakan menggunakan sensor kinect. Ketiga modul tersebut akan dibahas dalam bab ini secara terperinci.

4.1 Tahap Analisis

Pada sub bab ini akan dibahas mengenai proses-proses yang dilakukan dalam tahapan analisis aplikasi. Analisis akan dibagi menjadi analisis permasalahan dan analisis kebutuhan perangkat lunak.

4.1.1 Analisis Permasalahan

Beladiri saat ini mulai menjadi trend dikalangan masyarakat urban. Selain digunakan untuk kebutuhan berolahraga, beladiri juga dimanfaatkan untuk kebutuhan membela diri dalam keadaan mendesak. Hal ini yang menjadi asalan masyarakat urban mempelajari berbagai macam beladiri mulai dari beladiri sederhana hingga beladiri yang dapat dikategorikan sebagai beladiri olahraga prestasi. Salah satu beladiri yang cukup banyak memiliki penimat dan tumbuh pesat di daerah perkotaan adalah taekwondo.

Taekwondo merupakan cabang beladiri yang berasal dari Korea. Taekwondo merupakan beladiri yang memanfaatkan tangan dan kaki dengan gerakan yang cukup cepat. Meskipun demikian gerakan-gerakan dasar taekwondo tidak terlalu rumit dan dapat dilatih secara mandiri serta tidak memerlukan tempat yang

luas. Hal ini membuat masyarakat urban dapat berlatih taekwondo di mana saja.

Menanggapi hal tersebut dibutuhkan solusi bagaiama masyarakat dapat berlatih gerakan taekwondo dimana saja dan kapan saja tanpa harus menggangu kesibukan yang lainnya. Selain itu, bagaimana solusi ini dapat meminimalkan kesalahan dalam berlatih gerakan taekwondo bagi yang menggunakan. Karena itu aplikasi *basic taekwondo training system* dipilih sebagai solusi yang diangkat dalam tugas akhir ini. Aplikasi *basic taekwondo training system* memanfaatkan teknologi pendeteksi gerakan atau *motion capture* yang dimiliki oleh sensor kinect. Kinect mampu membaca gerakan penggunanya dan dijadikan sebagai masukan (*input*) dari sistem aplikasi. Dengan teknologi ini, gerakan dasar taekwondo ditangkap dan diolah oleh sistem kemudian memberikan keluaran (*output*) berupa nilai dari presisi gerakan yang dilakukan. Gerakan dasar taekwondo dipilih karena mudah dilakukan dan karena keterbatasan sensor yang belum sempura dalam membaca gerakan yang saling tumpang tindih.

4.1.2 Analisis Kebutuhan Bisnis

Berdasarkan solusi permasalaah yang ditawarkan, selanjutnya dilakukan analisis kebutuhan bisnis dari perangkat lunak yang akan dikembangkan. Untuk mendapatkan kebutuhan bisnis dari perangkat lunak, penulis menggunakan teknik studi literatur dari paper-paper serta perangkat lunak sejenis. Dari studi literatur didapatkan kebutuhan bisnis perangkat lunak pelatihan olahraga menggunakan sensor gerak, yaitu,

1. Menampilkan posisi tubuh pengguna. (Vij & Dawn, 2014)

2. Membaca kecepatan gerak pengguna. (Vij & Dawn, 2014)
3. Pengguna mengetahui gerakan yang dilakukan secara langsung. (Nike, 2012)
4. Pengguna memiliki trainer virtual sebagai pemandu latihan. (Nike, 2012)
5. Pengguna mendapatkan *feedback* latihan secara langsung. (Nike, 2012)
6. Monitoring perkembangan latihan. (Nike, 2012)

4.1.3 Analisis Spesifikasi Kebutuhan Perangkat Lunak

Dari hasil analisis arsitektur sistem, maka secara garis besar kebutuhan fungsional yang akan dipenuhi oleh sistem perangkat lunak adalah,

1. Proses penangkapan gerakan pengguna.
   Proses ini membaca kerangka pengguna dengan menggunakan SDK Microsoft Kinect dan menghasilkan informasi kerangka pengguna.
2. Proses pembacaan struktur kerangka model tiga dimensi.
   Pada proses ini kerangka yang dimiliki oleh model tiga dimensi dibaca oleh sistem dan akan diolah bersamaan dengan karangka pemain.
3. Proses transformasi kerangka.
   Kerangka milik pengguna yang masih dalam ruang sensor kinect, akan ditransformasikan oleh fungsi-fungsi transformasi *framework* XNA agar sesuai dengan ruang tiga dimensi XNA.

4. Proses menangkap dan menampilkan *depth stream*.
   Proses ini akan menangkap gerakan pengguna dengan menggunakan kamera infra merah yang dimiliki oleh sensor kinect. Keluaran dari proses ini adalah gambaran pengguna dalam ruang tiga dimensi yang memiliki kedalaman.
5. Proses animasi model tiga dimensi.
   Pada proses ini, hasil dari proses transformasi kerangka pemain akan dicocokan dengan kerangka model tiga dimensi. Sehingga struktur model tiga dimensi serupa dengan struktur kerangka pengguna memungkinkan model tiga dimensi dapat dianimasikan serupa dengan gerakan pengguna.
6. Proses pemberian nama gerakan.
   Pada proses ini pengguna diminta untuk memasukkan nama gerakan yang akan direkam.
7. Proses perekaman gerakan.
   Pada proses ini, gerakan pemain akan direkam oleh sistem. Pengguna menekan tombol mulai dan berhenti untuk melakukan perekaman dan mengehentikan proses rekaman. Proses ini menghasilkan keluaran berupa berkas teks informasi posisi kerangka pengguna dengan nama berkas sesuai dengan masukan dari pengguna.
8. Proses pembacaan berkas hasil rekaman gerakan.
   Pada proses ini, berkas teks yang dihasilakan oleh proses perekaman gerakan dibaca oleh sistem dan dipasangkan kembali ke model tiga dimensi. Proses ini terjadi pada modul pencocokan gerakan.

9. Proses pencocokan gerakan.
   Proses ini melakukan kalkulasi pencocokan gerakan pemain dengan animasi model 3 dimensi yang telah mendapatkan informasi posisi kerangka dari berkas teks.
10. Proses kalkulasi nilai.
    Proses kalkulasi nilai melakukan penilaian terhadap keakuratan gerakan pengguna menirukan gerakan animasi model tiga dimensi, sesuai dengan keluaran dari proses pencocokan gerakan.

4.1.4 Analisis Aktor

Pada aplikasi *basic taekwondo training system* hanya terdapat satu aktor yaitu pengguna. Cakupan yang dilakukan oleh pengguna tersebar dalam beberapa modul dalam aplikasi. Dalam aplikasi ini pengguna dapat melakukan gerakan yang akan ditangkap oleh sensor kinect, melihat animasi model 3 dimensi, dan pencitraan RGB yang ditampilkan dalam antarmuka aplikasi. Pengguna juga dapat melakukan perekaman gerakan dengan memberi nama pada gerakan serta melakukan perintah untuk memulai rekam dan berhenti rekam. Selanjutnya pemain dapat melihat hasil pencocokan gerakan dan hasil latihan.

4.1.5 Deskripsi Umum

Aplikasi yang dikerjakan dalam tugas akhir ini menggunakan *framework* XNA dan *software development kit* (SDK) dari Micorosoft Kinect. Aplikasi terdiri dari 3 modul utama yaitu, pendeteksi gerakan pengguna, perekaman gerakan pengguna, dan pencocokan gerakan pengguna.

Modul pertama yaitu pendeteksi gerakan pengguna. Pada modul ini gerakan pengguna akan ditangkap oleh sensor kinect kemudian diolah oleh aplikasi dan menghasilkan output koordinat-koordinat dari setiap titik sendi pengguna. Sensor kinect yang digunakan dalam tugas akhir ini adalah versi pertama yang hanya mampu menangkap 20 sendi pengguna. Gerakan-gerakan seperti berbalik, posisi menyamping, rukuk, jongkok, atau menyilangkan tangan tidak dapat dibaca secara sempurna. Hal ini memang dikarenakan keterbatasan yang dimiliki sensor. Keterbatasan lainnya adalah, jarak sensor terhadap pengguna dan lebar jangkauan yang dapat dideteksi.

4.1.5.1 Modul Pendeteksi Gerakan

Pada modul ini juga dilakukan pemanfaatan keluaran koordinat yang ditangkap oleh sensor kinect untuk ditransformasikan ke dalam susuan koordinat model tiga dimensi. Dengan cara tersebut gerakan dari pengguna dapat ditirukan oleh model tiga dimensi secara langsung dan terus menerus selama aplikasi berjalan.

Adapun proses yang dilakukan oleh modul pendeteksi gerakan adalah,

1. Masukan, masukan atau *input* dari modul ini adalah gerakan pengguna yang ditangkap oleh sensor kinect yang akan dibaca oleh sistem memanfaatkan SDK dari sensor kinect. Masukan berupa informasi koordinat dari kerangka pengguna pada ruang tiga dimensi.

2. Proses, proses yang dimaksudkan adalah memanfaatkan data koordinat dari kerangka pengguna kemudian ditransformasikan pada koordinat kerangka model tiga dimensi. Perhitungan ini dilakukan secara bersamaan sehingga menghasilkan animasi model tiga dimensi yang seolah menirukan gerakan pengguna.
3. Keluaran, keluaran yang dihasilkan dari modul ini adalah animasi model tiga dimensi yang mengikuti gerakan pengguna aplikasi serta tampilan *depth stream* dari pengguna untuk mengetahui gerakan pengguna yang sesungguhnya. Selain itu ditambahkan pula informasi posisi sudut sensor kinect dan jumlah *frame per second* (fps) aplikasi.

4.1.5.2 Modul Perekaman Gerakan

Modul perekaman gerakan digunakan untuk merekam gerakan pengguna yang kemudian akan disimpan. Keluaran utama dari modul ini adalah berkas teks yang akan digunakan pada modul pencocokan gerakan. Proses yang dilakukan dalam modul perekaman gerakan adalah,

1. Masukan, masukan dari modul ini selain dari gerakan pengguna, terdapat juga masukan dari keyboard dan mouse. Masukan dari keyboard dan mouse digunakan pengguna untuk memasukkan nama gerakan yang akan direkam.

2. Proses, proses yang terjadi pada modul ini adalah pengguna diminta untuk memasukkan nama gerakan pada tampilan masukan teks pada aplikasi. Kemudian pengguna melakukan perekaman dengan menekan tombol rekam pada tampilan antar muka aplikasi. Jika perekaman sudah dirasa cukup maka pengguna harus mengehentikan rekaman dengan menekan tombol berhenti merekam. Permulaan dan akhir perekaman gerakan dilakukan secara manual karena gerakan taekwondo tidak berpatokan pada pola hitungan tertentu seperti halnya pada senam.
   Sistem akan melakukan pecatatan ketika ada perubahan koordinat pada kerangka pengguna. Sistem akan menuliskan informasi tersebut dalam berkas *binary* sebanyak 30 kali dalam 1 detik. Pengambilan ini berdasarkan jumlah *frame per second* (fps) yang dihasilkan oleh sensor kinect.
3. Keluaran dari modul ini adalah berkas *binary* yang berisi informasi koordinat sendi pengguna tiap frame dan jumlah frame yang akan digunakan dalam modul pencocokan gerakan.

4.1.5.3 Modul Pencocokan Gerakan

Modul pencocokan gerakan mengambil peran sebagai pembanding antara gerakan yang dilakukan oleh pengguna dengan gerakan model tiga dimensi yang dihasilkan

oleh modul perekaman gerakan. Proses yang dilakukan dalam modul pencocokan gerakan adalah,

1. Masukan informasi sendi pengguna tiap frame dari hasil modul perekaman dan informasi sendi pengguna pada saat modul pencocokan gerakan dijalankan. Penulis memberikan istilah *recordedSkeleton* dan *actualSkeleton*.
2. Proses, proses yang dilakukan dalam modul pencocokan gerakan adalah membandingkan 2 data yaitu data dari *recordedSkeleton* dan *actualSkeleton* yang berisi informasi sendi tiap framenya. Pembandingan akan dilakukan ketika kedua data tersebut sama dan dilakukan dalam urutan frame yang sama. Hal ini memungkinkan dilakukan pembandingan kedua data secara langsung tiap framenya.
3. Keluaran, keluaran yang dimiliki oleh modul pencocokan gerakan ada 2 yaitu, hasil pencocokan tiap frame secara langsung dan hasil pencocokan gerakan secara keseluhuran.

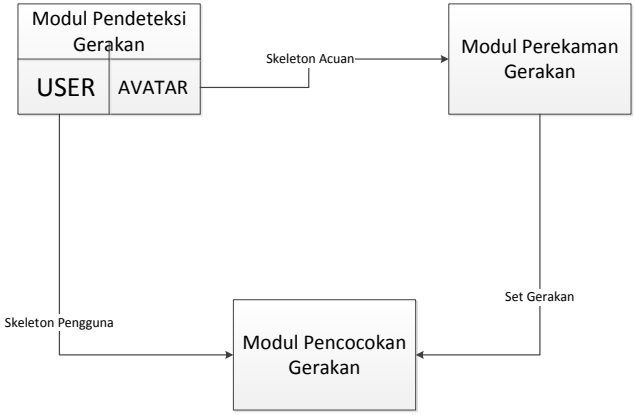
4.1.6 Arsitektur Perangkat Keras

Berikut merupakan aristektur yang digunakan dalam pengebangan aplikasi.

Gambar 4.1 Arsitektur Perangkat Keras

4.1.7 Arsitektur Sistem

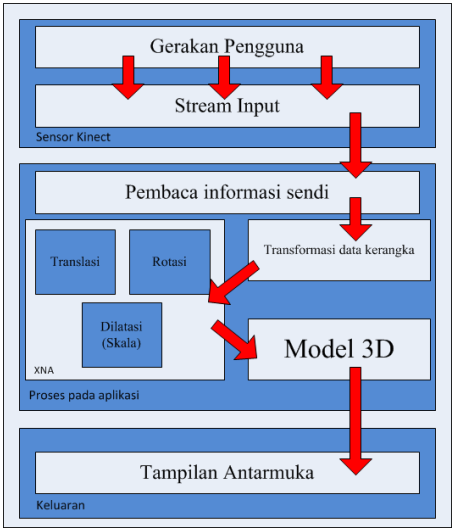
Aplikasi *basic taekwondo training system* dibangun berdasarkan tiga modul utama, yaitu modul pendeteksi gerakan, perekaman gerakan, dan pencocokan gerakan. Ketiga modul tersebut saling terhubung dan saling memanfaatkan keluaran yang dihasilkan. Hubungan antara ketiga modul tersebut dapat dilihat pada Gambar 4.2.



Gambar 4.2 Arsitektur Hubungan Antar Modul

Alur berjalannya aplikasi dimulai dari gerakan pemain ditangkap oleh sensor kinect dan diubah menjadi aliran data yang akan diolah oleh aplikasi

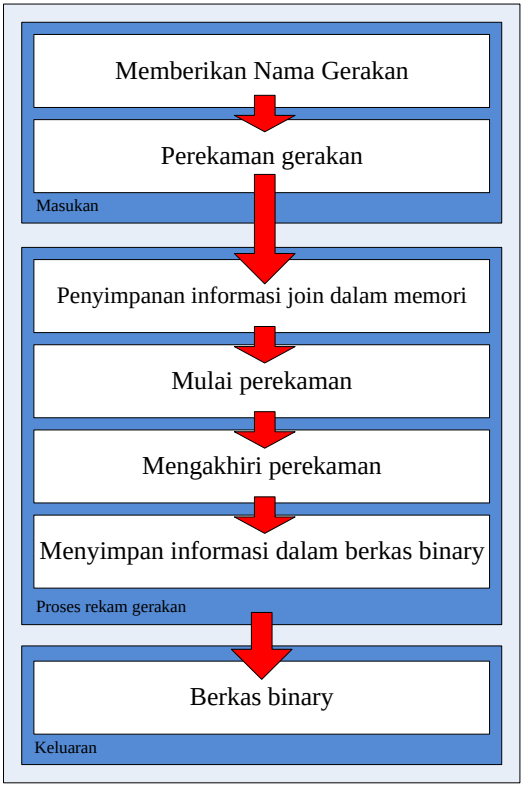
menggunakan Microsoft Kinect SDK. Aliran data tersebut kemudian oleh aplikasi diubah dan disimpan kedalam bentuk data informasi kerangka (*skeleton*) (Pramudita, 2012).



Gambar 4.3 Arsitektur Modul Pendeksi Gerakan

Data informasi kerangka yang didapat kemudian diolah kembali menjadi informasi kerangka yang dapat dipahami oleh *framework* XNA. Proses tersebut melakukan transformasi, penyekalaan, dan memberikan efek cermin dari informasi kerangka yang dikirim oleh sensor kinect. Selanjutnya informasi kerangka yang telah dirubah akan dipasangkan kedalam kerangka model tiga dimesi dan disesuaikan dengan informasi kerangka yang dimiliki oleh model. Proses ini berlangsung secara bersamaan dan terus terjadi selama aplikasi dijalankan. Pada saat ini juga akan ditampilkan tampilan *depth stream* yang menampilkan gerakan pengguna sebenarnya.

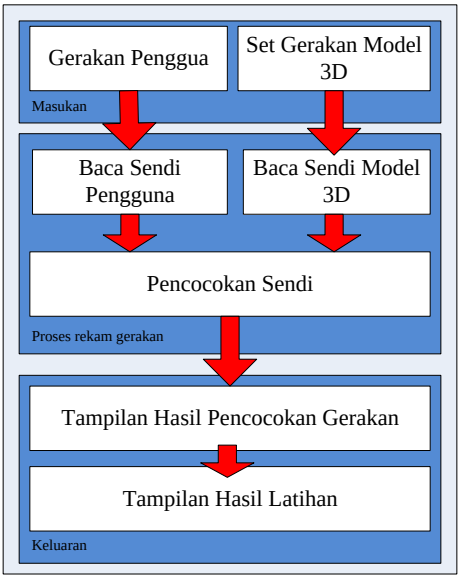
Tampilan *depth stream* dimaksudkan agar pengguna dapat memantau gerakan yang dilakukan.



Gambar 4.4 Arsistektur Modul Perekaman Gerakan

Selanjutnya terjadi animasi dari model tiga dimensi yang seolah menirukan gerakan pengguna aplikasi. Jika tahap ini sudah berjalan dengan baik, selanjutnya gerakan dasar taekwondo akan direkam dan diberikan nama. Gerakan akan direkam sebanyak 30 kali dalam satu detik. Perekaman ini disesuaikan dengan jumlah maksimal *frame* yang dapat dihasilkan oleh sensor

kinect. Alasan perekaman gerakan sebanyak 30 kali adalah untuk memberikan lebih banyak informasi dan pantokan penilaian karena hasil perekaman akan digunakan dalam modul pencocokan gerakan. Perekaman gerakan akan menghasilkan informasi posisi sendi kerangka pengguna dalam ruang tiga dimesi.

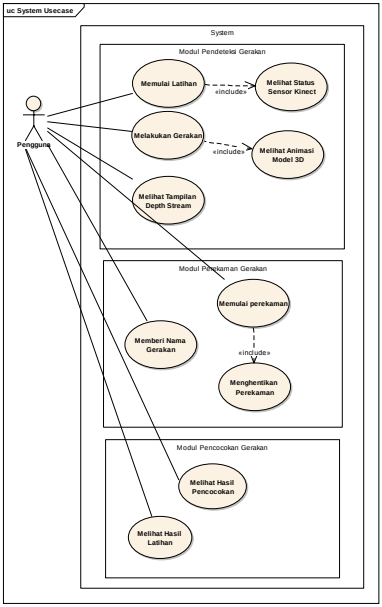


Gambar 4.5 Arsitetkur Modul Pencocokan Gerakan

Proses terakhir adalah melakukan pencocokan gerakan pengguna dengan gerakan animasi model tiga dimensi yang dihasilkan dari proses perekaman. Kemudian dilakukan penilai terhadap gerakan pemain sesuai dengan set gerakan yang telah direkam. Terakhir adalah memberikan nilai pada gerakan pemain agar pemain dapat memantau perkembangan latihan.

4.1.8 Skenario Penggunaan

Berdasarkan analisis aktor yang telah dijelaskan pada sub bab sebelumnya maka dibuatlah *use case* yang menjelaskan mengenai kebutuhan fungsionalitas dari aplikasi. Diagram *use case* dapat dilihat pada Gambar 4.6.



Gambar 4.6 Use Case Diagram Aplikasi

Dari diagram *use case* pada Gambar. Maka dapat diketahui terdapat sepuluh *use case* yang dibagi dalam 3 modul yang terdapat dalam aplikasi, yaitu modul pendeteksi gerakan, perekaman gerakan, dan pencocokan gerakan. Kesembilan *use case* tersebut terdapat proses dan fungsi yang digunakan dalam aplikasi. Terdapat juga aktor yang berperan dalam penggunaan *use case* yaitu pengguna aplikasi. Penjelasan tiap *use case* tiap modul dijelaskan pada Tabel 4.1. , Tabel 4.2, dan Tabel 4.3.

Tabel 4.1 Use Case Modul Pendeteksi Gerakan

No.	Kode *Use Case*	Nama *Use Case*	Keterangan
1.	UC-101	Memulai latihan	Pengguna mulai menjalankan aplikasi.
2.	UC-102	Melihat Status Sensor Kinect	Pengguna dapat melihat status yang dimiliki sensor kinect. Status yang digunakan dalam aplikasi yaitu, 1. Initializing 2. Connected 3. Disconnected 4. Not Powered 5. Not Ready 6. Error 7. Insufficient Bandwidth
3.	UC-103	Melakukan Gerakan	Pengguna melakukan gerakan kemudian gerakan tersebut akan terbaca oleh

			sensor kinect dan diolah dalam aplikasi.
4.	UC-104	Melihat Animasi Model 3D	Pada saat yang bersamaan hasil pengolahan informasi kerangka yang dikirmkan oleh sensor kinect dipasangkan pada struktur kerangka yang dimiliki oleh model tiga dimensi. Model tersebut akan bergerak (teranimasi) seolah menirukan gerkan pengguna. Animasi gerakan pengguna ini dapat dilihat secara langsung oleh pengguna.
5.	UC-105	Melihat Tampilan Depth Stream	Pada saat bersaman pengguna juga dapat melihat tampilan *depth stream* yang dihasilkan oleh sensor kinect. *Depth stream* yang ditampilkan menggambarkan keadaan sebenarnya dari pengguna.

Tabel 4.2 Use Case Modul Perekaman Gerakan

No.	Kode *Use Case*	Nama *Use Case*	Keterangan
1.	UC-201	Memberi Nama Gerakan	Termasuk dalam alur *use case* UC-201, pemain diminta untuk

			memasukkan nama gerakan.
2.	UC-202	Memulai Perekaman	Sebelum melakukan perekaman gerakan, pengguna diminta untuk menekan tombol rekam.
3.	UC-203	Menghentikan Perekaman	Untuk mengehentikan perekaman, pengguna diminta untuk menekan tombol berhenti.

Tabel 4.3 Use Case Pencocokan Gerakan

No.	Kode *Use Case*	Nama *Use Case*	Keterangan
1.	UC-301	Melihat Hasil Pencocokan	Pengguna dapat melihat hasil pencocokan gerakan, antara pengguna dengan gerakan model hasil rekaman.
2.	UC-302	Melihat Hasil Latihan	Pada akhir latihan, pengguna dapat melihat nilai akhir dari latihan untuk evaluasi latihan.

4.1.9 Deskripsi Use Case

Pada subbab ini akan lebih didetailkan kembali *use case* tiap modul pada aplikasi *basic taekwondo*

training system. Dalam deskripsi use case akan dijelaskan pula alur dari proses tiap use case dan disertakan diagram aktifitas tiap *use* case-nya.

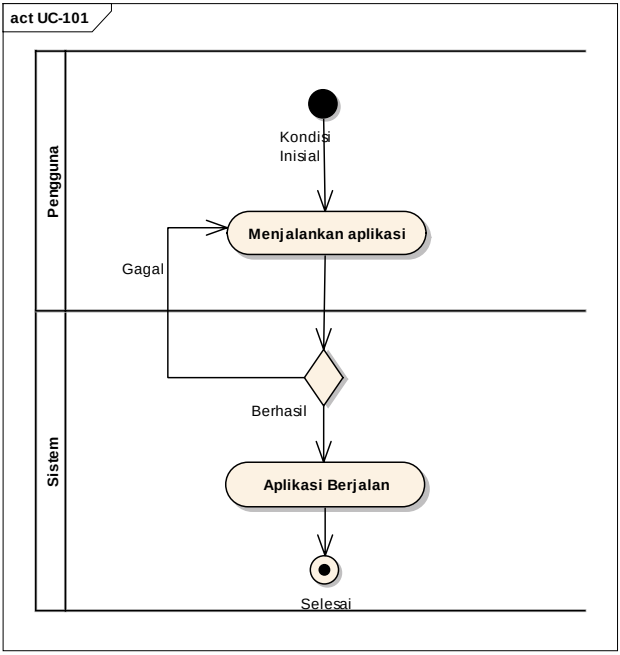
4.1.9.1 Deskripsi Use Case Modul Pendeteksi Gerakan

Berikut merupakan deskripsi *use case* yang terdapat dalam modul pendeteksi gerakan.

1. UC-101 : Memulai Latihan

Nama Use Case	Memulai Latihan
Kode Use Case	UC-101
Aktor	Pengguna
Deskripsi	Pengguna akan memulai latihan dengan menjalankan aplikasi.
Kondisi Awal	Pengguna belum menjalankan aplikasi.
Alur Normal	1. Pengguna menjalankan aplikasi. A1. Aplikasi *basic taekwondo training system* tidak berjalan. 2. Aplikasi berjalan. 3. Sistem menampilkan tampilan antarmuka utama. 4. Selesai.
Alur Alternatif	A1. Aplikasi *basic taekwondo training system* tidak berjalan. **1.** Pengguna menjalankan

	kembali aplikasi.
Kondisi Akhir	Aplikasi *basic taekwondo training system* berjalan.

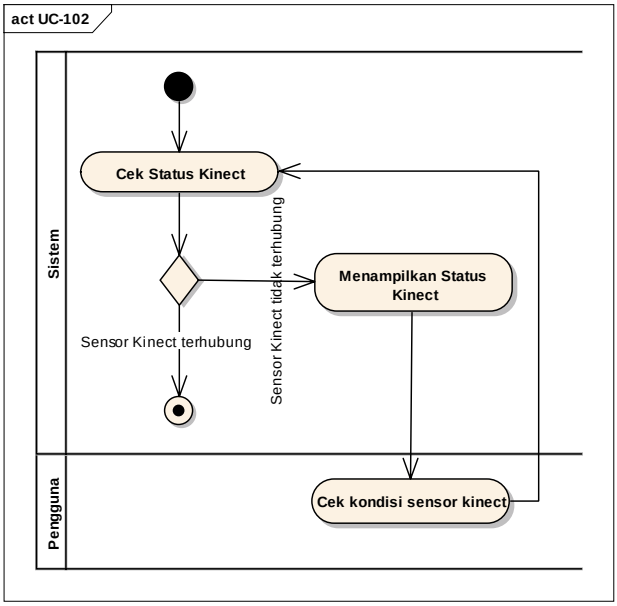


Gambar 4.7 Diagram Aktifitas UC-101

2. UC-102 : Cek Status Sensor Kinect

Nama Use Case	Melihat Status Sensor Kinect
Kode Use Case	UC-102
Aktor	Sistem, Pengguna

Deskripsi	Sistem melakukan pengecekan terhadap status sensor kinect.
Kondisi Awal	Aplikasi telah berjalan.
Alur Normal	1. Sistem melakukan pengecekan status kinect. 2. Status kinect terhubung. A1. Sensor kinect tidak terhubung. 3. Selesai
Alur Alternatif	A1. Sensor kinect tidak terhubung. 1. Sistem menampilkan status kinect. a. *Initializing* b. *Disconnected* c. *Not Powered* d. *Not Ready* e. *Error* f. *Insufficient Bandwidth* 2. Pengguna melakukan pengecekan koneksi sensor kinect ke sistem aplikasi. 3. Kembali ke alur normal no 2.
Kondisi Akhir	Sensor terhubung dengan sistem aplikasi.

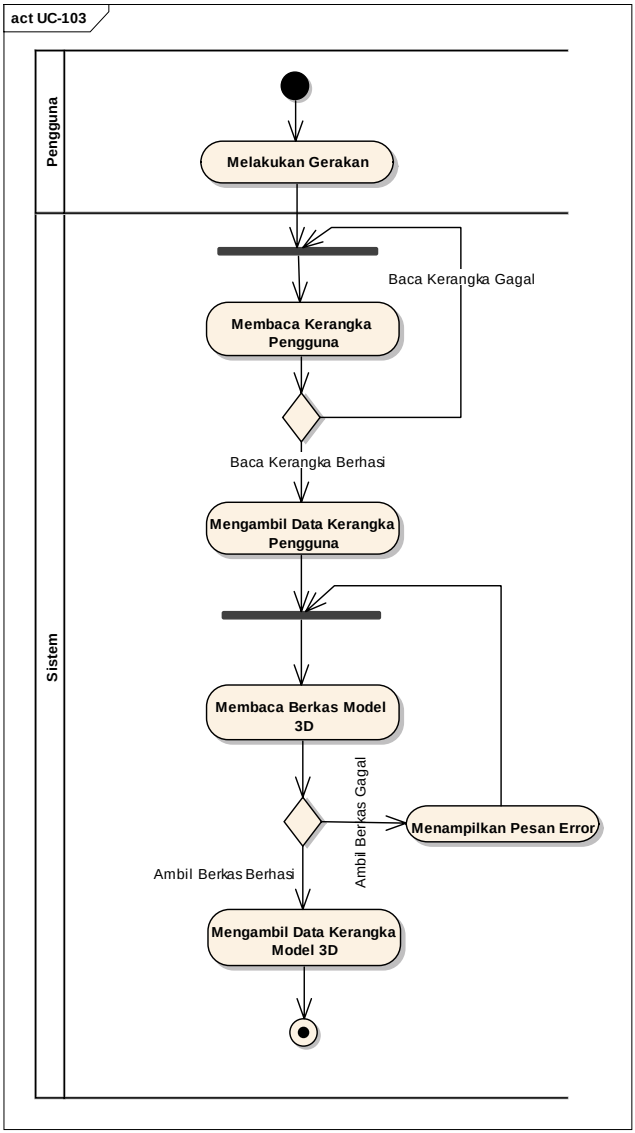


Gambar 4.8 Diagram Aktifitas UC-102

3. UC-103 : Melakukan Gerakan

Nama Use Case	Melakukan Gerakan
Kode Use Case	UC-103
Aktor	Pengguna
Deskripsi	Pengguna dalam jangkauan sensor kinect
Kondisi Awal	Pengguna berada pada jangkauan sensor kinect.
Alur Normal	1. Pengguna melakukan gerakan pada jangkauan sensor kinect.

	2. Sistem membaca struktur kerangka dari pengguna. A1. Sistem tidak dapat membaca struktur kerangka pengguna. 3. Sistem mengambil data struktur kerangaka pengguna. 4. Sistem membaca berkas model tiga dimensi. A2. Sistem tidak dapat membaca berkas model tiga dimensi. 5. Sistem mengambil data struktur kerangka model tiga dimensi. 6. Selesai.
Alur Alternatif	A1. Sistem tidak dapat membaca struktur kerangka pengguna. 1. Pengguna diluar jangkauan sensor kinect. 2. Kembali ke alur normal no 1. A2. Sistem tidak dapat membaca berkas model tiga dimensi. 1. Sistem menampilkan pesan bahwa berkas model 3 dimensi tidak dapat dibaca. 2. Kembali ke alur normal no 2.
Kondisi Akhir	Struktur kerangka pengguna dan model tiga dimensi didapatkan.

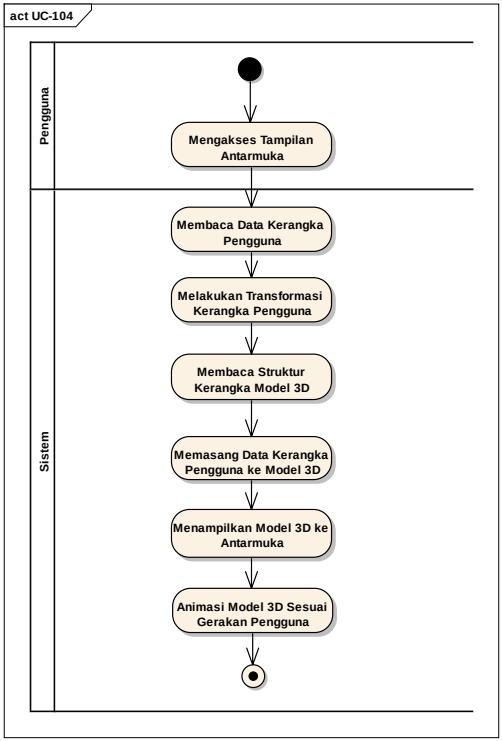


Gambar 4.9 Diagram Aktifitas UC-103

4. UC-104 : Melihat Animasi Model 3D

Nama Use Case	Melihat Animasi Model 3D
Kode Use Case	UC-104
Aktor	Pengguna
Deskripsi	Pengguna dapat melihat animasi gerakan model tiga dimensi sesuai dengan gerakan yang dilakukan.
Kondisi Awal	1. Pengguna berada pada jangkauan sensor kinect. 2. Data struktur kerangka pengguna dan model tiga dimensi telah didapatkan.
Alur Normal	1. Pengguna mengakses tampilan antarmuka. 2. Sistem membaca data kerangka pengguna sesuai dengan gerakan yang dilakukan. 3. Sistem melakukan perhitungan transformasi struktur kerangka pengguna. 4. Sistem membaca kerangka model tiga dimensi. 5. Sistem memasangkan data kerangaka pengguna yang telah ditransformasi ke model tiga dimensi. 6. Sistem menampilkan model tiga dimensi ke tampilan antarmuka.

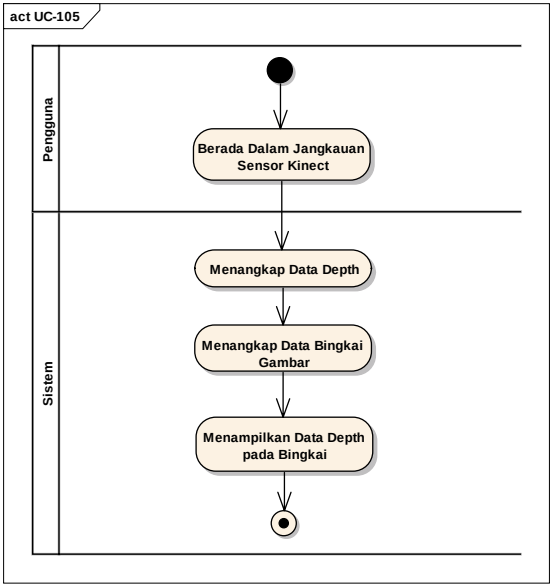
	7. Model tiga dimensi melakukan animasi sesuai dengan gerakan pengguna. 8. Selesai.
Alur Alternatif	-
Kondisi Akhir	Pengguna melihat animasi model tiga dimensi pada tampilan antarmuka.



Gambar 4.10 Diagram Aktifitas UC-104

5. UC-105 : Melihat Tampilan Depth Stream

Nama Use Case	Melihat Tampilan Depth Stream
Kode Use Case	UC-105
Aktor	Pengguna
Deskripsi	Pengguna dapat melihat tampilan representasi kondisi sebenarnya berformat *depth* dari obyek yang ditangkap oleh sensor kinect.
Kondisi Awal	Pengguna dalam jangkauan sensor kinect.
Alur Normal	1. Tubuh pengguna berada pada jangkauan sensor kinect. 2. Sistem membaca data yang ditangkap oleh sensor infra merah dari sensor kinect. 3. Sistem menangkap data bingkai gambar. 4. Sistem menampilkan data depth yang ditangkap ke bingkai gambar. 5. Selesai.
Alur Alternatif	-
Kondisi Akhir	Pengguna melihat tampilan *depth stream* pada tampilan antarmuka utama.



Gambar 4.11 Diagram Aktifitas UC-105

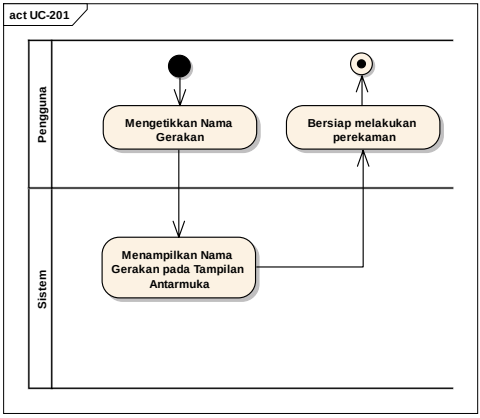
4.1.9.2 Deskripsi Use Case Modul Perekaman Gerakan

Berikut merupakan deskripsi *use case* yang terdapat dalam modul perekaman gerakan.

1. UC-201 : Memberi Nama Gerakan

Nama Use Case	Memberi Nama Gerakan
Kode Use Case	UC-201
Aktor	Pengguna

Deskripsi	Pengguna memberikan nama gerakan yang akan direkam.
Kondisi Awal	Pengguna dalam kondisi siap untuk melakukan perekaman.
Alur Normal	1. Pengguna mengetikkan nama gerakan dalam kolom teks. 2. Sistem menampilkan nama gerakan pada kolom teks 3. Pengguna bersiap menekan tombol mulai rekam. 4. Selesai
Alur Alternatif	-
Kondisi Akhir	Berkas dengan nama gerekan yang diketikkan pengguna siap dibuat oleh sistem.

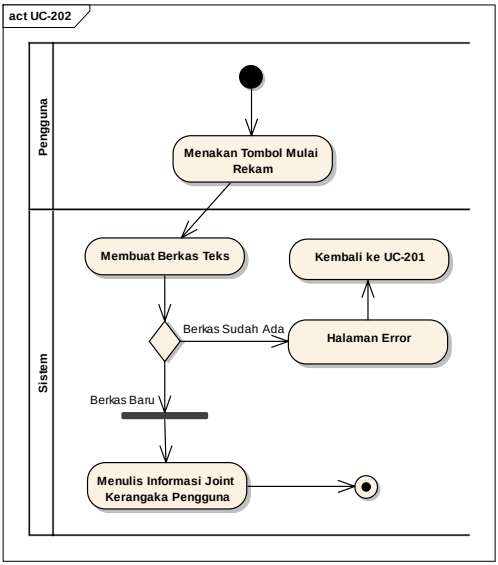


Gambar 4.12 Diagram Aktifitas UC-201

2. UC-202 : Memulai Perekaman

Nama Use Case	Memulai Perekaman
Kode Use Case	UC-202
Aktor	Pengguna
Deskripsi	Pengguna melakukan perekaman dengan menekan tombol mulai rekam.
Kondisi Awal	Pengguna dala kondisi siap melakukan perekaman. Nama gerakan telah diketikkan.
Alur Normal	1. Pengguna dalam kondisi siap untuk melakukan perekaman. 2. Pengguna telah memberikan nama gerakan. 3. Pengguna menekan tombol mulai perekaman. 4. Sistem membuat berkas teks dengan nama sesuai dengan masukan dari pengguna. A1. Sistem menemukan kesamaan nama berkas teks. 5. Sistem menuliskan informasi koordinat 20 joint kerangka pengguna dalam berkas teks yang telah dibuat sebanyak 30 kali dalam 1 detik. 6. Selesai.
Alur Alternatif	A1. Sistem menemukan kesamaan nama berkas teks.

	1. Sistem menuju halaman error. 2. Kembali ke alur normal UC-201 nomor 1.
Kondisi Akhir	Sistem menuliskan informasi joint kerangka pengguna dalam berkas teks yang dibuat.

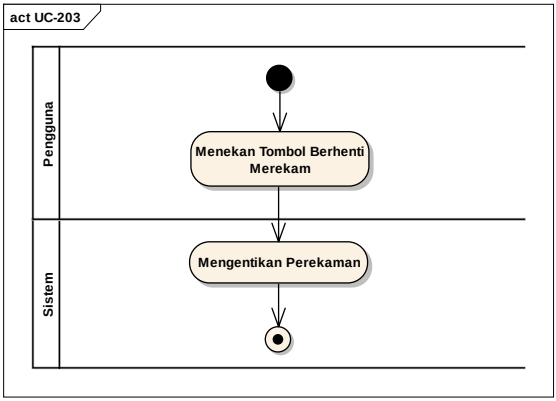


Gambar 4.13 Diagram Aktifitas UC-202

3. UC-203 : Menghentikan Perekaman

Nama Use Case	Menghentikan Perekaman
Kode Use Case	UC-203

Aktor	Pengguna
Deskripsi	Pengguna menghentikan perekaman gerakan.
Kondisi Awal	Pengguna dalam keadaan bergerkak dan sistem melakukan perekaman.
Alur Normal	1. Pengguna menekan tombol berhenti merekam. 2. Sistem menghentikan aktifitas perekaman. 3. Selesai.
Alur Alternatif	-
Kondisi Akhir	Sistem kembali pada kondisi UC-104.



Gambar 4.14 Diagram Aktifitas UC-203

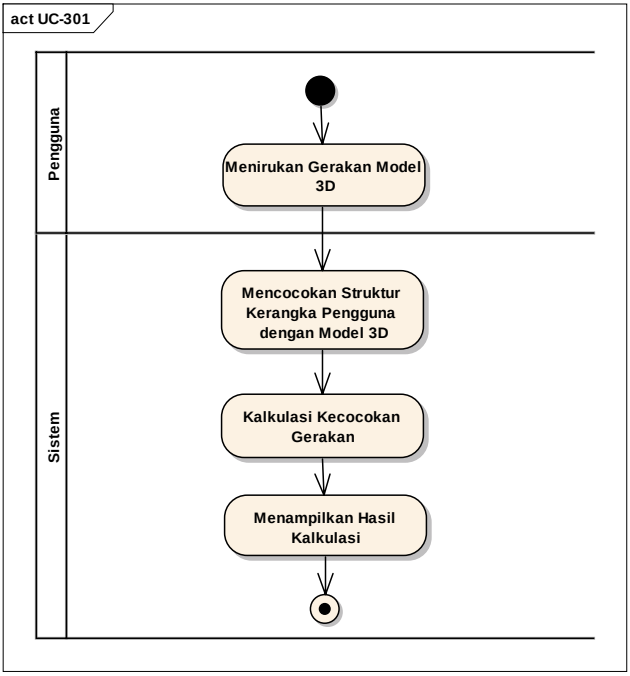
4.1.9.3 Deskripsi Use Case Modul Pencocokan Gerakan

Berikut merupakan deskripsi *use case* yang terdapat dalam modul pencocokan gerakan.

1. UC-301 : Melihat hasil pencocokan gerakan.

Nama Use Case	Melihat Hasil Pencocokan Gerakan
Kode Use Case	UC-301
Aktor	Pengguna
Deskripsi	Pengguna dapat melihat hasil pencocokan gerakan latihan dari gerakan yang dilakukan dengan gerakan model hasil perekaman.
Kondisi Awal	Pengguna menirukan gerakan model tiga dimenasi.
Alur Normal	1. Pengguna menirukan gerakan dari model tiga dimensi. 2. Sistem melakukan pencocokan struktur kerangka pengguna dengan struktur model tiga dimensi. 3. Sistem melakukan kalkulasi kecocokan gerakan pengguna dengan model. 4. Sistem menampilkan hasil pencocokan. 5. Pengguna dapat melihat hasil pencocokan secara

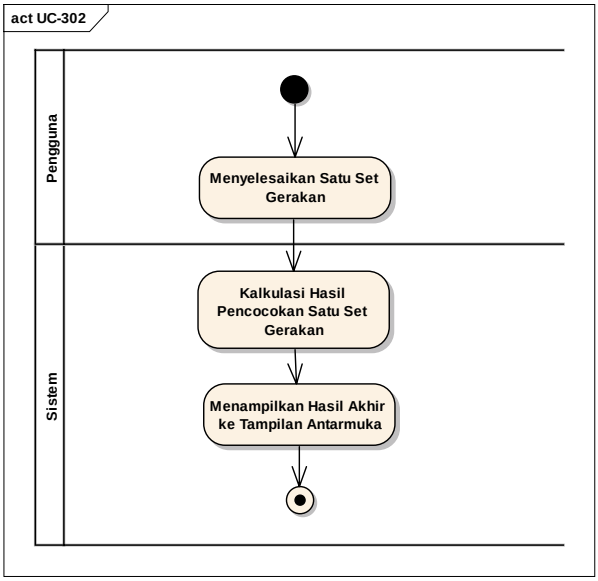
	langsung. 6. Selesai.
Alur Alternatif	-
Kondisi Akhir	Pengguna dapat melihat hasil pencocokan gerakan pada saat latihan gerakan secara langsung.



Gambar 4.15 Diagram Aktifitas UC-301

2. UC-302 : Melihat Hasil Latihan

Nama Use Case	Melihat Hasil Latihan.
Kode Use Case	UC-302
Aktor	Pengguna
Deskripsi	Pengguna melihat hasil latiha.
Kondisi Awal	Pengguna menyelesaikan satu set gerakan.
Alur Normal	1. Pengguna menyelesaikan satu set gerakan. 2. Sistem melakukan kalkulasi hasil akhir dari informasi pencocokan gerakan yang telah dilakukan. 3. Sistem menampilkan hasil akhir kalkulasi ke tampilan antarmuka. 4. Selesai.
Alur Alternatif	-
Kondisi Akhir	Pengguna melihat nalai hasil latihan.



Gambar 4.16 Diagram Aktifitas UC-302

4.2 Tahap Perancangan

Pada subbab ini akan dijelaskan mengenai rancangan dalam pengembangan apliaksi *basic taekwondo training system*. Hasil dari tahap analisis akan digunakan sebagai acuan dalam perancangan aplikasi.

4.2.1 Perancangan Umum Aplikasi

Dalam subbab ini akan dijelaskan rancangan umum dari aplikasi *basic taekwondo training system*. Berdasarkan analisis *use case* dilakukan perancangan aplikasi menggunakan *class diagram*. *Class diagram* aplikasi *basic taekwondo training system* dapat dilihat pada Lampiran A.

Secara umum *class diagram* aplikasi dibagi menjadi 2 bagian, bagian utama aplikasi dan bagian filter. Bagian utama berisi kelas-kelas yang merepresentasikan hasil analisis. Sedangkan bagian filter digunakan untuk membantu kelas utama khususnya dalam meminimalisir keterbasan sensor kinect.

Kelas-kelas utama terdiri dari,

1. KinectCheck
2. KinectHelper
3. Game1
4. Object2D
5. DepthStreamRenderer
6. SkeletonStreamRenderer
7. SkinningData
8. AvaratAnimator

Sedangkan untuk kelas-kelas filter teridir dari,

1. BoneOrientationConstraint
2. BoneOrientationDoubleExponentianFilter
3. SkeletonJointFilterClippedLegs
4. SkeletonJointSensorOffsetCorrection
5. SkeletonJointsPositionDoubleExponentialFilt -er
6. Timer
7. TimedLerp

4.2.2 Perancangan Proses Pendeteksi Gerakan

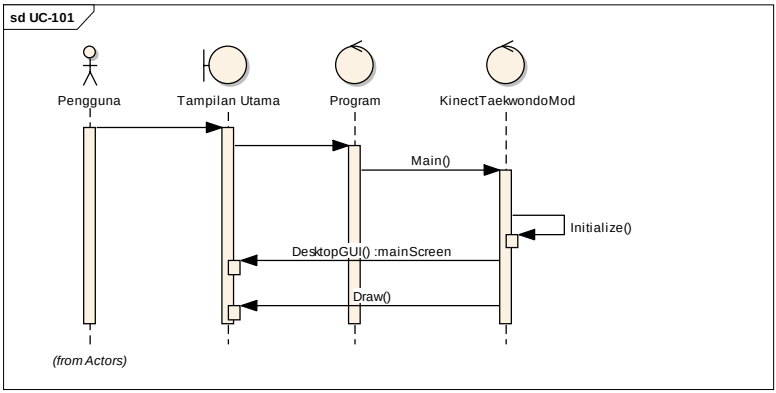
Sesuai dengan analisis *use case* dan *class diagram* yang telah dilakukan dalam tahapan analisis. Secara umum proses yang terjadi dalam modul pendeteksi gerakan dibagi menjadi 5 yaitu,

1. Memulai Latihan.

2. Melihat Status Sensor.
3. Melakukan Gerakan.
4. Melihat Animasi Model 3D.
5. Melihat Tampilan ColorStream.

4.2.2.1 Proses Memulai Latihan

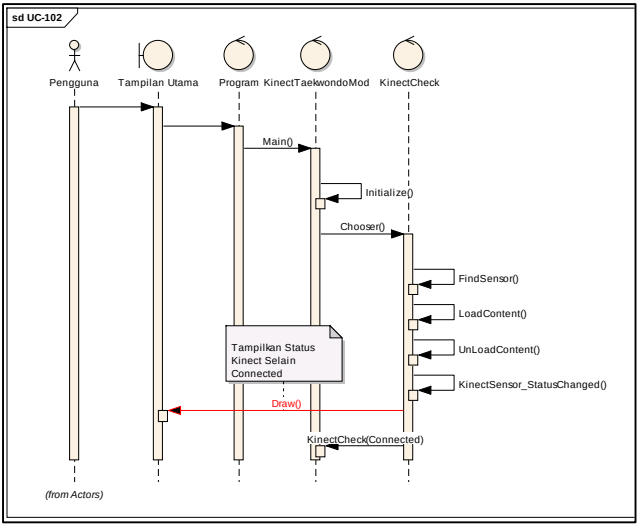
Proses memulai latihan ini dilakukan oleh pengguna aplikasi. Proses ini dimulai dari pengguna menjalankan aplikasi hingga sistem aplikasi berjalan. Proses memulai latihan dapat dilihat pada diagram alur pada Gambar 4.17.



Gambar 4.17 Sequence Diagram Memulai Latihan

4.2.2.2 Proses Melihat Status Sensor

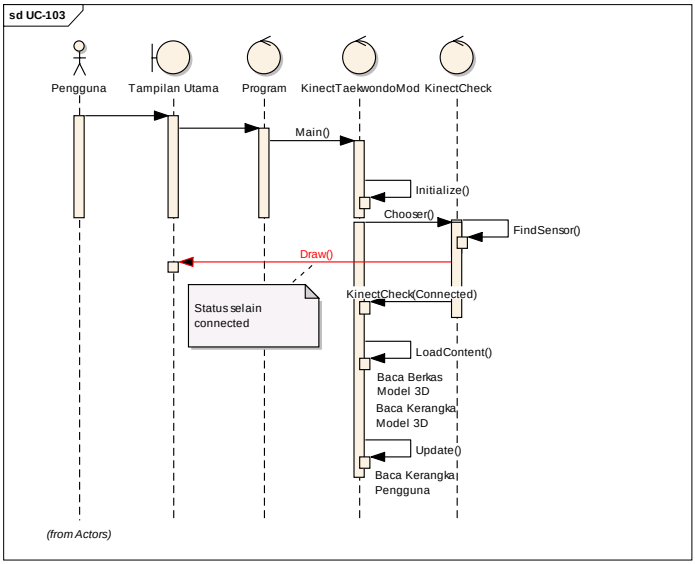
Proses melihat status sensor berjalan setelah proses mulai latihan. Proses utama dari melihat sensor adalah inisiasi sensor. Setelah proses inisasi terdapat proses pengecekan status sensor, apakah sensor terhubung atau tidak. Berikut *sequence diagram* dari proses melihat status sensor.



Gambar 4.18 Sequence Diagram Melihat Status Sensor

4.2.2.3 Proses Melakukan Gerakan

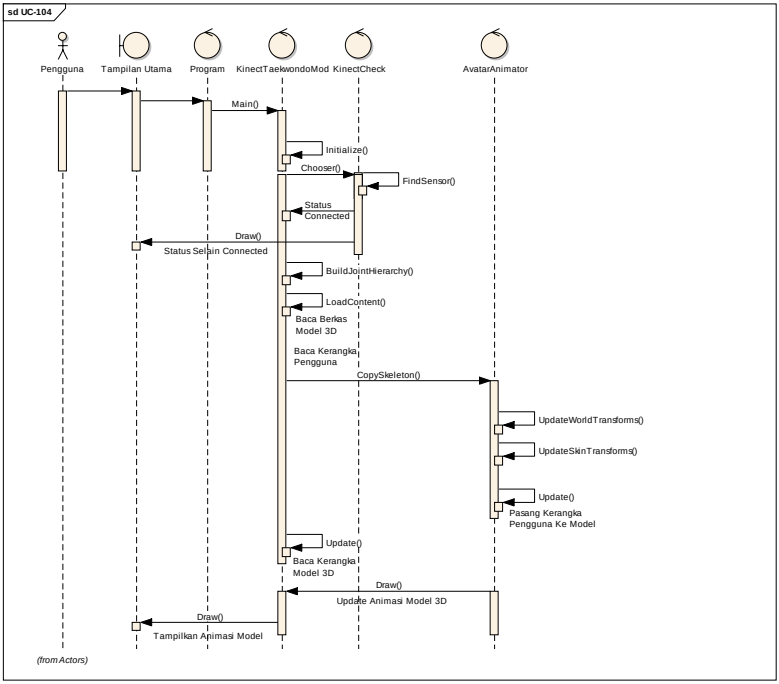
Pada proses melakukan gerakan, pengguna melakukan gerakan dalam jangkauan sensor kinect. Kemudian sistem akan membaca *skeleton* pengguna dan membaca berkas model tiga dimensi. *Sequence diagram* dari porses melakukan gerakan dapat dilihat pada Gambar 4.19.



Gambar 4.19 Sequence Diagram Melakukan Gerakan

4.2.2.4 Proses Melihat Animasi Model Tiga Dimensi

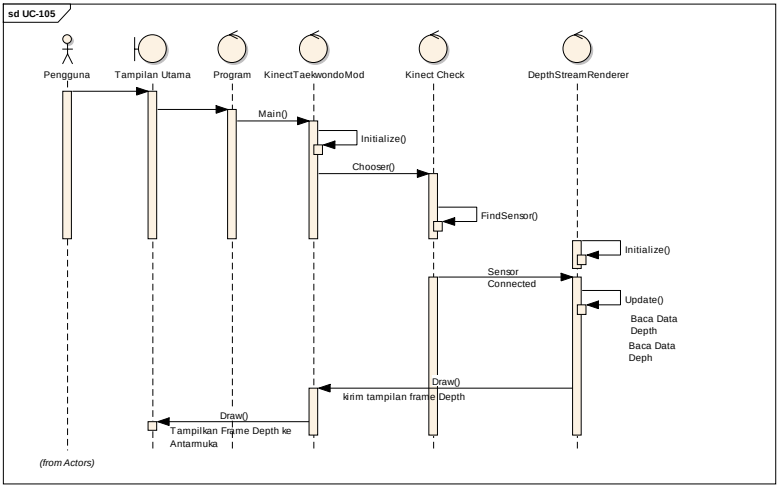
Proses ini melakukan transfrormasi *skeleton* dari pengguna untuk dipasangkan pada model tiga dimensi. Pada tahap akhir dari proses ini, model tiga dimensi akan melakukan animasi sesuai dengan gerakan pengguna dan ditampilkan pada antarmuka utama. *Sequence diagram* dari proses ini dapat lihat pada Gambar 4.20.



Gambar 4.20 Sequence Diagram Melihat Animasi Model 3D

4.2.2.5 Proses Melihat Tampilan Depth Stream

Pada proses ini sistem melakukan proses menampilkan tampilan *depth stream* keadaan sebenarnya dari pengguna. Sequence diagram dari proses ini dapat dilihat pada Gambar 4.21.



Gambar 4.21 Sequence Diagram Melihat Tampilan Depth Stream

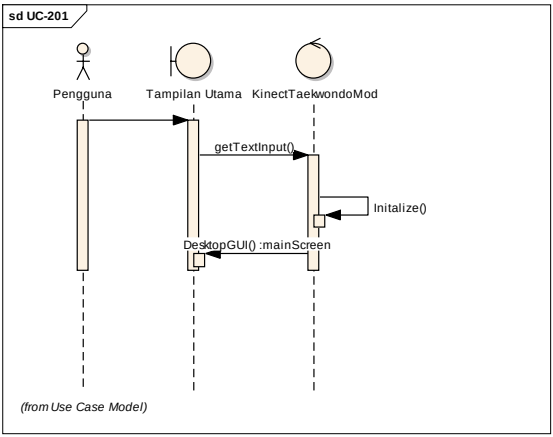
4.2.3 Perancangan Proses Perekaman Gerakan

Proses perekaman gerakan merupakan proses lanjutan dari proses pendeteksi gerak. Pada proses ini terdapat subproses yaitu,

1. Proses memberi nama gerakan
2. Proses memulai perekaman
3. Proses menghentikan perekaman

4.2.3.1 Proses Memberi Nama Gerakan

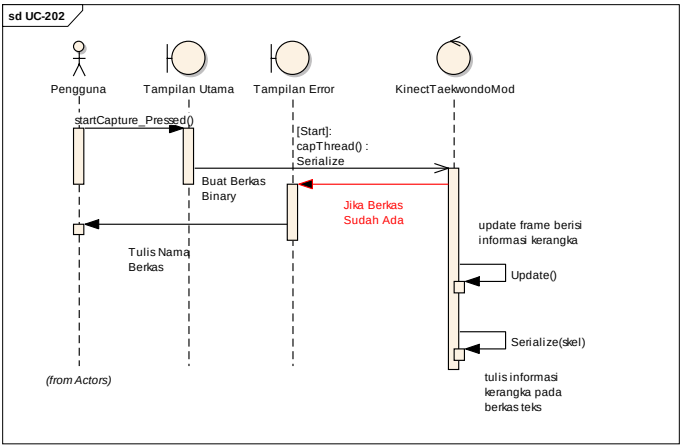
Pada proses ini, pengguna diminta untuk memasukkan nama gerakan yang akan direkam. Sequence diagram dari proses ini dapat dilihat pada Gambar 4.22.



Gambar 4.22 Sequence Diagram Memberi Nama Gerakan

4.2.3.2 Proses Memulai Perekaman

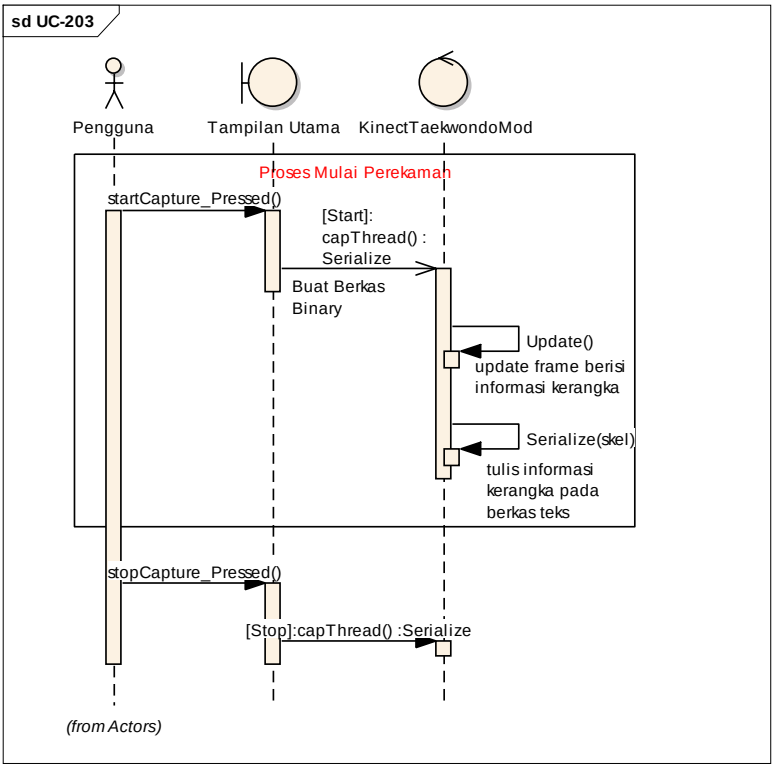
Subproses selanjutnya adalah memulai perekaman. Pada proses ini, pengguna akan menekan tombol rekam kemudian perekaman gerakan pengguna akan berjalan hingga proses selanjutnya. *Sequence diagram* dari proses memulai perekaman dapat dilihat pada Gambar 4.23.



Gambar 4.23 Sequence Diagram Memulai Perekaman

4.2.3.3 Proses Menghentikan Perekaman

Proses mengehentikan perekaman merupakan lanjutan dari proses memulai perekaman. Proses ini merupakan akhir dari proses perekaman. Proses mengehentikan perekaman bertugas untuk menghentikan perekaman gerakan pada proses sebelumnya kemudian menuliskan hasil perekaman pada berkas teks. Sequence diagram dari proses menghentikan perekaman dapat dilihat pada Gambar 4.24.



Gambar 4.24 Sequence Diagram Mengehentikan Perekaman

4.2.4 Perancangan Proses Pencocokan Gerakan

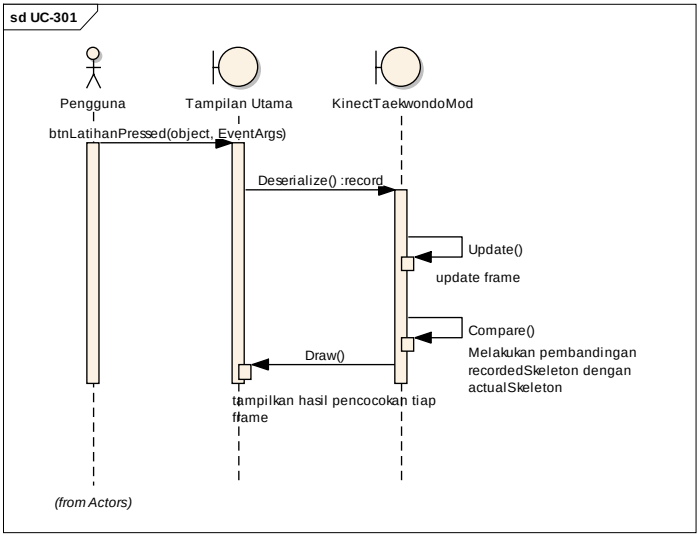
Pada proses ini akan dilakukan pencocokan gerakan antar gerakan hasil rekaman (recordedSkeleton) dan gerakan pengguna yang ditangkap secara langsung oleh sensor (actualSkeleton). Proses pencocokan gerakan dibagi menjadi 2 subproses yang saling berkelanjutan, yaitu,

1. Menampilkan hasil pencocokan tiap frame.

2. Menampilkan hasil akhir latihan.

4.2.4.1 Proeses Melihat Hasil Pencocokan Tiap Frame

Proses ini diawali dengan pengguna memulai latihan kemudian fungsi pencocokan akan dijalankan ketika kedua data kerangka, *recordedSkeleton* dan *actualSkeleton* terbaca oleh sistem. Rancangan proses melihat hasil pencocokan gerakan dapat dilihat pada Gambar 4.25.

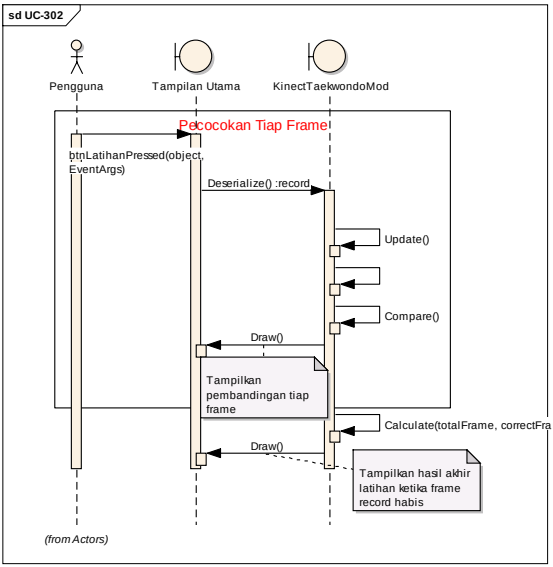


Gambar 4.25 Squence Diagram Proses Melihat Pencocokan Gerakan

4.2.4.2 Proses Melihat Hasil Latihan

Proses melihat hasil latihan adalah lanjutan dari sub proses melihat hasil pencocokan gerakan. Proses ini akan menampilkan hasil

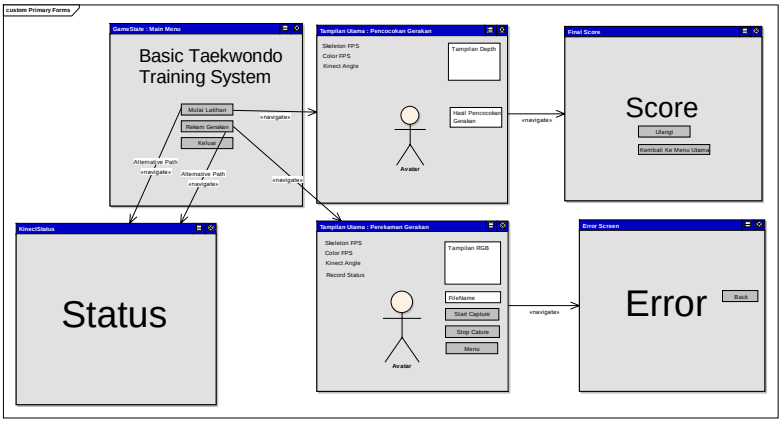
akhir dari latihan, dimana akhir latihan adalah akhir dari frame yang didapat dari hasil perekaman gerakan. Rancangan proses melihat hasil latihan dapat dilihat pada Gambar 4.26.



Gambar 4.26 Sequence Diagram Proses Melihat Hasil Latihan

4.2.5 Rancangan Tampilan Antarmuka

Pada subbab ini akan dijelaskan rancangan tampilan antarmuka aplikasi *basic taekwondo training system*. Tampilan keseluruhan dari rancangan tampilan antarmuka aplikasi dapat dilihat pada Gambar 4.27.



Gambar 4.27 Rancangan Antarmuka Aplikasi

Dari gambar dapat diketahui bahwa aplikasi *basic taekwondo training system* memiliki 4 tampilan utama yang mewakili *state-state* aplikasi, yaitu,

1. State main menu.
2. State kinect status.
3. State pencocokan gerakan.
4. State hasil akhir latihan.
5. State perekaman gerakan.
6. State error.

Halaman ini sengaja dikosongkan.

BAB V
IMPLEMENTASI DAN UJI COBA SISTEM

Pada bab ini, penulis membahas mengenai proses yang dilakukan selama tahap implementasi dan uji coba sistem. Di bawah ini merupakan gambar alur pelaksanaan implementasi yang akan dilakukan.

5.1 Lingkungan Pengembangan Aplikasi

Pengembangan aplikasi *basic taekwondo training system* ini dilakukan pada komputer *notebook* dengan spesifikasi tertera pada Tabel 5.1.

Tabel 5.1 Spesifikasi Komputer Notebook

Spesifikasi	
Prosesor	Intel Core i7 Q740 @1.73 Ghz
Memori	8 GB RAM
Kartu Grafis	NVIDIA GeFroce GT425M 2GB
Sistem Operasi	Windows 7 Home Premium 64-bit

Dalam implementasi dibutuhukan juga beberapa teknologi baik perangkat lunak maupun perangkat keras. Teknologi yang digunakan dalam implementasi aplikasi *basic taekwondo training system* terdapat pada Tabel 5.2.

Tabel 5.2 Teknologi Pengembangan Aplikasi

Teknologi	Versi
Sensor Gerak	Microsoft Kinect for XBOX v1 + USB Adapter
Integrated Development Environment (IDE)	Microsoft Visual Studio 2010 Professional (Student License)
Application Programming Interface (API)	Microsoft Kinect SDK v1.8
Framework Utama	XNA Game Studio 4.0

Framework Tambahan	Nuclex Framework for XNA
Pengolah Model 3D	Autodesk 3ds Max 2011 (Student License)

5.2 Implementasi Perangkat Sensor Microsoft Kinect for XBOX

Penggunaan sensor Microsoft Kinect for XBOX 360 pada komputer sedikit berbeda dengan sensor Microsoft Kinect for Windows. Perbedaan mendasar kedua sensor tersebut dalam sudut pandang perangkat keras yaitu bagian *port*. *Port* yang dimiliki oleh sensor untuk XBOX 360 serupa dengan USB *port* namun dengan bentuk trapesium. Sedangkan sensor untuk Windows, sensor memiliki *port* USB.

Gambar 5.1 Port Microsoft Kinect for XBOX 360

Penggunaan sensor kinect untuk XBOX 360 pada perangkat komputer memerlukan tambahan adapter yang berfungsi

sebagai sumber daya listrik pada sensor dan mengubah port XBOX 360 menjadi *port* USB.

Gambar 5.2 USB Adapter Kinect for XBOX 360

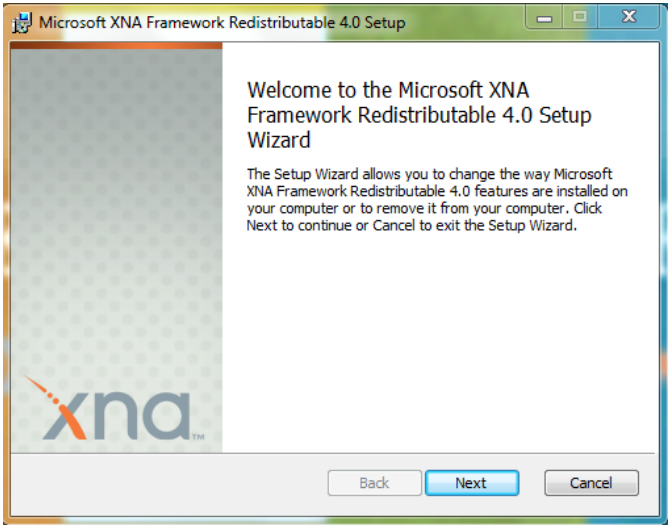
Jika sensor kinect untuk XBOX 360 sudah dilengkapi dengan tambahan adapter, selanjutnya *port* usb siap dihubungkan dengan perangkat komputer dan sensor siap digunakan.

5.3 Implementasi Teknologi Perangkat Lunak

Proses implementasi teknologi perangkat lunak dimaksudkan pada proses persiapan lingkungan teknologi perangkat lunak yang digunakan dalam implementasi pembuatan aplikasi. Proses ini terdiri dari instalasi framework XNA, Nuclex, dan instalasi Microsoft Kinect SDK v1.8.

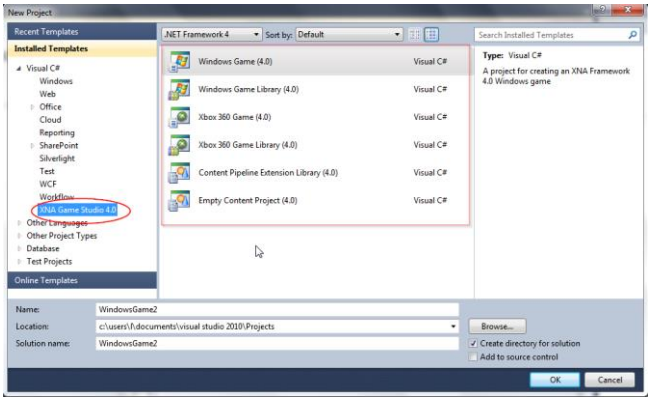
5.3.1 Instalasi Framework XNA

Proses implementasi teknologi perangkat lunak yang pertama adalah instalasi framework XNA. Berkas framework XNA dapat diunduh di website Microsoft. Selanjutnya adalah proses instalasi framework XNA.



Gambar 5.3 Proses Instalasi *Framework* XNA

Jika framework XNA sudah terinstal maka akan muncul pilihan development XNA pada IDE Visual Studio.

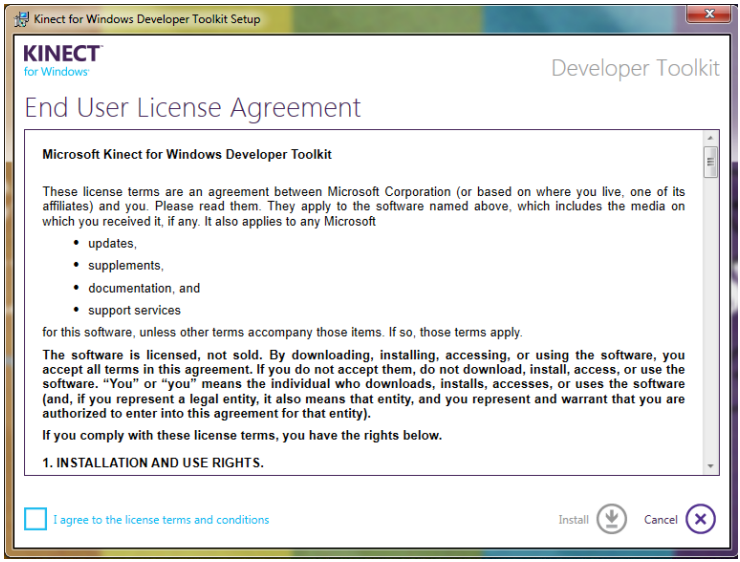


Gambar 5.4 Halaman Proyek XNA Baru pada Visual Studio 2010

Setelah framework XNA dapat terintegrasi dengan IDE Visual Studio, maka framework XNA siap digunakan.

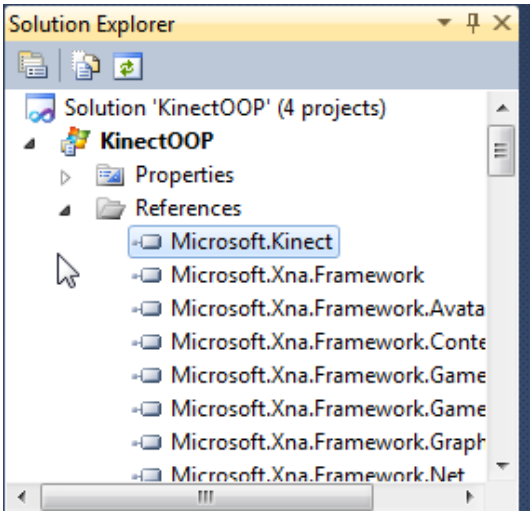
5.3.2 Instalasi Microsoft Kinect SDK

Proses implementasi selanjutnya adalah pemasangan *application programming interface* (API) dari sensor kinect yaitu dengan menggunakan Microsoft Kinect SDK. Untuk mendapatkan berkas Microsoft Kinect SDK dapat diunduh melalui website Microsoft.



Gambar 5.5 Proses Installasi Kinect for Windows SDK

Selanjutnya adalah memasukkan library SDK pada proyek pengerjaan aplikasi sebagai refrensi project. Libary kinect SDK dapat diambil dari direktori \Microsoft SDKs\Kinect\v1.8\Assemblies\Microsoft.Kinect.dll.



Gambar 5.6 Kumpulan Library Pada Solution Explorer Pengembangan Aplikasi

Jika telah terpasang, proses implementasi selanjutnya adalah jika membutuhkan untuk menggunakan library Microsft.Kinect.dll maka library tersebut harus dipanggil dalam kode project.

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Kinect;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;
using System.IO;
```

Potongan Kode 5.1 Inisiasi Library

5.3.3 Instalasi Nuclex Framework

Proses implementasi selanjutnya adalah pemasangan library nuclex framework sebagai tambahan framework XNA. Fungsi utama dari nuclex framework adalah memungkinkan aplikasi yang dibuat menggunakan XNA memiliki komponen antarmuka seperti pada *Windows Presentation Form* (WPF). Berkas nuclex framework dapat diunduh di http://nuclexframework.codeplex.com. Pada implementasi pengguna nuclex framework, penulis hanya akan menggunakan dua library dari nuclex framework, yaitu Nuclex.Input.dll untuk merespon masukan dari pengguna dan Nuclex.UserInterface.dll untuk kebutuhan komponen antarmuka. Untuk menggunakan nuclex framewok, library harus dimasukkan ke dalam refrensi project dan dipanggil fungsinya dalam kode.

5.4 Implementasi Proses Melakukan Gerakan

Berikut merupakan langkah-langkah implementasi proses melakukan gerakan sesuai dengan perancangan proses yang telah dilakukan.

5.4.1 Implementasi Proses Memulai Permainan

Pada implementasi ini dibuat fungsi untuk menjalankan kelas utama dari aplikasi yaitu Game1. Kelas untuk memulai permainan bernama Program yang memuat fungsi untuk memanggil kelas Game1. Kelas Program merupakan *main class* dari aplikasi. Kode fungsi dalam kelas Program dapat dilihat pada Potongan Kode 5.2.

```
static void Main(string[] args)
        {
            using (Game1 game = new Game1())
            {
                game.Run();
            }
        }
```

Potongan Kode 5.2 Implementasi Mulai Latihan

5.4.2 Implementasi Proses Melihat Status Sensor

Implementasi proses melihat status sensor terdapat pada kelas KinectCheck. Untuk melakukan pengecekan terhadap status sensor, pertama dilakukan inisiasi sensor. Implementasi inisiasi sensor akan menggunakan fungsi *property* karena inisiasi pada sensor ini akan digunakan pada kelas-kelas yang membutuhkan. Implementasi inisiasi sensor dapat dilihat pada Potongan Kode 5.3.

```
public KinectSensor kinect { get; private set; }

public KinectStatus lastStatus { get; private set; }
```

Potongan Kode 5.3 Inisiasi Sensor

Variabel **kinect** digunakan untuk menginisiasi sensor kinect yang digunakan sedangkan variabel **lastStatus** digunakan untuk menginiasi status-status yang dimiliki oleh sensor kinect.

Selanjutnya adalah iniasi daftar status yang dimiliki oleh sensor kinect. Penulis akan menggunakan *Dictonary* dalam implementasi daftar status sensor. Implementasi daftar status sensor dapat dilihat pada Potongan Kode 5.4.

```
this.statusKinect.Add(KinectStatus.Initializing, "Initializing . . . ");
this.statusKinect.Add(KinectStatus.Connected, "Connected");
this.statusKinect.Add(KinectStatus.Disconnected, "Kinect is Disconnect, Please Connect
Kinect Sensor !");
this.statusKinect.Add(KinectStatus.NotPowered, "Kinect is Not Powered");
this.statusKinect.Add(KinectStatus.NotReady, "Kinect is Not Ready");
this.statusKinect.Add(KinectStatus.Error, "Error");
this.statusKinect.Add(KinectStatus.InsufficientBandwidth, "Insufficient Bandwidth");
```

Potongan Kode 5.4 Daftar Status Sensor Kinect

Untuk melakukan cek status sensor, dibuat fungsi FindSensor(). Pada fungsi FindSensor, aplikasi akan melakukan cek status sensor secara berulang hingga didapatkan status *Connected*. Selain itu jika ditemukan status *Connected* maka FindSensor() akan mengaktifkan fungsi-fungsi utama sensor seperti *ColorStream*, *DepthStream*, dan *SkeletonStream*. Implementasi fungsi FindSensor() dapat dilihat pada Potongan Kode 5.5.

```
private void FindSensor()
        {
            this.kinect = KinectSensor.KinectSensors.FirstOrDefault();

            if (this.kinect != null)
            {

                this.lastStatus = this.kinect.Status;


                if (this.lastStatus == KinectStatus.Connected)
                {
                    try
                    {
                        //this.kinect.ElevationAngle = 0;
                        this.kinect.SkeletonStream.Enable();
                        this.kinect.ColorStream.Enable(this.colorImageFormat);
                        this.kinect.DepthStream.Enable(this.dephImageFormat);

                        this.kinect.SkeletonStream.EnableTrackingInNearRange = true;

                        try
                        {
                            this.kinect.Start();
                        }

                        catch (IOException)
                        {
                            this.kinect = null;
                        }
                    }

                    catch (InvalidOperationException)
                    {
                        this.kinect = null;
                    }
                }
            }

            else
            {
                this.lastStatus = KinectStatus.Disconnected;
            }

        }
```

Potongan Kode 5.5 Fungsi FindSensor()

Langkah terakhir dari subproses ini adalah menampilkan pesan status kinect jika sensor tidak terbuhung atau kondisi lainnya selain *Connected*. Jika status sensor *Connected* maka akan langsung berlanjut ke proses selanjutnya dan status tidak ditampilkan. Untuk menampilkan status ini digunakan fungsi

Draw() yang merupakan fungsi dasar dari *framework* XNA untuk menampilkan objek. Implementasi fungsi Draw() dapat dilihat pada Potongan Kode 5.6.

```
public override void Draw(GameTime gameTime)
        {
            base.Draw(gameTime);
            if (this.SharedSpriteBatch == null)
            {
                this.Initialize();
            }

            if (this.kinect == null || this.lastStatus != KinectStatus.Connected)
            {
                this.SharedSpriteBatch.Begin();
                string status = "Kinect is Disconnect, Please Connect Kinect Sensor
!"; // Nilai awal status kinect

                if (this.kinect != null)
                {
                    status = this.statusKinect[this.lastStatus];
                }

                //Render statusKinect
                Vector2 sizeText = this.font.MeasureString(status);
                this.SharedSpriteBatch.DrawString(this.font, status,
                    new Vector2((Game.GraphicsDevice.Viewport.Width - sizeText.X) / 2,
(Game.GraphicsDevice.Viewport.Height - sizeText.Y) - 10 ), Color.White);
                this.SharedSpriteBatch.End();
            }

        }
```

Potongan Kode 5.6 Fungsi Draw() untuk Menampilkan Status Sensor Kinect

5.4.3 Implementasi Proses Melakukan Gerakan

Proses memelakukan gerakan diawal dengan mengaktifkan fungsi *SkeletonStream* yang telah dilakukan pada proses melihat status sensor pada fungsi FindSensor(). Selanjutnya gerakan pengguna akan dibaca menggunakan *frame* dari fungsi *SkeletonStream*. Fungsi *SkeletonStream* sendiri diimplementasi pada bagian fungsi Update() pada kelas Game1. Implementasi pembacaan gerakan pengguna dapat dilihat pada Potongan Kode 5.7.

```
using (var skeletonFrame = this.Chooser.kinect.SkeletonStream.OpenNextFrame(1000))
            {

                // Sometimes we get a null frame back if no data is ready
                if (null != skeletonFrame)
                {
                    newFrame = true;
                    if (this.curState != GameState.welcomeScreen)
                    {
                        this.frame++;
                    }
                    ++this.totalFrames;
                    this.GetFPS();
                    // Reallocate if necessary
                    if (null == SkeletonData || SkeletonData.Length != 
skeletonFrame.SkeletonArrayLength)
                    {
                        SkeletonData = new Skeleton[skeletonFrame.SkeletonArrayLength];
                    }

                    skeletonFrame.CopySkeletonDataTo(SkeletonData);

                    // Select the first tracked skeleton we see to avateer
                    Skeleton rawSkeleton =
                        (from s in SkeletonData
                            where s != null && s.TrackingState == 
SkeletonTrackingState.Tracked
                            select s).FirstOrDefault();
```

Potongan Kode 5.7 Pembacaan Gerakan

Subproses selanjutnya adalah sistem akan membaca model tiga dimenasi yang akan digunakan. Pembacaan model tiga dimensi menggunakan kelas SkinnedModelProcessor yang bertugas untuk membaca data struktur kerangka dan animasi pada model. Kelas ini dapat diimplentasi pada fitur *property* model tiga dimenasi. Implementasi kelas SkinnedModelProcessor dapat dilihat pada Potongan Kode 5.8.

```
public override ModelContent Process(NodeContent input, ContentProcessorContext context)
        {
            ValidateMesh(input, context, null);

            // Find the skeleton.
            BoneContent skeleton = MeshHelper.FindSkeleton(input);
            if (skeleton == null)
            {
                throw new InvalidContentException("Input skeleton not found.");
            }
            //Convert to local coordinate
            FlattenTransforms(input, skeleton);
            // Read the bind pose and skeleton hierarchy data.
            IList<BoneContent> bones = MeshHelper.FlattenSkeleton(skeleton);
            if (bones.Count > SkinnedEffect.MaxBones)
            {
                throw new InvalidContentException(
                    string.Format(
                    "Skeleton has {0} bones, but the maximum supported is {1}.",
                    bones.Count,
                    SkinnedEffect.MaxBones));
            }
            List<Matrix> bindPose = new List<Matrix>();
            List<Matrix> inverseBindPose = new List<Matrix>();
            List<int> skeletonHierarchy = new List<int>();

            foreach (BoneContent bone in bones)
            {
                bindPose.Add(bone.Transform);
                inverseBindPose.Add(Matrix.Invert(bone.AbsoluteTransform));
                skeletonHierarchy.Add(bones.IndexOf(bone.Parent as BoneContent));
            }

            // Chain to the base ModelProcessor class so it can convert the model data.
            ModelContent model = base.Process(input, context);

            // Store our custom animation data in the Tag property of the model.
            model.Tag = new SkinningData(bindPose, inverseBindPose, skeletonHierarchy);

            return model;
        }
```

Potongan Kode 5.8 Kelas SkinnedModelProcessor

Jika model memiliki data struktur kerangka, data tersebut akan disimpan dalam kelas SkinnedData yang nantinya akan digunakan untuk mengolah animasi model tiga dimenasi bersamaan dengan data struktur kerangka pengguna. Implementasi SkinnedData dapat dilihat pada Potongan Kode 5.9.

```
public class SkinningData
    {

        public SkinningData(List<Matrix> bindPose, List<Matrix> inverseBindPose,
List<int> skeletonHierarchy)
        {
            BindPose = bindPose;
            InverseBindPose = inverseBindPose;
            SkeletonHierarchy = skeletonHierarchy;
        }

        private SkinningData()
        {
        }


        [ContentSerializer]
        public List<Matrix> BindPose { get; private set; }

        [ContentSerializer]
        public List<Matrix> InverseBindPose { get; private set; }

        [ContentSerializer]
        public List<int> SkeletonHierarchy { get; private set; }
    }
```

Potongan Kode 5.9 Kelas SkinnedData

5.4.4 Implementasi Proses Melihat Animasi Model Tiga Dimensi

Proses melihat animasi model tiga dimenasi bertugas untuk transformasi data skeleton yang didapat dari pengguna agar sesuai dengan data skeleton yang dimiliki oleh model tiga dimensi. Proses transformasi dilakukan pada fungsi SetJointTransformation. Pada fungsi ini seluruh informasi sendi akan ditransoformasi sesuai dengan jenisnya masing-masing. Contoh implementasi bagian sendi pinggang pada fungsi SetJointTransformation dapat dilihat pada Potongan Kode 5.10.

```
private void SetJointTransformation(BoneOrientation bone, Skeleton skeleton, Matrix
bindRoot, ref Matrix[] boneTransforms)
        {

            if (bone.StartJoint == JointType.HipCenter && bone.EndJoint ==
JointType.HipCenter)
            {

                bindRoot.Translation = Vector3.Zero;
                Matrix invBindRoot = Matrix.Invert(bindRoot);
                Matrix hipOrientation =
KinectHelper.Matrix4ToXNAMatrix(bone.HierarchicalRotation.Matrix);

                Matrix pelvis = boneTransforms[1];
                pelvis.Translation = Vector3.Zero;
                Matrix invPelvis = Matrix.Invert(pelvis);

                Matrix combined = (invBindRoot * hipOrientation) * invPelvis;

                this.ReplaceBoneMatrix(JointType.HipCenter, combined, true, ref
boneTransforms);
            }
```

Potongan Kode 5.10 Fungsi Transformasi HipCenter

Selelah seluruh sendi ditransformasikan sesuai dengan kerangka yang dimiliki oleh model tiga dimenasi, selanjutnya sendi ini akan dipasangkan pada model tiga dimensi dan menggantikan kerangka yang dimiliki model. Untuk itu digunakan fungsi ReplaceBoneMatrix untuk memasangkan sendir hasil transformasi pada model tiga dimensi. Implementasi fungsi ReplaceBoneMatrix dapat dilihat pada Potongan Kode 5.11.

```
private void ReplaceBoneMatrix(JointType joint, Matrix boneMatrix, bool
replaceTranslationInExistingBoneMatrix, ref Matrix[] boneTransforms)
        {
            int meshJointId;
            bool success = this.nuiJointToAvatarBoneIndex.TryGetValue(joint, out
meshJointId);

            if (success)
            {
                Vector3 offsetTranslation = boneTransforms[meshJointId].Translation;
                boneTransforms[meshJointId] = boneMatrix;

                if (replaceTranslationInExistingBoneMatrix == false)
                {
                 boneTransforms[meshJointId].Translation = offsetTranslation;
              }
            }
        }
```

Potongan Kode 5.11 Fungsi ReplaceBoneMatrix()

Selelah proses transformasi selesai, selanjutnya hasil transformasi kerangka pengguna akan ditransformasikan kembali terhadap matrix world yang merupakan representasi dari lokasi model berada. Selanjutnya dilakukan transformasi perubahan skin dan mesh dari model sesuai dengan transformasi sendi pengguna. Implementasi transformasi terhadap world matrix diwakili fungsi UpdateWorldTransforms dan tranformasi skin dan mesh model diwaliki oleh fungsi UpdateSkinTransforms. Kedua implementasi fungsi tersebut dapat dilihat pada Potongan Kode 5.12.

```
  private void UpdateWorldTransforms(Matrix rootTransform)
        {

            this.worldTransforms[0] = this.boneTransforms[0] * rootTransform;


            for (int bone = 1; bone < this.worldTransforms.Length; bone++)
            {
                int parentBone = this.skinningDataValue.SkeletonHierarchy[bone];


                this.worldTransforms[bone] = this.boneTransforms[bone] *
this.worldTransforms[parentBone];
            }
        }

 private void UpdateSkinTransforms()
        {
            for (int bone = 0; bone < this.skinTransforms.Length; bone++)
            {
                this.skinTransforms[bone] = this.skinningDataValue.InverseBindPose[bone]
* this.worldTransforms[bone];
            }
        }
```

Potongan Kode 5.12 Fungsi UpdateWorldTransforms() dan UpdateSkinTransforms()

Tahapan terakhir dari proses melihat animasi model adalah menampilkan animasi model pada tampilan antarmuka. Dalam menampilkan model tiga dimenasi diperlukan fungsi prespektif atau sudut padang kamera untuk mengetahui sudut padang model yang akan ditampilkan. Fungsi ini diimplementasi oleh fungsi UpdateViewingCamera pada kelas Game1. Implemetasi fungsi UpdateViewingCamera dapat dilihat pada Potongan Kode 5.13.

```
protected void UpdateViewingCamera()
        {
            GraphicsDevice device = this.graphics.GraphicsDevice;

            // Compute camera matrices.
            this.view = Matrix.CreateTranslation(0, -CameraHeight, 0) *

Matrix.CreateRotationY(MathHelper.ToRadians(this.cameraRotation)) *
                            Matrix.CreateRotationX(MathHelper.ToRadians(this.cameraArc)) *
                            Matrix.CreateLookAt(
                                                    new Vector3(0, 0, -this.cameraDistance),
                                                    new Vector3(0, 0, 0),
                                                    Vector3.Up);

            // Kinect vertical FOV in degrees
            float nominalVerticalFieldOfView = 45.6f;

            if (null != this.check && null != this.Chooser.kinect &&
this.Chooser.kinect.IsRunning && KinectStatus.Connected == this.Chooser.kinect.Status)
            {
                nominalVerticalFieldOfView =
this.check.kinect.DepthStream.NominalVerticalFieldOfView;
            }

            this.projection = Matrix.CreatePerspectiveFieldOfView(

nominalVerticalFieldOfView * (float)Math.PI / 180.0f,
device.Viewport.AspectRatio,1,10000);
        }
```

Potongan Kode 5.13 Fungsi UpdateViewingCamera()

Langkah terakhir adalah menampilkan model pada tampilan utama. Implementasi subproses ini dilakukan oleh fungsi Draw(). Fungsi Draw() dapat dilihat pada Potongan Kode.

```
public void Draw(GameTime gameTime, Matrix world, Matrix view, Matrix projection)
        {
            // Render the 3D model skinned mesh with Skinned Effect.
            foreach (ModelMesh mesh in this.avatarModel.Meshes)
            {
                foreach (SkinnedEffect effect in mesh.Effects)
                {
                    effect.SetBoneTransforms(this.skinTransforms);

                    effect.World = world;
                    effect.View = view;
                    effect.Projection = projection;

                    effect.EnableDefaultLighting();

                    effect.SpecularColor = new Vector3(0.25f);
                    effect.SpecularPower = 16;
                }

                mesh.Draw();
            }


            if (this.drawLocalAxes && null != this.localAxes)
            {

                Game.GraphicsDevice.DepthStencilState = DepthStencilState.None;

                foreach (Matrix boneWorldTrans in this.worldTransforms)
                {

                    this.localAxes.Draw(gameTime, boneWorldTrans * world, view,
projection);
                }


                Game.GraphicsDevice.DepthStencilState = DepthStencilState.Default;
            }

            if (this.drawBoneConstraintsSkeleton && null !=
this.boneOrientationConstraints && null != this.check)
            {
                this.boneOrientationConstraints.Draw(gameTime, this.skeleton,
this.check.SeatedMode, this.kinectLineSkeletonWorldOffsetMatrix * world, view,
projection);
            }

            this.SkeletonDrawn = true;

            base.Draw(gameTime);
        }
```

Potongan Kode 5.14 Fungsi Draw() Menampilkan Model 3D

5.4.5 Implementasi Proses Melihat Tampilan Depth Stream

Proses melihat tampilan depth stream diawali dengan mengaktifkan fungsi depth stream yang dimiliki oleh sensor kinect. Implementasi ini telah dilakukan pada proses melihat status sensor. Selanjutnya adalah inisiasi format *depth stream* yang digunakan. Berikut implementasi format *depth stream* yang digunakan dalam aplikasi.

```
this.chooser = new KinectCheck(this, ColorImageFormat.RgbResolution640x480Fps30,
    DepthImageFormat.Resolution320x240Fps30);
this.Services.AddService(typeof(KinectCheck), this.chooser);
```

Potongan Kode 5.15 Inisiasi Format ColorStream

Selanjutnya adalah menangkap seluruh perubahan yang terjadi yang ditangkap oleh sensor informa merah. kinect dan menyimpannya dalam variabel depthData. Data yang didapatkan dari depthData akan digunakan oleh frame yang akan menampilkan data *depth stream*. Implementasi ini terjadi pada fungsi Update() pada kelas DepthStreamRenderer.

```
using (var frame = this.Chooser.Kinect.DepthStream.OpenNextFrame(0))
            {
                if (null == frame)
                {
                    return;
                }

                if (null == this.depthData || this.depthData.Length !=
frame.PixelDataLength)
                {
                    this.depthData = new short[frame.PixelDataLength];

                    this.depthTexture = new Texture2D(
                        Game.GraphicsDevice,
                        frame.Width,
                        frame.Height,
                        false,
                        SurfaceFormat.Bgra4444);

                    this.backBuffer = new RenderTarget2D(
                        Game.GraphicsDevice,
                        frame.Width,
                        frame.Height,
                        false,
                        SurfaceFormat.Color,
                        DepthFormat.None,
                        this.Game.GraphicsDevice.PresentationParameters.MultiSampleCount,
                        RenderTargetUsage.PreserveContents);
                }

                frame.CopyPixelDataTo(this.depthData);
                this.needToRedrawBackBuffer = true;
            }
```

Potongan Kode 5.16 Pengolahan Data ColorStream

Proses terarkhir adalah menampilkan frame *depth stream* pada tampilan utama. Proses ini dilakukan oleh fungsi Draw() pada kelas DepthStreamRenderer yang akan dipanggil kembali oleh fungsi Draw() pada kelas TaekwondoTraining. Implementasi fungsi Draw() pada kelas DepthStreamRender dapat dilihat pada Potongan Kode 5.17.

```
public override void Draw(GameTime gameTime)
{
    if (null == this.depthTexture || null == this.SharedSpriteBatch)
    {
        return;
    }

    if (false == this.initialized)
    {
        this.Initialize();
    }

    if (this.needToRedrawBackBuffer)
    {

        Game.GraphicsDevice.SetRenderTarget(this.backBuffer);
        Game.GraphicsDevice.Clear(ClearOptions.Target, Color.Black, 1.0f, 0);

        this.depthTexture.SetData<short>(this.depthData);

        this.SharedSpriteBatch.Begin(SpriteSortMode.Immediate, null, null, null, null,
this.kinectDepthVisualizer);
        this.SharedSpriteBatch.Draw(this.depthTexture, Vector2.Zero, Color.White);
        this.SharedSpriteBatch.End();

        this.Game.GraphicsDevice.SetRenderTarget(null);
        this.needToRedrawBackBuffer = false;
    }

    this.SharedSpriteBatch.Begin();
    this.SharedSpriteBatch.Draw(
        this.backBuffer,
        new Rectangle((int)ObjectPosition.X, (int)ObjectPosition.Y, (int)ObjectSize.X,
(int)ObjectSize.Y),
        null,
        Color.White);
    this.SharedSpriteBatch.End();

    base.Draw(gameTime);
}
```

Potongan Kode 5.17 Fungsi Draw() untuk Menampilkan ColorStream

5.5 Implementasi Proses Perekaman Gerakan

Pada implementasi proses perekaman gerakan akan memanfaatkan fungsi-fungsi dari library tambahan yaitu Nuclex framework. Fungsi yang akan digunakan yaitu InputManager dan GUI. Proses implementasi yang pertama dilakukan adalah melakukan inisiasi InputControl dan GUI.

```
//Keyboard Text Input
private InputManager textInput;
private String enteredText;

//XNA GUI
private GuiManager gui;
private Screen mainScreen;
private bool kinectCon = false;
private InputControl fileName;
```

Potongan Kode 5.18 Inisiasi InputControl dan GUI

5.5.1 Implementasi Proses Memberi Nama Gerakan

Proses memberi nama gerakan digunakan untuk menangkap masukan keyboard dari pengguna digunakan fungsi getTextInput(). Implementasi fungsi getTextInput() dapat dilihat pada Potongan Kode 5.19.

```
private void getTextInput(Char character)
{
  if (character == 8)
  {
   if (this.enteredText.Length != 0)
       this.enteredText = this.enteredText.Substring(0, (enteredText.Length - 1));
  }
  else
  {
    this.enteredText += character;
  }
```

Potongan Kode 5.19 Fungsi getTextInput()

Proses implementasi terakhir adalah nemapilkan masukan pengguna pada tampilan GUI. Nama berkas yang dimasukkan oleh pengguna akan muncul pada InputControl yang terdapat pada fungs DesktopGUI(). Implementasi DesktopGUI dapat dilihat pada Potongan Kode 5.20.

```
this.fileName = new InputControl();
this.fileName.Bounds = new UniRectangle(new
UniVector((this.graphics.GraphicsDevice.Viewport.Width - 170), 175), new UniVector(160,
30));
this.fileName.Text = "" + this.enteredText;
```

Potongan Kode 5.20 Implementasi InputControl pada GUI

5.5.2 Implementasi Proses Memulai Perekaman

Implementasi ini diawali dengan pembuatan fungsi Serialize() dimana fungsi ini akan merekam seluruh koordinat sendi pengguna tiap frame, dan total frame. selama proses tidak dihentikan. Pada fungsi Serialize() juga diimplementasikan fungsi serialization yang dimiliki oleh C# untuk mengubah data parameter array skeleton menjadi byte dan disimpan pada memori sebelum dituliskan pada berkas binary. Implementasi fungsi Serialize() terdapat pada Potongan Kode 5.21.

```
    private void Serialize(Skeleton[] skel)
        {
            try
            {
                BinaryFormatter binF = new BinaryFormatter();
                binF.Serialize(this.fs, skel);
            }
            catch (Exception e)
            {
                e.ToString();
            }

        }
```

Potongan Kode 5.21 Fungsi Serialize()

Fungsi Serialize() akan berjalan ketika ada event bahwa tombol StartCapture telah ditekan oleh pengguna. Event ini akan memuat thread yang berisi fungsi Serialize(). Penggunaan thread dimaksudkan untuk mempermudah proses mengakhiri perekaman. Implementasi event tombol StartCapture terdapat pada Potongan Kode 5.22.

```
private void startCapture_Pressed(object sender, EventArgs e)
{
    this.path = @".\Records\" + this.enteredText + ".bin";
    if (File.Exists(this.path))
    {
      this.curState = GameState.errorScreen;
    }
      else
    {
      Thread capThread = new Thread(() => Serialize(SkeletonData));
      this.capThreadStatus = true;
      this.fs = new FileStream(this.path, FileMode.CreateNew);
      capThread.Start();
    }
```

Potongan Kode 5.22 Event Handling tombol StarCapture

Pada event ini juga berkas teks yang digunakan untuk menyimpan informasi sendi pengguna dibuat oleh sistem namun dalam keadaan kosong.

5.5.3 Implementasi Proses Mengenhentikan Perekaman

Implementasi proses mengenhentikan perekaman berbentuk event yang akan mengehentikan proses mulai perekaman. Event yang digunakan dalam proses ini adalah ketika tombol StopCapture ditekan, maka thread yang berisi fungsi SkelCapture() akan dihentikan. Implementasi event StopCapture dapat dilihat pada Potongan Kode 5.23.

```
private void stopCapture_Pressed(object sender, EventArgs e)
{
    Thread capThread = new Thread(() => Serialize(SkeletonData);
    this.capThreadStatus = false;
    this.fs.Dispose();
    capThread.Abort();
}
```

Potongan Kode 5.23 Event Handling Tombol StopCapture

Pada saat yang bersamaan setelah penghentian perekaman, maka informasi sendi pengguna akan dituliskan dalam berkas teks yang telah dibuat.

5.6 Implementasi Proses Pencocokan Gerakan

Pada implementasi proses pencocokan gerakan dibagi menjadi 2 subproses sesuai dengan perencanaan yang telah dilakukan. Secara umum implementasi pada proses ini melakukan fungsi *replay* dan pencocokan gerakan. Fungsi *replay* yang dimaksud adalah melakukan animasi model sesuai dengan data rekaman yang telah dilakukan sebelumnya. Kemudian pencocokan adalah fungsi untuk mencocokkan *recorededSkeleton* dan *actualSkeleton* tiap frame. Gerakan pertama yang akan digunakan sebagai acuan adalah *yop-jireugi* atau *side punch*. Gerakan ini tidak termasuk pada gerakan dasar yang direncanakan karena keterbatasan sensor. Gerakan ini terdapat pada rangkaian gerakan taekwondo bernama *Taebaek*. Gerakan kedua yang dijadikan acuan adalah *are maki* atau gerakan tangkisan yang merupakan gerakan dasar.

5.6.1 Implementasi Proses Melihat Hasil Pencocokan

Impeletantasi proses melihat gerakan diwakili oleh sub proses *replay* dan pencocokan. Implementasi *replay* dilakukan oleh fungsi Deserialize() yang memiliki nilai kembalian berupa array dari data *Skeleton*. Fungsi Deserialize() dapat dilihat pada Potongan Kode 5.24.

```
private Skeleton[] Deserialize()
{
  Skeleton[] record;

  BinaryFormatter binF = new BinaryFormatter();
  record = (Skeleton[])binF.Deserialize(this.fs);
  if (this.fs.Length == this.fs.Position)
  {
    this.fs.Dispose();
    this.curState = GameState.hasilScreen;
  }
    return record;
}
```

Potongan Kode 5.24 Fungsi Deserialize()

Fungi Deserialize() melakukan tugas untuk membaca berkas binary yang dihasilkan oleh fungsi Serialize() pada proses perekaman gerakan, dan dikembalikan lagi menjadi data yang dapat dikenali oleh sistem. Hasil dari fungsi Deserialize() akan digunakan sebagai acuan animasi model tiga dimensi. Pemanfaatan fungsi Deserialize() dapat dilihat pada Potongan Kode 5.25.

```
this.recSkel =
       (from s in this.Deserialize()
        where s != null && s.TrackingState == SkeletonTrackingState.Tracked
        elect s).FirstOrDefault();
```

Potongan Kode 5.25 Penggunaan Fungsi Deserialize()

Tahapan selanjutnya adalah melakukan kalkulasi pencocokan gerakan tiap frame. Implementasi pencocokan gerakan terdapat pada fungsi Compare(). Potongan fungsi Compare() dapat dilihat pada Potongan Kode.

Setelah dilakakukan pecocokan gerakan, maka sistem akan menampilkan hasil pencocokan tiap framenya. Hasil akan ditampilkan menggunakan fungsi Draw(). Fungsi menampilkan hasil pencocokan tiap frame dapat dilihat pada Potongan Kode 5.26.

```
this.SharedSpriteBatch.Begin();
this.SharedSpriteBatch.DrawString(this.font, fps, new Vector2(10, 10), Color.Gold);
this.SharedSpriteBatch.DrawString(this.font, "Count Frame : " + this.frame.ToString(),
new Vector2(10, 25), Color.Gold);
this.SharedSpriteBatch.DrawString(this.font, angle, new Vector2(10, 40), Color.Gold);
this.SharedSpriteBatch.DrawString(this.jFont, "Judge :"+this.jString, new
Vector2(GraphicsDevice.Viewport.Width / 2, 300), Color.Gold);
this.SharedSpriteBatch.End();
```

Potongan Kode 5.26 Menampilkan Hasil Pencocokan

Hingga tahapan ini, proses melihat hasil pencocokan tiap frame telah selesai. Proses akan dilanjutkan dengan mencocokkan gerakan seluruh frame dan melakukan kalkulasi secara keseluruhan. Proses ini terdapat pada proses melihat hasil latihan.

```
var gX = new double[] { g1x, g2x, g3x, g4x, g5x, g6x, g7x, g8x, g9x, g10x, g11
g13x, g14x, g15x, g16x, g17x, g18x, g19x, g20x };
var gY = new double[] { g1y, g2y, g3y, g4y, g5y, g6y, g7y, g8y, g9y, g10y, g11
g13y, g14y, g15y, g16y, g17y, g18y, g19y, g20y };

if (gX.All(jx => jx <= 0.2500) && gY.All(jy => jy <= 0.2500))
{
  this.jString = "Correct";
  this.correctFrame++;
}
else
{
  this.jString = "false";
  this.falseFrame++;
}
```

Potongan Kode 5.27 Fungsi Compare

5.6.2 Implementasi Proses Melihat Hasil Latihan

Proses melihat hasil latihan merupakan proses kelanjutan dari proses melihat hasil pencocokan. Proses melihat hasil latihan merupakan kalkulasi akhir dari keseluruhan set gerakan yang telah dilakukan. Implementasi dari proses melihat hasil latihan diwakili oleh fungsi Calculation(). Fungsi Calculation() dapat dilihat pada Potongan Kode 5.28.

```
private void Calculation()
{
    if (this.curState == GameState.hasilScreen)
    {
        this.calcResult = Math.Round(((double)this.correctFrame / this.fr
        this.jString = "Your Performance is " + this.calcResult.ToString(
    }
}
```

Potongan Kode 5.28 Implementasi Fungsi Calculation()

Fungsi Calculation() memiliki dua paramaeter bertipe integer yaitu totalFrame dan correctFrame. Parameter totalFrame merupakan nilai dari keseluruhan frame satu set gerakan, sedangkan correctFrame merupakan frame dimana terdapat gerakan yang benar. Keluaran dari fungsi Calculation adalah prosestase frame yang memiliki nalai benar yang ditampung dalam variabel result.

5.7 Implementasi Atarmuka

Dalam subbab ini akan dibahas mengenai implementasi dari rancangan antarmuka aplikasi yang telah dibahas sebelumnya. Sesuai dengan tahap perancangan yang telah dilakukan sebelumnya, antarmuka aplikasi terdiri dari 5 tampilan antarmuka yang merepresentasikan tiap *state* dalam aplikasi. Perpindahan antar *state* dilakukan dengan pembuatan *enumeration* pada aplikasi sesuai dengan kebutuhan aplikasi. Tampilan *state* akan diatur dalam satu fungsi dalam kelas KinectTaekwondoMod yaitu fungsi Draw(). Implementasi *enumeration* dapat dilihat pada Potongan Kode.

```
public enum GameState
{
  welcomeScreen,
  latihanScreen,
  rekamScreen,
  hasilScreen,
  errorScreen
}
```

Potongan Kode 5.29 Implementasi Enumerasi Gamestate

Penjelasan enumeration sebagai berikut,

1. welcomeScreen merupakan state untuk main menu.

2. latihanScreen merupakan state untuk pecocokan gerakan.
3. rekamScreen meruapakan state untuk perekaman gerakan.
4. hasilScreen merupakan state untuk hasil akhir latihan.
5. errorScreen merupakan state untuk melakukan *handling* apabila pengguna memasukkan nama berkas yang sama. Pada state ini pengguna akan diminta untuk kembali ke state rekamScreen hingga nama untuk berkas sesuai.

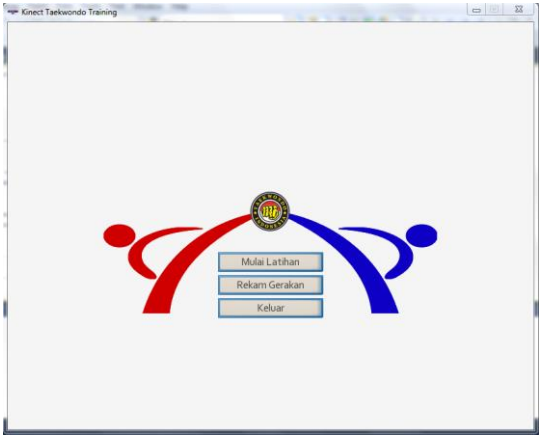
State kinect status tidak dimasukkan kedalam enumeration karena state kinect status akan diload secara otomatis ketika status kinect memiliki nilai selain *connected*. Implementasi state ini terdapat pada kelas KinectCheck.cs. Seluruh fungsi dari kelas KinectCheck.cs akan dimuat secara otomatis dengan fungsi XNA yaitu, Component.Add(). Implementasi state kinect status pada kelas utama dapat dilihat pada Potongan Kode 5.30.

```
this.chooser = new KinectCheck(this, ColorImageFormat.RgbResolution640x480Fps30,
DepthImageFormat.Resolution320x240Fps30);
this.Services.AddService(typeof(KinectCheck), this.chooser);

this.chooser.NearMode = false;
this.chooser.SeatedMode = false;

this.Components.Add(this.chooser);
```

Potongan Kode 5.30 Memasukkan KinectCheck.cs Pada Component

5.7.1 Implementasi State Main Menu

Berikut merupakan implementasi tampilan antarmuka dari state main menu.



Gambar 5.7 Tampilan Main Menu

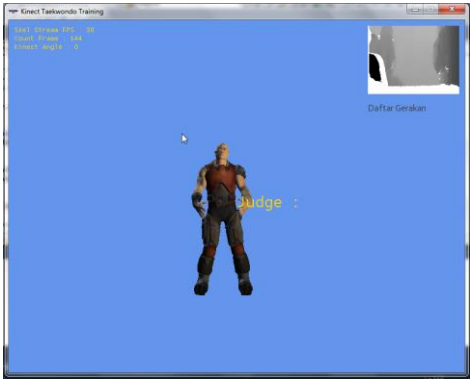
5.7.2 Implementasi State Kinect Status

Berikut meruapakan salah satu implementasi state kinect status. Keseluruhan state kinect status akan dibahas dalam subbab uji coba sistem.

Gambar 5.8 Tampilan Kinect Status

5.7.3 Implementasi State Pecocokan Gerakan

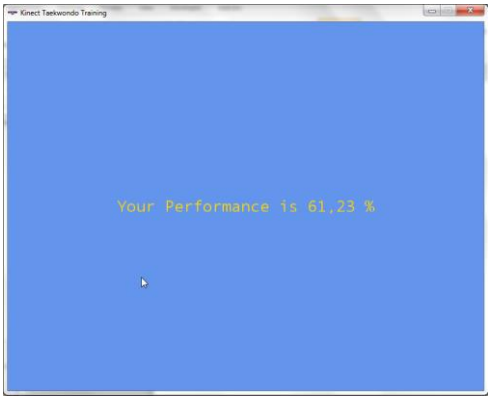
Berikut merupakan implementasi dari state pencocokan gerakan.



Gambar 5.9 Tampilan Pencocokan Gerakan

5.7.4 Implementasi State Hasil Akhir Latihan

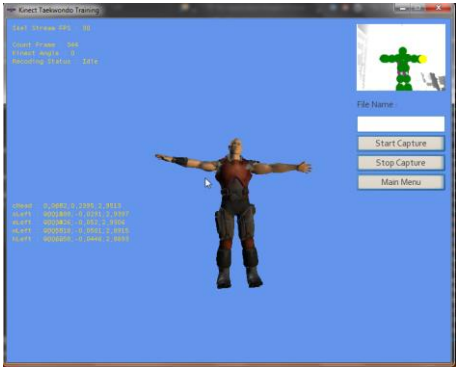
Berikut merupakan implementasi state hasil akhir latihan.



Gambar 5.10 Tampilan Hasil Akhir Latihan

5.7.5 Implementasi State Perekaman

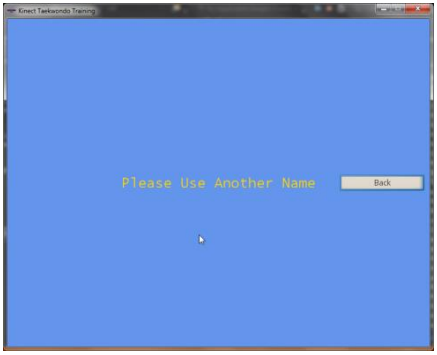
Berikut merupakan implementasi dari state perekaman.



Gambar 5.11 Tampilan Perekaman

5.7.6 Implementasi State Error

Berikut merupakan implementasi *state* errorScreen.



Gambar 5.12 Tampilan Error

5.8 Uji Coba Sistem

Pada sub bab ini akan dibahas mengenai uji coba sistem yang telah dibuat. Uji coba akan dilakukan dalam 2 tahapan yaitu uji coba berdasarkan use case yang telah dibuat dan uji coba performa aplikasi dalam lingkungan uji coba. Lingkungan uji coba yang akan digunakan dalam uji coba sistem terdapat pada Tabel 5.3.

Tabel 5.3 Spesifikasi Lingkungan Uji Coba

Spesifikasi	
Prosesor	Intel Core i7 Q740 @1.73 Ghz
Memori	8 GB RAM
Kartu Grafis	NVIDIA GeFroce GT425M 2GB
Sistem Operasi	Windows 7 Home Premium 64-bit
Tools Pendukung	1. Microsoft Visual Studio 2013 Profesional (Student Lisence). 2. FRAPS (Free lisence). 3. Kiwi Application Monitor (Free Lisence).

5.8.1 Uji Coba Toleransi Kalkulasi

Uji coba toleransi kalkulasi dilakukan untuk mendapatkan nilai toleransi dalam kalkulasi penilaian gerakan latihan. Pemberian toleransi dikarenakan keterbatasan sensor yang tidak dapat membaca sendi secara sempurna dan keterbatasan sensor dalam membaca kecepatan gerak pengguna.

Untuk mendapatkan nilai toleransi, dilakukan percobaan yang membandingkan hasil perekaman gerakan acuan dengan gerakan langsung dari pengguna. Percobaan ini dilakukan pada 20 sendi yang terbaca oleh sensor pada satu set gerakan. Dari percobaan tersebut didapatkan rata-rata selisih (Δ)

nilai koordinat X dan Y. Nilai koordinat X dan Y menjadi acuan dalam kalkulasi hasil latihan. Data hasil percobaan dapat dilihat pada Tabel 5.4.

Tabel 5.4 Hasil Uji Coba Toleransi Kalkulasi

No	Sendi	Δ X (m)	Δ Y (m)
1	*AnkleLeft*	0,132466784	0,009946713
2	*AnkleRight*	0,014825913	0,008429622
3	*ElbowLeft*	0,138961068	0,129848899
4	*ElbowRight*	0,084784746	0,025486557
5	*FootLeft*	0,133443789	0,014986829
6	*FootRight*	0,017974592	0,010257875
7	*HandLeft*	0,276476692	0,214657378
8	*HandRight*	0,082490647	0,034281892
9	*Head*	0,083178614	0,029824833
10	*HipCenter*	0,078666355	0,030589551
11	*HipLeft*	0,077921526	0,030758907
12	*HipRight*	0,078176462	0,029872683
13	*KneeLeft*	0,109570465	0,014100891
14	*KneeRight*	0,046378461	0,007901217
15	*ShoulderCenter*	0,082412972	0,029171146
16	*ShoulderLeft*	0,083046884	0,03528456
17	*ShoulderRight*	0,08385806	0,025119762
18	*Spine*	0,080028464	0,029222661
19	*WristLeft*	0,237801848	0,199782126
20	*WristRight*	0,078509614	0,019537492

Dilakukan perhitungan untuk mencari rata-rata nilai ΔX dan ΔY. Didapatkan dinilai ΔX yaitu 0,18 dan nilai ΔY yaitu 0,1. Pencarian nilai rata-rata dilakukan untuk mencari nilai Δ general sehingga dapat diaplikasikan kepada seluruh sendi untuk mempermudah proses kalkulasi. Penulis penggunakan

nilai Δ 0,18 sebagai nilai toleransi awal koordinat X maupun Y.

Selanjutnya dilakukan pecobaan untuk mengetahui apakah nilai Δ yang dihasilkan dari pecobaan sebelumnya akurat sebagai acuan toleransi kalkulasi. Akurat atau tidaknya gerakan dinilai berdasarkan hasil kalkulasi sistem dibandingkan dengan penilaian ahli. Hasil percobaan keakuratan kalkulasi berdasarkan Δ dapat dilihat pada Tabel 5.5.

Tabel 5.5 Hasil Uji Nilai Delta

No	Nilai Δ (m)	Hasil
1	0.0500	Tidak akurat
2	0.0750	Tidak akurat
3	0.1000	Tidak akurat
4	0.1250	Tidak akurat
5	0.1600	Tidak akurat
6	0.1800	Tidak akurat
7	0.2000	Tidak akurat
8	0.2250	Tidak akurat
9	0.2500	Akurat
10	0.2750	Tidak akurat
11	0.3000	Tidak akurat
12	0.3250	Tidak akurat
13	0.3500	Tidak akurat
14	0.3750	Tidak akurat
15	0.4000	Tidak akurat

Dari hasil percobaan keakuratan nilai Δ diketahui bahwa nilai Δ 0,18 tidak dapat digunakan sebagai acuan toleransi karena memberikan nilai kalkukasi yang terlalu rendah, namun gerakan yang dilakukan adalah benar menurut ahli. Sehingga nilai Δ yang

digunakan adalah 0,25 dikarenakan dapat memberikan hasil kalkukali yang baik dari segi sistem maupun ahli.

5.8.2 Uji Coba Akurasi Kalkulasi Penilaian Aplikasi

Uji coba ini dilakukan untuk mengetahui prosentase hasil kalkulasi ketepatan gerakan yang dilakukan oleh aplikasi dibandingkan dengan penilaian ahli. Penilaian oleh ahli dibagi menjadi 3 kriteria yaitu,

1. Benar (***G**ood*).
   Kriteria gerakan benar adalah gerakan yang sesuai dengan contoh yang diberikan ahli. Gerakan ahli direpresentasikan oleh gerakan avatar. Kuda-kuda dan pukulan atau tangkisan harus sesuai dengan gerakan yang dicontohkan.

Gambar 5.13 Contoh Benar Gerakan *Yop-Jireugi*

(Sumber : https://www.kukkiwon.or.kr/english/images/information/030203a3.gif)

Gambar 5.14 Contoh Benar Gerakan *Are Maki*

(Sumber : http://clubcentral.free.fr/Pages/Images/ap%20segui%20are%20maki.jpg)

2. Cukup (***O****k*).
   Kriteria cukup didasarkan pada kriteria benar dengan toleransi yang dianggap wajar oleh ahli. Contoh, untuk set gerakan yop-jireugi, toleransi diberikan pada bentuk pukulan dan kuda-kuda. Pukulan dianggap wajar jika posisinya tidak bergerser lebih dari 10 derajat dari posisi seharusnya (lihat Gambar 5.13). Untuk kuda-kuda *juchum seogi* tidak boleh terlalu duduk atau tegak.
3. Salah (***F****alse*).
   Kriteria salah dinilai dari gerakan yang sama sekali tidak merepresentasikan set gerakan.

5.8.2.1 Hasil Uji Coba Gerakan *Yop-Jireugi*

Uji coba gerakan yop-jireugi dilakukan sebanyak 120 kali percobaan. Hasil uji coba dapat dilihat pada Tabel 5.6.

Tabel 5.6 Hasil Uji Coba Gerakan *Yop-Jireugi*

No	Penilaian Ahli	Penilaian Aplikasi
1	Salah	64,84 %
2	Cukup	79,72 %
3	Benar	89,95 %

Selanjutnya dilakukan uji coba manipulasi koordinat z atau kedalaman pada gerakan *yop-jireugi*. Fokus manipulasi gerakan yaitu pada gerakan pukulan. Hasil uji coba manipulasi gerakan terhadapa koordinat z dapat dilihat pada Tabel 5.7.

Tabel 5.7 Hasil Uji Coba Manipulasi Z Pada Gerakan *Yop-Jireugi*

No	Manipulasi Z	Penilaian Aplikasi
1	+10˚	85,61 %
2	+45˚	85,51 %
3	-10˚	85,51 %

Untuk mengetahui penilaian aplikasi terhadap gerakan yang sengaja disalahkan dengan kondisi kesalahan yang ekstrim atau fatal dilakukan uji coba kondisi ekstrim. Hasil uji coba kondisi ekstrim pada gerakan yop-jireugi dapat dilihat pada Tabel 5.8.

Tabel 5.8 Hasil Uji Coba Kondisi Ekstrim Gerakan Yop-Jireugi

No	Kondisi Ekstrim	Penilaian Aplikasi
1	Gerakan ke arah berlawanan	30,31%
2	Pukulan kearah	64,41 %

	depan (*momtong-jireugi*)	

5.8.2.2 Hasil Uji Coba Gerakan *Are Maki*

Uji coba fungsional gerakan *are maki* dilakukan sebanyak 80 kali percobaan. Hasil dari uji coba gerakan are maki dapat dilihat pada Tabel 5.9.

Tabel 5.9 Hasil Uji Coba Gerakan *Are Maki*

No	Penilaian Ahli	Penilaian Aplikasi
1	Salah	55,21 %
2	Cukup	80,20 %
3	Benar	83,28 %

Uji coba manipulasi koordinat z (kedalaman) pada gerakan *are maki* tidak memungkinkan untuk dilakukan dikarenakan bentuk gerakan yang unik. Uji coba dilanjutkan dengan menguji kondisi ekstrim. Hasil dari uji coba kondisi ekstrim pada gerakan *are maki* dapat dilihat pada Tabel 5.10.

Tabel 5.10 Hasil Uji Coba Kondisi Ekstrim Gerakan *Are Maki*

No	Kondisi Ekstrim	Penilaian Aplikasi
1	Gerakan ke arah berlawanan	15,75%

Keseluruhan hasil uji coba gerakan *are maki* dapat dilihat pada lampiran D2.

5.8.3 Uji Coba Fungsional

Uji coba fungsional dilakukan untuk mengetahui apakah seluruh fitur utama aplikasi dapat berjalan dengan baik. Uji coba fungional diterapkan pada seluruh use case sesuai dengan perencanaan. Beberapa uji fungsional yang dilakukan adalah,

1. Pengujian pendeteksi gerakan pengguna.
   Pada pengujian ini, gerakan pengguna dapat dibaca oleh sensor dan sendi pengguna dapat ditransformasikan ke model tiga dimensi sehingga model dapat dianimasikan sesuai dengan gerakan pengguna.
2. Pengujian pemberian nama gerakan.
   Pemberian nama gerakan berhasil dilakukan. Penggunaan bermacam karakter dalam penamaan juga dapat dilakukan. Akan tetapi sistem tidak dapat melakukan duplikasi nama otomatis jika ditemukan nama gerakan yang telah ada sebelumnya.
3. Pengujian perekaman gerakan.
   Fitur perekaman dan pemutaran kembali berkas hasil perekaman dapat dilakukan sistem dengan baik.
4. Pengujian pencocokan gerakan secara langsung.
   Pada pengujian ini, sistem berhasil mencocokkan gerakan pengguna dengan gerakan model sesuai urutan frame yang ada.
5. Pengujian kalkulasi hasil akhir latihan.
   Pada pengujian kalkulasi hasil akhir latihan, sistem sudah dapat melakukan kalkulasi dengan baik dengan toleransi kalkulasi yang telah diberikan pada pengujian pencocokan gerakan.

Keseluruhan hasil uji coba fungsional secara lengkap dapat dilihat pada Lampiran B1.

5.8.4 Uji Coba Non-Fungsional

Pada subbab ini akan dibahas mengenai hasil uji coba non-fungsional aplikasi.

5.8.4.1 Uji Coba Jarak Optimal Sensor

Uji coba jarak sensor dilakukan untuk mengetahui jarak optimal sensor terhadap pengguna. Jarak optimal dibutuhkan agar seluruh sendi kerangka pengguna dapat terbaca oleh sensor sehingga mengoptimalkan proses kalkukasi performa latihan. Hasil uji coba jarak sensor dapat dilihat pada Tabel.

Tabel 5.11 Hasil Uji Coba Jarak Sensor

No	Jarak	Hasil
1	0,7 m	Sendi tidak terbaca
2	1,7 m	Sendi tidak terbaca sempurna pada bagian kaki
3	2,7 m	Sendi terbaca sempurna
4	3,7 m	Sendi tidak terbaca sempurna pada bagian kaki

Pada jarak 0,7 meter sendi tidak terbaca karena jarak minimal sensor adalah 1,2 meter. Jarak ini tetap dimasukkan dalam pengujian untuk mengetahui kemungkinan sendi terbaca. Pada jarak 1,7 meter, seluruh sendi dapat terbaca kecuali kaki kanan dan kaki kiri. Pada jarak 3,7 meter, kedua sendi kaki kembali terbaca *inferred* oleh sensor. Hal ini dikarenakan pengguna hampir berada pada

batas maksimum jangkauan sensor yaitu 4 meter.

5.8.4.2 Uji Coba Performa Aplikasi

Uji coba performa dilakukan untuk mengetahui spesifikasi perangkat minimum untuk menjalankan aplikasi. Pengujian dilakukan di 4 lingukungan pengujian berbeda. Lingkungan uji coba yang digunakan dapat dilihat pada Tabel.

Tabel 5.12 Lingkungan Pengujian

Lingkungan Pengujian 1 (LP1)	
CPU	Intel Core i7 Q740 @1,73 Ghz
RAM	8 GB
GPU	NVIDIA GeForce GT425M – 2GB
Lingkungan Pengujian 2 (LP2)	
CPU	Intel Core i5-3330 @ 3.00 Ghz
RAM	4 GB
GPU	NVDIA GeForce GTX 650 Ti – 1.5 GB
Lingkungan Pengujian 3 (LP3)	
CPU	Intel Core 2 Duo E7500 @ 2.93 Ghz
RAM	4 GB
GPU	Intel On Board
Lingkungan Pengujian 3 (LP4)	
CPU	Intel Core 2 Duo E7500 @ 2.93 Ghz
RAM	2 GB
GPU	Intel On Board

Hasil uji coba performa aplikasi dapat dilihat pada Lampiran C1.

5.9 Analisa Hasil Uji Coba Sistem

Setelah dilakukan uji coba sistem, terdapat faktor-faktor yang mempengaruhi kinerja sistem. Untuk itu dilakukan analisa untuk mengetahui faktor-faktor tersebut.

Dari hasil uji toleransi, ditemukan bahwa sistem memerlukan toleransi untuk menentukan hasil akhir latihan. Hal ini disebabkan oleh keterbatasan sensor dalam membaca kecepatan gerak pengguna dan keterbatasan sensor dalam membaca persendian yang tidak dapat sepenuhnya akurat karena penggunaan avatar. Selain itu diketahui bahwa manipulasi kedalaman koordinat sendi (sumbu z) tidak mempengaruhi aplikasi dalam hasil pencocokan gerakan dan kalkulasi hasil latihan.

Dari hasil uji coba jarak optimal sensor, diketahui bahwa jarak optimal sensor terhadap pengguna adalah 2,7 meter. Pada jarak kurang dari 1,2 meter, persendian pengguna tidak dapat terbaca. Sedangkan pada jarak 1,7 meter dan 3,7 meter, pesendian pengguna tidak dapat terbaca sempurna oleh sensor atau terbaca *inferred* pada bagian kaki. Maksud dari *inferred* adalah sendi dapat terbaca dengan bantuan algoritma perkiraan posisi sendi oleh sensor. Sendi yang terbaca *inferred* ditandai oleh *node* berwarna kuning pada tampilan *depth stream*. Pada uji coba jarak optimal diketehaui pula bahwa postur tubuh pengguna tidak mempengaruhi sensor dalam membaca sendi pengguna.

Uji coba performa aplikasi dilakukan untuk mengetahui spesifikasi minimal perangkat komputer yang mampu menjalankan aplikasi dengan baik. Berdasarkan hasil uji coba pada 4 lingkungan pengujian, aplikasi dapat berjalan dengan baik pada spesifikasi perangkat,

1. CPU : Setara Core i5.
2. GPU : GPU On Board.
3. Memori : 2 GB (Memori minimal Kinect).

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Dari pengerjaan tugas akhir dapat disimpulkan bahwa,

1. Teknologi *motion capture* pada Microsoft Kinect dapat dimanfaatkan sebagai sarana latihan beladiri khususnya taekwondo dengan batasan, gerakan harus menghadap ke sensor, memerlukan toleransi dalam kalkulasi koordinat sendi pengguna, dan kecepatan pengguna harus sesuai dengan model acuan (avatar).
2. Perhitungan pada jari tangan tidak dilakukan karena keterbatasan sensor dan SDK.
3. Aplikasi tidak melakukan perhitungaan terhadap koordinat kedalaman (sumbu z) sendi pengguna.
4. Terdapat temnuan bahwa postur atau ukuran tubuh pengguna tidak mempengaruhi kemampuan sensor dalam membaca sendi-sendi kerangka pengguna.

6.2 Saran

Dalam pengembangan aplikasi *basic taekwondo training system* masih terdapat banyak kekurangan dalam proses pengembangannya. Untuk itu terdapat beberapa hal yang dapat dijadikan acuan untuk pengembangan aplikasi selanjutnya, yaitu

1. Aplikasi *basic taekwondo training system* merupakan penelitian untuk menguji apakah sensor kinect mampu untuk melakukan kalkulasi setiap frame pada kasus gerakan-gerakan bela diri khususnya taekwondo. Sehingga untuk kedepannya dapat dilakukan pengujian gerakan bela diri lain atau gerakan lain seperti senam atau tari.
2. Penggunaan pelatih virtual menggunakan avatar menjadi masalah utama dari pencocokan gerakan

dikarenakan avatar tidak dapat melakukan gerakan persis dengan data kerangka. Dengan masalah ini perlu dilakukan penelitian lebih lanjut bagaimana cara pemanfaatan avatar dengan lebih baik dalam pembacaan data kerangka.

3. Munculnya teknologi baru dari sensor kinect yaitu Kinect for Windows 2.0 atau Kinect for XBOX One serta Kinect for Windows SDK 2.0 dapat dimanfaatkan untuk penelitian selanjutnya karena kemampuannnya dalam mendeteksi 25 titik sendi manusia.

DAFTAR PUSTAKA

3ds Max Tutorial. (t.thn.). *Skin Modifier*. Dipetik Oktober 7, 2014, dari 3ds Max Tutorial: http://www.3dmax-tutorials.com/Skin_Modifier.html

Autodesk. (2014). *3ds Max Overview*. Dipetik Oktober 5, 2014, dari Autodesk: http://www.autodesk.com/products/3ds-max/overview

Berg, T., Chattopadhyay, D., Schedel, M., & Vallier, T. (t.thn.). *Interactive Music : Human Motion Initiated Music Generation Using Skeletal Tracking By Kinect.*

Catuhe, D. (2012). *Programming With The Kinect for Windows Software Development Kit.* Washington: Microsoft Press.

Jana, A. (2012). *Kinect for Windows SDK Programming Guide.* Birmingham: Packt Publishing.

Jean, J. S. (2012). *Kinect Hacks : Tips & Tools for Motion and Pattern Detection.* California: O'Reilly.

Microsoft. (2010). *Getting Started with XNA Game Studio Development*. Dipetik Februari 7, 2014, dari Microsoft Developer Network: http://msdn.microsoft.com/en-us/library/cc178930.aspx

Microsoft. (2010). *Introducing Visual Studio.* Dipetik Ferbuari 7, 2014, dari Microsoft Developer Network: http://msdn.microsoft.com/en-us/library/fx6bk1f4(v=vs.100).aspx

Microsoft. (2013). *Frequently Ask Question*. Dipetik Februari 6, 2014, dari Kinect fot Windows: http://www.microsoft.com/en-us/kinectforwindows/faq.aspx

Microsoft. (2013). *Human Interface Guidelines.* Microsoft.

Microsoft. (2014). *Depth Stream*. Dipetik Februari 7, 2014, dari Microsoft Developer Network: http://msdn.microsoft.com/en-us/library/jj131028.aspx

Microsoft. (2014). *Kinect Studio*. Dipetik Februari 7, 2014, dari Microsoft Developer Network: http://msdn.microsoft.com/en-us/library/hh855389.aspx

Muhammad, D. (2013, Desember 27). *Kapolri Klaim Kriminalitas 2013 Menurun*. Dipetik Maret 26, 2014, dari Republika Online: http://www.republika.co.id/berita/nasional/hukum/13/12/27/mygtns-kapolri-klaim-kriminalitas-2013-menurun

Nike. (2012). *Nike+ Kinect Training*. Dipetik Desember 12, 2014, dari Nike: http://www.nike.com/us/en_us/c/training/nike-plus-kinect-training

Pramudita, R. N. (2012). *Rancang Bangun Modul Pendeteksi Gerakan Secara Waktu Nyata Pada Permainan SKJ Ekspres Menggunakan Microsoft Kinect SDK.* Surabaya: ITS.

Slick, J. (t.thn.). *What is Rigging*. Dipetik Oktober 5, 2014, dari About Technology: http://3d.about.com/od/Creating-3D-The-CG-Pipeline/a/What-Is-Rigging.htm

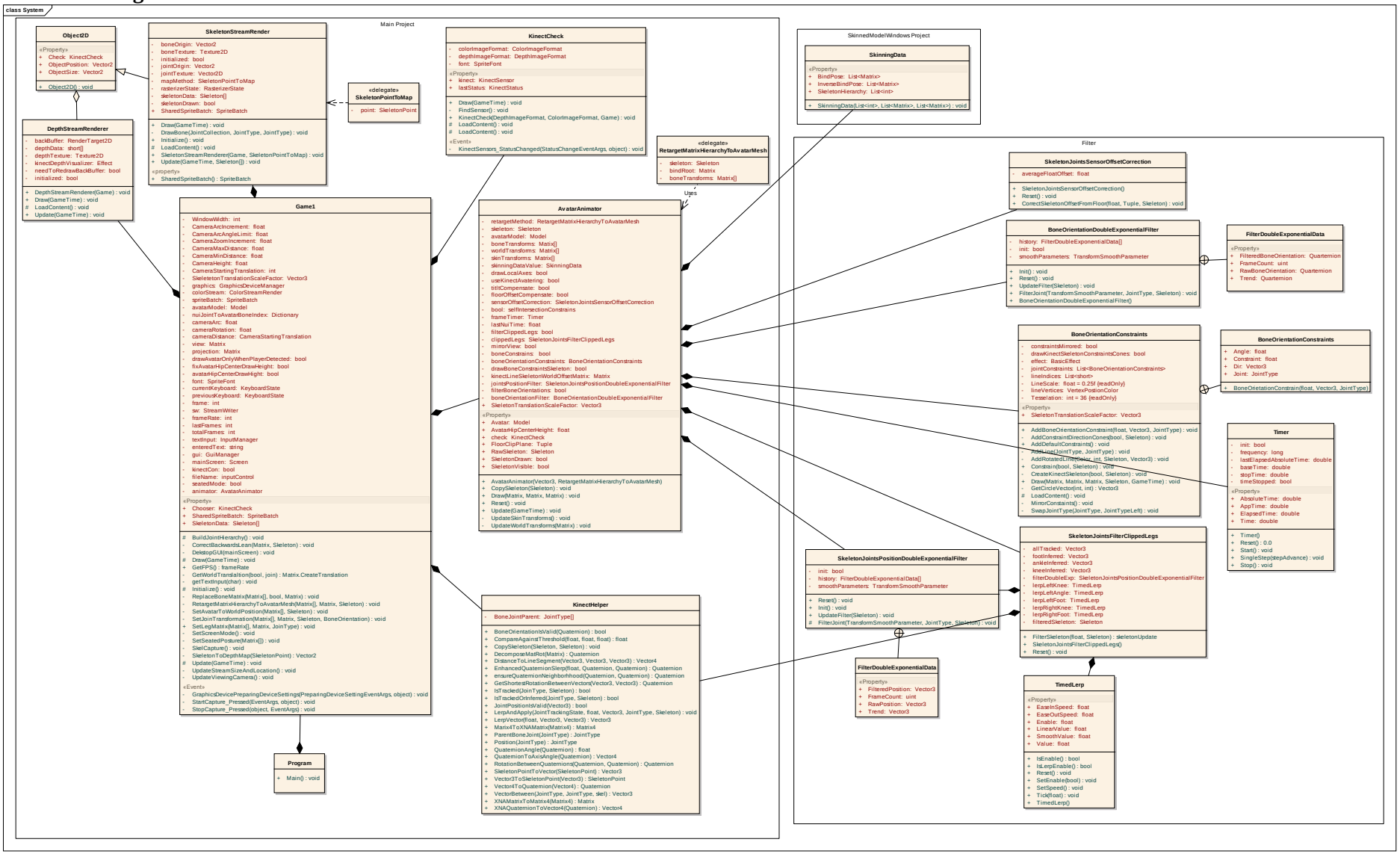
Vij, M., & Dawn, S. (2014). Corrective Self Defense Training Unit using sensing of Kinect maps. *International Journal of Computer Applications*, 8-11.

Wittenberg, I. (2013). *Performance and Motion Capture Using Multiple Xbox Kinect.* Rochester Institute of Technology.

World Taekwondo Federation. (2013). *WTF : World Taekwondo Federation*. Dipetik Oktober 7, 2014, dari What is Taekwondo:

http://www.worldtaekwondofederation.net/what-is-taekwondo

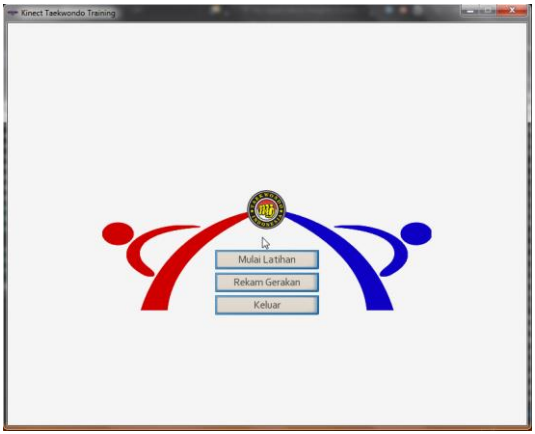
A1. Class Diagram



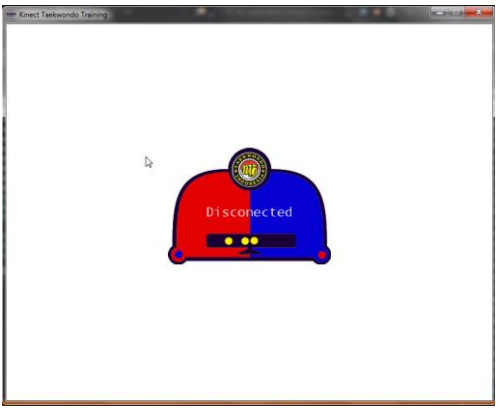
B1. Uji Coba Fungsional

Uji Coba Fungsional 1	
UC-101	Memulai Latihan
Aktor	Pengguna
Kondisi	Spesifikasi lingkungan uji coba
Hasil	
Alur Normal	1. Aplikasi berjalan optimal. 2. Render grafis berjalan pada 60 fps. 3. Konsumsi memori rata-rata 148 MB. 4. Konsumsi CPU pada proses normal rata-rata 20%.
Alur Alternatif	-
Kesimpulan	1. Aplikasi berjalan dengan baik pada lingkungan uji coba. 2. Pengujian fungsional 1 akan dilanjutkan dengan pengujian non-fungsional 1.

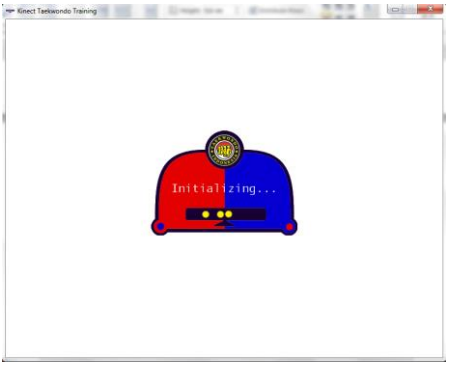
Uji Coba Fungsional 2	
UC-102	Melihat Status Sensor Kinect
Aktor	Pengguna
Kondisi	Status Sensor : **Connected** Sensor kinect terhubung dengan perangkat komputer

	sesuai dengan spesifikasi lingkungan uji coba.
Hasil	
Alur Normal	Pengguna dapat menjalankan aplikasi dengan baik. Aplikasi langsung masuk ke state utama. 
Alur Alternatif	-
Kesimpulan	Dengan kondisi status sensor *connected* aplikasi berjalan sesuai dengan alur normal.

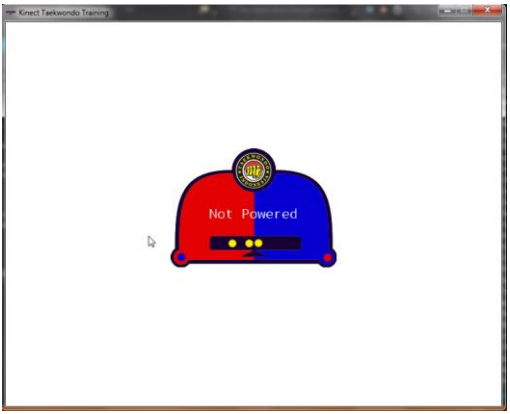
Uji Coba Fungsional 3	
UC-102	Melihat Status Sensor Kinect
Aktor	Pengguna

Kondisi	Status Sensor : **Disconnected** Sensor kinect tidak terhubung dengan perangkat komputer.
Hasil	
Alur Normal	-
Alur Alternatif	Aplikasi menampilkan *state handling* untuk status sensor *disconnected*. 
Kesimpulan	*State handling* kinect status berjalan dengan baik.

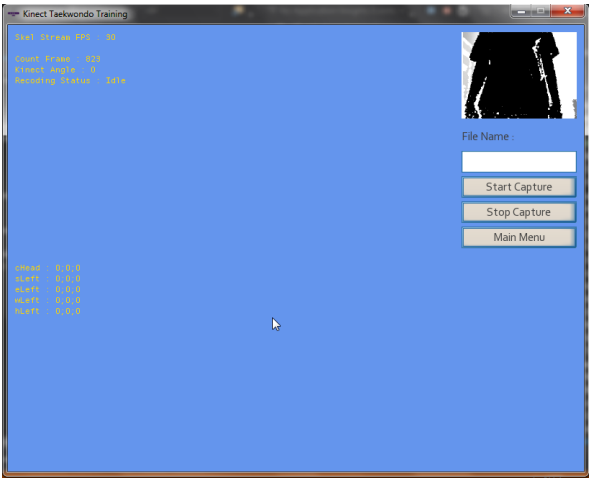
Uji Coba Fungsional 4	
UC-102	Melihat Status Sensor Kinect
Aktor	Pengguna
Kondisi	Status Sensor : **Initializing** Sensor kinect dalam keadaan baru dihubungkan pada

	perangkat komputer pada saat aplikasi telah dijalankan.
Hasil	
Alur Normal	-
Alur Alternatif	Aplikasi menampilkan *state handling* untuk status sensor *initializing*. 
Kesimpulan	*State handling* kinect status berjalan dengan baik.

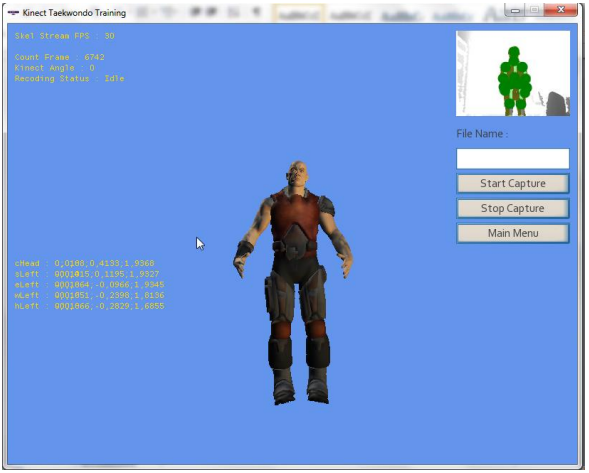
Uji Coba Fungsional 5	
UC-102	Melihat Status Sensor Kinect
Aktor	Pengguna
Kondisi	Status Sensor : **Not Powered** Sensor dalam keadaan tehubung oleh perangkat komputer namun tidak terhubung dengan arus listrik.

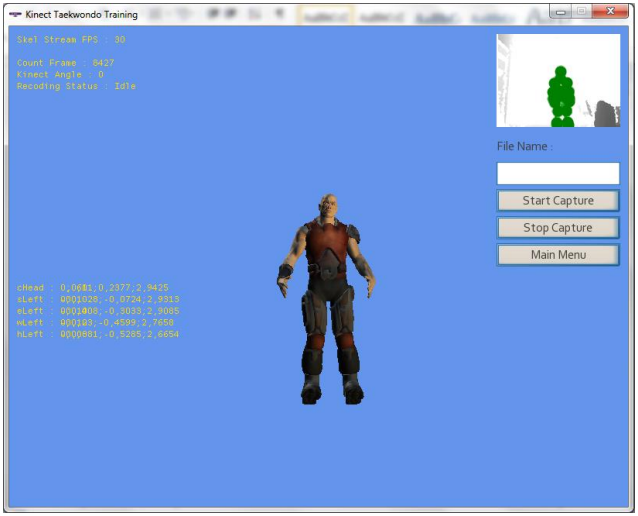
Hasil	
Alur Normal	-
Alur Alternatif	Aplikasi menampilkan *state handling* untuk status sensor *not powered*. 
Kesimpulan	*State handling* kinect status berjalan dengan baik.

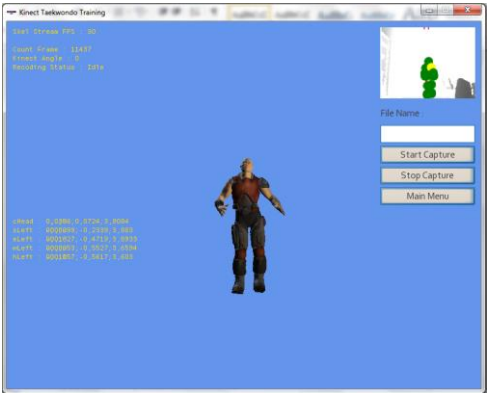
Uji Coba Fungsional 6	
UC-103 / **UC-104**	Melakukan Gerakan / Melihat Avatar
Aktor	Pengguna
Kondisi	1. Pengguna berada pada jarak 0,7. Tinggi sensor 0,84 meter dari permukaan. 2. Sudut sensor terhadap pengguna 0 derajat.
Hasil	

Alur Normal	-
Alur Alternatif	1. Avatar tidak dapat terbaca karena tubuh pengguna hanya terbaca sebagian oleh sensor. 2. DepthStream dapat menanggap pengguna tetapi tidak dapat memetakan kerangka pengguna. 3. Penggua tetap dapat melakukan gerakan tetapi fungsi aplikasi tidak dapat berjalan. 
Kesimpulan	Pengguna harus berada dalam jangkauan sensor. Fungsi aplikasi berjalan dengan baik.

Uji Coba Fungsional 7	
UC-103 / **UC-104**	Melakukan Gerakan / Melihat Avatar

Aktor	Pengguna
Kondisi	1. Pengguna berada pada jarak 1,7 meter 2. Tinggi sensor 0,84 meter dari permukaan. 3. Sudut sensor terhadap pengguna 0 derajat.
Hasil	
Alur Normal	1. Sensor dapat membaca seluruh kerangka tubuh pengguna kecuali kaki. 2. Penggua dapat melakukan gerakan dan avatar sudah dapat diproyeksikan kedalam ruang 3 dimensi aplikasi. 
Alur Alternatif	
Kesimpulan	Pembacaan gerakan pengguna oleh sensor pada jarak ini belum optimal.

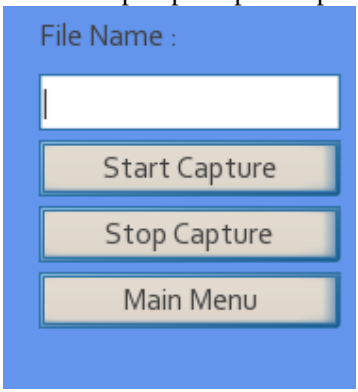
Uji Coba Fungsional 8	
UC-103 / **UC-104**	Melakukan Gerakan / Melihat Avatar
Aktor	Pengguna
Kondisi	1. Pengguna berada pada jarak 2,7 meter 2. Tinggi sensor 0,84 meter dari permukaan. 3. Sudut sensor terhadap pengguna 0 derajat.
Hasil	
Alur Normal	1. Gerakan pengguna dibaca optimal oleh sensor. 2. Gerakan avatar dapat sesuai dengan pengguna. 
Alur Alternatif	-
Kesimpulan	Jarak optimal untuk melakukan gerakan adalah 2,7.

Uji Coba Fungsional 9	
UC-103 / **UC-104**	Melakukan Gerakan / Melihat Avatar
Aktor	Pengguna
Kondisi	1. Pengguna berada pada jarak 3,7 meter 2. Tinggi sensor 0,84 meter dari permukaan. 3. Sudut sensor terhadap pengguna 0 derajat.
Hasil	
Alur Normal	1. Gerakan pengguna masih dapat dibaca oleh sensor namun tidak optimal. 2. Tanda kuning pada depthStream menandakan sendi tidak dapat dibaca baik oleh sensor melainkan hasil algoritma perkiraan aplikasi. 
Alur Alternatif	-

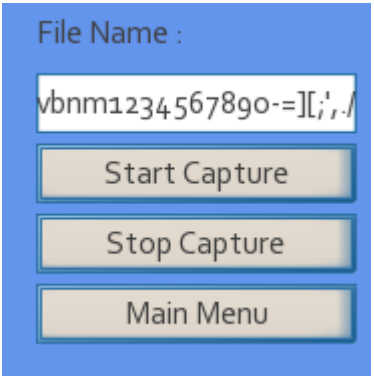
Kesimpulan	Jarak ini sudah dapat digunakan lagi untuk melakukan latihan atau perekaman dikarenakan sensor tidak dapat membaca kerangka pengguna dengan optimal.

Uji Coba Fungsional 10	
UC-105	Melihat Tampilan Depth Stream
Aktor	Pengguna
Kondisi	1. Aplikasi berjalan dengan baik. 2. Kondisi cahaya terang. 3. Sensor terhubung dengan perangkat komputer.
Alur Normal	1. Tampilan depth stream dapat ditampilkan dengan baik. 
Alur Alternatif	-
Kesimpulan	Depth Stream dapat berjalan dengan baik

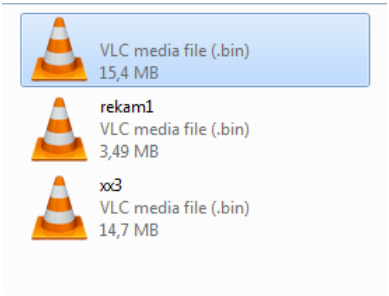
Uji Coba Fungsional 11	
UC-201	Memberi Nama Gerakan

Aktor	Pengguna
Kondisi	1. Aplikasi berjalan dengan baik. 2. Pengguna tidak menuliskan nama gerakan.
Hasil	
Alur Normal	1. Tidak berdampak pada proses perekaman. 
Alur Alternatif	-
Kesimpulan	Aplikasi berjalan dengan baik.

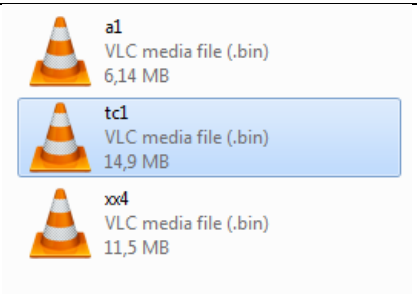
Uji Coba Fungsional 12	
UC-201	Memberi Nama Gerakan
Aktor	Pengguna
Kondisi	1. Aplikasi berjalan dengan baik. 2. Pengguna menuliskan nama gerakan.
Hasil	
Alur Normal	1. Proses perekaman berjalan normal. 2. Seluruh karakter dapat digunakan.

	
Alur Alternatif	-
Kesimpulan	Aplikasi berjalan dengan baik. Seluruh karakter dapat digunakan.

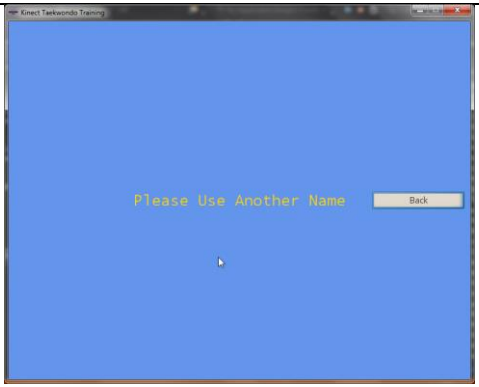
Uji Coba Fungsional 13	
UC-202	Mulai Perekaman
Aktor	Pengguna
Kondisi	1. Aplikasi berjalan dengan baik. 2. Perekaman dalam kondisi ideal. 3. Waktu perekaman 1 menit. 4. Pengguna tidak menuliskan nama gerakan.
Hasil	
Alur Normal	1. Proses perekaman tetap berjalan. 2. Berkas binary hasil perekaman tidak memiliki nama. 3. Ukuran berkas cukup besar.

	
Alur Alternatif	-
Kesimpulan	Aplikasi tidak memberikan peringatan karena pengguna tidak memberikan nama gerakan.

Uji Coba Fungsional 14	
UC-202	Mulai Perekaman
Aktor	Pengguna
Kondisi	1. Aplikasi berjalan dengan baik. 2. Perekaman dalam kondisi ideal. 3. Waktu perekaman 1 menit. 4. Pengguna menuliskan nama gerakan (tc1).
Hasil	
Alur Normal	1. Proses perekaman tetap berjalan. 2. Berkas binary hasil perekaman dengan nama sesuai dengan masukan pengguna. 3. Ukuran berkas cukup besar.

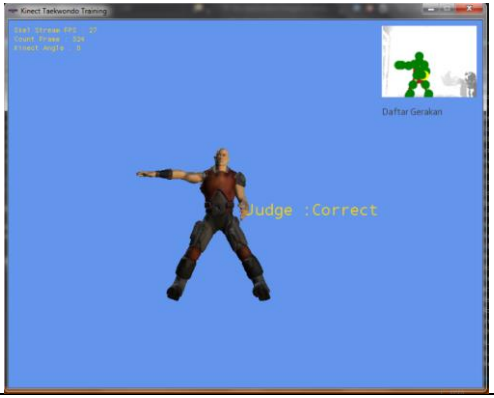
	
Alur Alternatif	-
Kesimpulan	Proses perekaman berjalan sesuai dengan perencanaan aplikasi.

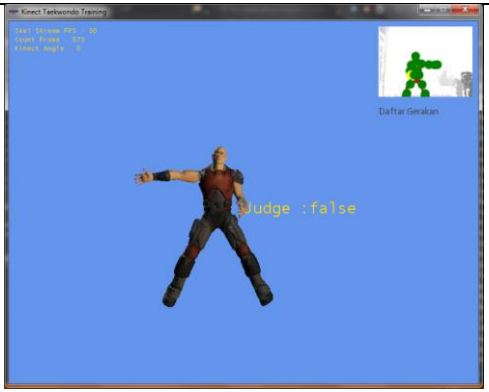
Uji Coba Fungsional 15	
UC-202	Mulai Perekaman
Aktor	Pengguna
Kondisi	1. Aplikasi berjalan dengan baik. 2. Perekaman dalam kondisi ideal. 3. Waktu perekaman 1 menit. 4. Pengguna menuliskan nama gerekan yang sudah ada (tc1).
Hasil	
Alur Normal	1. Sistem akan mengarahkan pada tampilan rekam. 2. Pengguna diminta untuk kembali ketampilan perekaman.

	
Alur Alternatif	-
Kesimpulan	Aplikasi belum bisa meng-*handling* untuk melakukan duplikat berkas dengan nama berbeda secara otomatis.

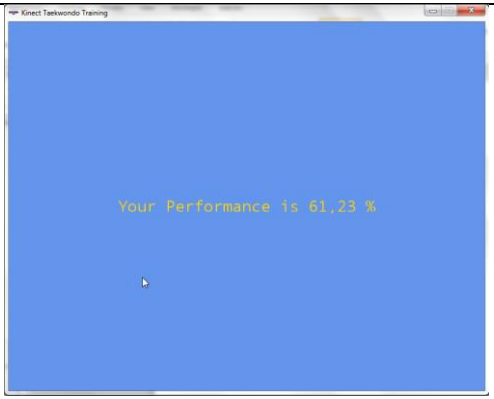
Uji Coba Fungsional 16	
UC-203	Menghentikan Perekaman
Aktor	Pengguna
Kondisi	1. Aplikasi berjalan dengan baik. 2. Fungsi perekaman sedang berjalan
Hasil	
Alur Normal	1. Proses perekaman dapat dihentikan dengan baik. 2. Berkas binary tersimpan.
Alur Alternatif	-

Kesimpulan	Aplikasi dapat menghentikan perekaman dengan baik.

Uji Coba Fungsional 17	
UC-203	Melihat Hasil Pencocokan
Aktor	Pengguna
Hasil	
Kondisi	1. Aplikasi berjalan dengan baik. 2. Fungsi perekaman sedang berjalan.
Alur Normal	1. Sistem dapat mencocokkan gerakan sesuai dengan toleransi yang diberikan. 2. Toleransi sebesar 0.25 m. 

	
Alur Alternatif	-
Kesimpulan	Pencocokan gerakan dapat berjalan sesuai dengan perencanaan aplikasi.

Uji Coba Fungsional 18	
UC-203	Melihat Hasil Akhir Latihan
Aktor	Pengguna
Kondisi	1. Aplikasi berjalan dengan baik. 2. Fungsi perekaman sedang berjalan.
Hasil	
Alur Normal	1. Sistem dapat mencocokkan gerakan sesuai dengan toleransi yang diberikan. 2. Perhitungan prosentase sesuai dengan kalkulasi jumlah frame benar dibandingkan dengan keseluruhan frame latiha.

	
Alur Alternatif	-
Kesimpulan	Hasil latihan dapat ditampilkan dengan hasil kalkulasi yang benar.

C1. Uji Coba Non-Fungsional

	Kondisi	CPU	GPU	Memori	Penggunaan CPU	FPS	Penggunaan Memori
1	Normal	Intel Core i7 Q740 @1.73 Ghz	NVIDIA GeForce GT425M – 2 GB	8 GB	20% - 30%	Min : 56 fps Max : 62 fps Avg : 60 fps	±148 MB
2		Intel Core i5 3330 @3.00 Ghz	NVIDIA GeForce GTX 480 – 1,5 GB	4 GB	20% - 30%	Min : 56 fps Max : 62 fps Avg : 60 fps	±148 MB
3		Intel Core 2 Duo E7500 @2.93 Ghz	Intel on Board	4 GB	60% - 90%	Min : 48 fps Max : 62 fps Avg : 57 fps	±148 MB
4		Intel Core 2 Duo E7500 @2.93 Ghz	Intel on Board	2 GB	90% - 100%	Min : 38 fps Max : 61 fps Avg : 54 fps	±148 MB
5	Perekaman	Intel Core i7 Q740 @1.73 Ghz	NVIDIA GeForce GT425M – 2 GB	8 GB	20% - 50%	Min : 52 fps Max : 62 fps Avg : 60 fps	±153 MB
6		Intel Core i5 3330 @3.00 Ghz	NVIDIA GeForce GTX 480 – 1,5 GB	4 GB	20% - 50%	Min : 30 fps Max : 62 fps Avg : 39 fps	±153 MB
7		Intel Core 2 Duo E7500 @2.93 Ghz	Intel on Board	4 GB	60% - 80%	Min : 47 fps Max : 62 fps Avg : 57 fps	±153 MB
8		Intel Core 2 Duo E7500 @2.93 Ghz	Intel on Board	2 GB	90% - 100%	Min : 15 fps Max : 62 fps Avg : 54 fps	±154 MB
9	Latihan	Intel Core i7 Q740 @1.73 Ghz	NVIDIA GeForce GT425M – 2 GB	8 GB	20% - 30%	Min : 56 fps Max : 62 fps Avg : 60 fps	±148 MB
10		Intel Core i5 3330 @3.00 Ghz	NVIDIA GeForce GTX 480 – 1,5 GB	4 GB	20% - 30%	Min : 56 fps Max : 62 fps Avg : 60 fps	±148 MB
11		Intel Core 2 Duo E7500 @2.93 Ghz	Intel on Board	4 GB	70% - 90%	Min : 45 fps Max : 61 fps Avg : 54 fps	±148 MB
12		Intel Core 2 Duo E7500 @2.93 Ghz	Intel on Board	2 GB	90% - 100%	Min : 38 fps Max : 60 fps Avg : 52 fps	±148 MB

D1. Hasil Uji Coba Penilaian Ahli Terhadap Kalkulasi Aplikasi Gerakan Yop-Jireugi

No	Benar (G)	Cukup (O)	Salah (F)	Penilaian Aplikasi (%)
1	√			81,36
2	√			81,72
3	√			84,62
4	√			80,99
5	√			86,44
6	√			86,68
7	√			80,99
8	√			86,32
9	√			88,01
10	√			87,53
11	√			81,48
12	√			81,6
13	√			85,96
14	√			82,32
15	√			73,73
16	√			83,05
17	√			85,59
18	√			87,41
19	√			83,9
20	√			85,23
21	√			79,9
22	√			87,7
23	√			71,75
24	√			81,6
25	√			78,21
26	√			79,42
27	√			79,9
28	√			85,96
29	√			80,75

30	√			81,36
31	√			91,4
32	√			89,71
33	√			87,29
34	√			89,83
35	√			86,92
36	√			89,47
37	√			87,53
38	√			88,01
39	√			81,96
40	√			84,34
41		√		74,82
42		√		70,94
43		√		76,27
44		√		75,67
45		√		77,24
46		√		78,57
47		√		79,9
48		√		74,82
49		√		75,67
50		√		79,9
51		√		78,45
52		√		83,9
53		√		76,76
54		√		73,12
55		√		79,9
56		√		71,91
57		√		84,14
58		√		66,83
59		√		80,99
60		√		80,39
61		√		75,79
62		√		84,14
63		√		84,26

64		√		81,84
65		√		83,9
66		√		76,76
67		√		86,68
68		√		79,3
69		√		82,69
70		√		85,71
71		√		75,91
72		√		81,23
73		√		80,87
74		√		84,99
75		√		85,35
76		√		83,17
77		√		85,11
78		√		85,35
79		√		83,17
80		√		82,57
81			√	60,05
82			√	67,8
83			√	64,89
84			√	65,74
85			√	57,02
86			√	64,04
87			√	56,05
89			√	62,47
90			√	58,6
91			√	62,47
92			√	60,53
93			√	63,44
94			√	62,71
95			√	67,19
96			√	57,99
97			√	57,38
98			√	59,32

99			√	64,65
100			√	58,11
101			√	72,03
102			√	68,4
103			√	76,27
104			√	70,82
105			√	71,43
106			√	69,01
107			√	68,77
108			√	74,21
109			√	69,01
110			√	70,22
111			√	61,99
112			√	69,49
113			√	68,64
114			√	70,7
115			√	72,52
116			√	62,95
117			√	69,25
118			√	63,08
119			√	67,31
120			√	69,85

D2. **Hasil Uji Coba Penilaian Ahli Terhadap Aplikasi Gerakan *Are Maki***

No	Benar (G)	Cukup (O)	Salah (F)	Penilaian Aplikasi (%)
1	√			87,93
2	√			80,54
3	√			83,52
4	√			81,68
5	√			84,23
6	√			83,52

7	√			86,08
8	√			85,23
9	√			85,51
10	√			83,81
11	√			85,23
12	√			86,65
13	√			80,54
14	√			83,81
15	√			84,09
16	√			85,37
17	√			84,52
18	√			86,51
19	√			84,94
20	√			84,09
21	√			87,22
22	√			85,23
23	√			86,51
24	√			85,23
25	√			83,38
26	√			84,38
27	√			83,95
28	√			77,56
29	√			75,99
30	√			81,39
31	√			75,28
32	√			80,54
33	√			83,95
34	√			84,52
35	√			85,8
36	√			82,67
37	√			75,28
38	√			81,82
39	√			81,68
40	√			80,97

41		√		77,27
42		√		77,98
43		√		79,83
44		√		74,43
45		√		80,11
46		√		83,66
47		√		79,69
48		√		80,26
49		√		76,28
50		√		82,95
51		√		81,39
52		√		79,97
53		√		76,99
54		√		81,96
55		√		80,11
56		√		82,24
57		√		82,95
58		√		79,69
59		√		83,66
60		√		82,67
61			√	63,35
62			√	63,92
63			√	60,23
64			√	50,57
65			√	48,72
66			√	46,88
67			√	58,66
68			√	58,81
69			√	51,42
70			√	51,7
71			√	56,39
72			√	52,41
73			√	54,59
74			√	55,26

BIODATA PENULIS

Penulis dilahirkan di Pati pada 28 November 1991. Penulis pernah menempuh pendidikan formal di MIT Yaummi Fatimah (1998-2004), SMPIT Ittihadul Muwahidin (2004-2007), dan SMA Negeri 1 Pati (2007-2010). Pada tahun 2010 penulis diterima di Jurusan Sistem Informasi ITS.

Penulis sempat mengikuti UKM Taekwondo ITS dan mengikuti beberapa kali pertandingan. Kesibukan lain selain sebagai mahasiswa, penulis juga aktif sebagai asisten mata kuliah sistem operasi mulai tahun 2011 hingga 2014. Pada tahun 2013 penulis tergabung dalam Badan Eksekutif Mahasiswa (BEM) Fakultas Teknologi Informasi (FTIf) sebagai staf divisi pengembangan dan riset. Penulis juga tercatat sebagai asisten laboratorium e-Bisnis jurusan Sistem Informasi mulai tahun 2013 hingga 2015.

Tugas akhir yang dipilih penulis di Jurusan Sistem Informasi ini masuk ke dalam bidang minat E-Business. Penulis dapat dihubungi melalui e-mail afif.hendrawan@gmail.com atau about.me/afifhendrawan