BAB III

ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis

Punch training merupakan bentuk latihan fisik dimana dilakukan gerakan memukul dengan cara mengepalkan tangan lalu menghantamkannya ke sebuah sasaran. *Punch* termasuk gerakan pertahanan yang digunakan untuk memberi jarak antara diri sendiri dengan musuh atau lawan, sehingga terdapat jeda waktu untuk melarikan diri ataupun memberi serangan lanjutan. Gerakan *punch* biasanya dipakai dalam beberapa seni beladiri ataupun olahraga petarung. Salah satu contohnya adalah tinju.

Jab adalah contoh dari gerakan paling dasar dari sebuah *punch*, yaitu dengan mengepalkan tangan dan memukul lurus kedepan sejajar dengan dada sampai tangan benar-benar lurus, jab dapat dilakukan dengan cepat dan memiliki efek yang cukup besar untuk melumpuhkan target sementara. Biasanya jab digunakan untuk membuat pengalih perhatian, bertahan ataupun membuat jarak.

Kekuatan pukulan seorang atlit tinju atau siapapun yang melakukan pukulan dapat dilihat dari kekuatannya saat melakukan jab dimana kekuatan jab ditentukan oleh beberapa faktor diantaranya berat tangan si pemukul dan kecepatan tangan saat melakukan jab. Sehingga semakin cepat gerakan tangan dan semakin berat tangan si pemukul maka akan memiliki jab yang lebih bertenaga daripada yang kecepatan pukulannya pelan tetapi tangannya berat atau sebaliknya.

Sehingga untuk melihat kekuatan jab dapat menggunakan Kinect sebagai sensor penangkap gerakan dan menggunakan rumus fisika untuk menghitung data

yang dihasilkan oleh Kinect. Untuk menghitung berat tangan dari pemukul akan menggunakan rumus segmen tubuh milik de Leva yaitu berat satu tangan manusia adalah 4,715 persen dari berat tubuh keseluruhan. Kemudian untuk mengukur kecepatan tangan saat melakukan jab *punch* akan menggunakan Kinect, dimana Kinect sudah memiliki kemampuan untuk mengukur jarak dengan menggunakan *Depth-Sensor*, *Depth-Sensor* bekerja menggunakan *Infrared-beam* dengan memanfaatkan waktu pantulan dari cahaya *Infrared*, semakin cepat pantulan cahaya *Infrared* ditangkap oleh Kinect maka semakin dekat objeknya dengan Kinect, sehingga untuk menentukan jarak suatu objek menjadi lebih akurat. Terdapat pula *Runtime-Tracking* yang berfungsi untuk mendeteksi posisi tubuh seperti tangan, kepala, kaki, badan, dan lainnya.

Dari hasil perhitungan tersebut nantinya akan dimunculkan sebuah grafik kepada *user* sehingga *user* dapat mengetahui seberapa kuat tenaga jab yang dia lakukan dan untuk mengukur performa kekuatan pukulan saat melakukan *training* akan diberikan sebuah penghitung waktu dan batas pukulan tertentu sebagai standar *training* yang harus dilewati, dimana standar tersebut dapat dimasukkan secara manual.

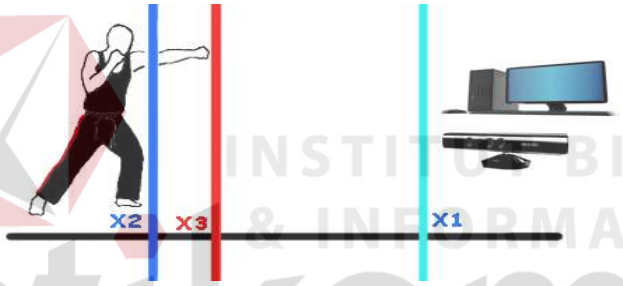
3.1.1 Mengukur Kecepatan Jab

Untuk mengukur kecepatan Jab *user* harus berdiri di depan menghadap Kinect seperti gambar 3.1, dimana X1 adalah jarak 0 cm yang akan ditampilkan oleh Kinect, jarak ini didapat oleh kemampuan *Depth-Sensor* milik Kinect, sedangkan X2 merupakan jarak antara *user* dan Kinect dimana Kinect tidak akan

memulai perhitungan kecepatan apabila *user* tidak berada di area X2, area X2 ini nantinya dapat ditentukan jaraknya didalam program yang akan dibuat.

Gambar 3.1 Mengukur Kecepatan Jab

Setelah *user* berada di area X2 untuk dapat dilakukan perhitungan kecepatan maka *user* harus melakukan gerakan Jab lurus ke arah Kinect seperti gambar dibawah.



Gambar 3.2 Mengukur Kecepatan Jab 2

Kinect yang telah melakukan tracking terhadap tangan akan mencatat perubahan jarak dan waktu tempuh tangan dari X2 ke X3. Sehingga dari waktu tempuh tersebut akan dibagi dengan jarak dari X2 ke X3, maka ditemukanlah sebuah variabel kecepatan v, dimana v ini disimpan kedalam memori aplikasi dan nantinya akan digunakan untuk perhitungan kekuatan pukulan.

3.1.2 Mengukur Kekuatan Pukulan

Untuk pengukuran kekuatan pukulan akan digunakan akan menggunakan perhitungan Energi Kinetik ($Ek = ½\ m.v^2$), yaitu setengah dikalikan massa benda

dikalikan kuadrat kecepatan. Pertama dibutuhkan masukan massa, massa merupakan bobot yang dimiliki suatu benda atau objek, dikarenakan massa yang akan dihitung adalah hanya massa sebuah tangan maka akan digunakan tabel segmentasi tubuh manusia oleh de Leva, dimana sebuah tangan memiliki bobot 4,715 persen dari keseluruhan tubuh manusia. Massa tersebut didapatkan dari masukan massa tubuh *user* kedalam program. Dimana massa tersebut akan menjadi variable masukan M, kemudian masukan M akan dicari massa tangannya (MT) dengan rumus *M x 4,715 / 100*, hasil dari perkalian tersebut akan menghasilkan massa tangan dengan variabel MT, sehingga rumus Energi Kinetik berubah menjadi $Ek = ½ MT.v^2$.

Setelah variabel MT berisi nilai selanjutnya adalah mengkuadratkan variabel kecepatan v yang telah didapat sebelumnya. Hasil dari pengkuadratan v akan dikalikan dengan MT, kemudian hasil perkalian tersebut akan dikalikan dengan setengah maka akan di dapat sebuah persamaan akhir $Ek = ½ MT.V$, nilai tersebut yang akan dijadikan variabel Kekuatan (Ek) di dalam program.

Sedangkan untuk pencatatan performa akan menggunakan sistem nilai tertinggi, dimana nilai tersebut didapat dari nilai variabel Ek yang telah ditemukan sebelumnya, kemudian Ek yang dihasilkan oleh Jab selama latihan akan dicatat tenaga pukulan terkuat dan pukulan terlemah, selanjutnya disimpan sebagai rekor pribadi. Apabila nilai dalam satu sesi lebih tinggi dari sesi sebelumnya maka nilai tertinggi tersebut akan menggantikan nilai yang sebelumnya. Sehingga setiap *user* melakukan *training* dapat meningkatkan performa kekuatan pukulan dengan mengalahkan nilai *training* di sesi yang lalu.

3.1.3 Spesifikasi Perangkat Lunak

Dari hasil analisis diatas maka dibutuhkan sebuah aplikasi yang dapat melakukan hal berikut :

A. Ketika *user* berada dalam jangkauan Kinect maka tampilan dilayar harus menampilkan tubuh *user* secara virtual.

B. Dapat menangkap masukan gerakan *user* dari Kinect. Dengan cara melakukan *tracking* terhadap kedua tangan dan memberi ID pada setiap tangan. Sehingga apabila setiap ID tersebut mengenai *triger* dalam program akan menjalankan aksi sesuai *triger* yang ada, misal memillih menu.

C. Menggunakan masukan tersebut untuk menghitung kecepatan jab dari *user* menggunakan teknik perpindahan objek. Yaitu dengan cara menghitung jarak dari titik awal ke titik akhir pergerakan kemudian dihitung dengan waktu perpindahan.

D. Menghitung kekuatan jab menggunakan cara yang sudah ditentukan dibagian analisis.

E. Menggunakan nilai dari kekuatan jab sebagai parameter penilaian saat melakukan *training*.

F. Setiap hasil *training* akan dibandingkan dengan nilai latihan sebelumnya dan memberikan peringkat setiap sesi *training*.

G. Dapat menyajikan grafik performa dalam satu sesi *training*.

3.1.4 Inisialisasi Kinect

Prosedur inisialisasi digunakan untuk melakukan inisialisasi awal guna mempersiapkan Kinect yang akan dipakai. Terdapat beberapa proses inisialisasi diantaranya :

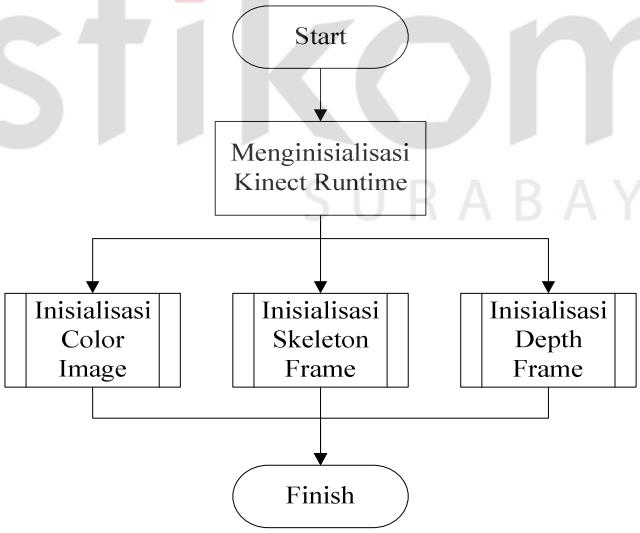
A. Proses inisialisasi *runtime* Kinect yang terdiri atas 3 sub-proses. Dimana proses tersebut berperan untuk mempersiapkan Kinect yang akan digunakan

B. Sub-proses inisialisasi color image berperan untuk menginisialisasi sensor *live feed camera* atau *RGB* kamera pada Kinect.

C. Sub-proses inisialisasi *depth frame* berperan untuk menginisialisasi 3D *depth sensor* pada Kinect.

D. Sub-proses inisialisasi *runtime frame* berperan untuk menginisialisasi *runtime tracking* pada Kinect.

Prosedur inisialisasi Kinect dapat dilihat pada Gambar 3.3 :



Gambar 3.3 *Flowchart* Inisialisasi Kinect

3.1.5 Mendeteksi Objek dan *Drawing User*

Prosedur yang dipakai untuk mendeteksi objek dan menggambar *user* adalah sebagai berikut :

A. Prosedur *RuntimeFrameReady* digunakan untuk mendeteksi *user runtime* dengan cara menggambar titik-titik pada persendian (*joint*) tubuh

A.1 Proses *clear image* berperan dalam melakukan *refresh* pada komponen *canvas* image pada WPF-*form*, berguna agar gambar *runtime* tidak tertumpuk setiap kali *user* melakukan pergerakan.

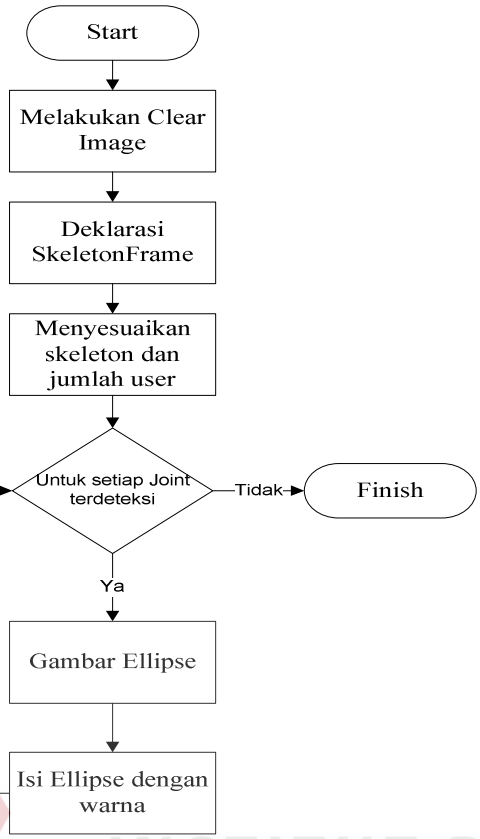
A.2 Proses deklarasi variabel bertipe *RuntimeFrame* berperan untuk menampung *runtime data user*.

A.3 Proses penyesuaian *runtime* dan jumlah maksimal *user* untuk membatasi *runtime* agar tidak mendeteksi *runtime* lain setelah *runtime* pertama terdeteksi. Hal ini bertujuan agar meningkatkan kinerja aplikasi dalam melakukan pendeteksian *user*.

A.4 Proses menggambar *ellipse* digunakan untuk menempelkan gambar *ellipse* dari *form* pada setiap *runtime joint* yang terdeteksi.

A.5 Proses isi *ellipse* dengan warna agar setiap *ellipse* yang dibuat memiliki warna.

Proses yang terjadi pada prosedur *Runtime*FrameReady dapat dilihat pada Gambar 3.4.



Gambar 3.4 *Flowchart Runtime*FrameReady

B. Prosedur *DrawPoint* digunakan untuk menggambar titik-titik pada tubuh *user* dan digunakan pada prosedur *RuntimeFrameReady*.

Beberapa proses yang terjadi pada saat *DrawPoint* antara lain:

B.1 Proses deklarasi variabel *scaledJoint* berperan untuk menandai *joint* pada titik tubuh *user*.

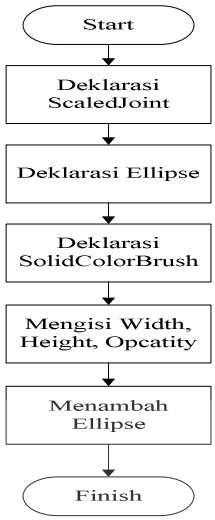
B.2 Proses deklarasi variabel bertipe *ellipse* digunakan untuk membuat *ellipse* yang nantinya akan ditempelkan pada titik *joint* dari *user*.

B.3 Proses deklarasi *SolidColorBrush* digunakan untuk mengisi warna yang nantinya berperan dalam proses pewarnaan *ellipse*.

B.4 Proses mengisi *width, geight* dan *opacity* dari *ellipse* digunakan untuk menentukan ukuran tinggi, lebar serta tingkat transparansi dari *ellipse* yang dibuat.

B.5 Proses menambah *ellipse* berperan dalam membuat *ellipse* dengan atribut parameter yang diatur sebelumnya.

Proses yang terjadi pada prosedur DrawPoint dapat dilihat pada Gambar 3.5.



Gambar 3.5 *Flowchart DrawPoint*

Flowchart pseudocode berikut digunakan untuk mendeteksi kecepatan pukulan dari *user* dan mendeteksi adanya hantaman dengan objek virtual yang dibuat.

Penjelasan proses yang terjadi pada pendeteksian kecepatan dan pendeteksian hantaman dengan objek virtual adalah sebagai berikut :

A. Proses Pendeteksian User

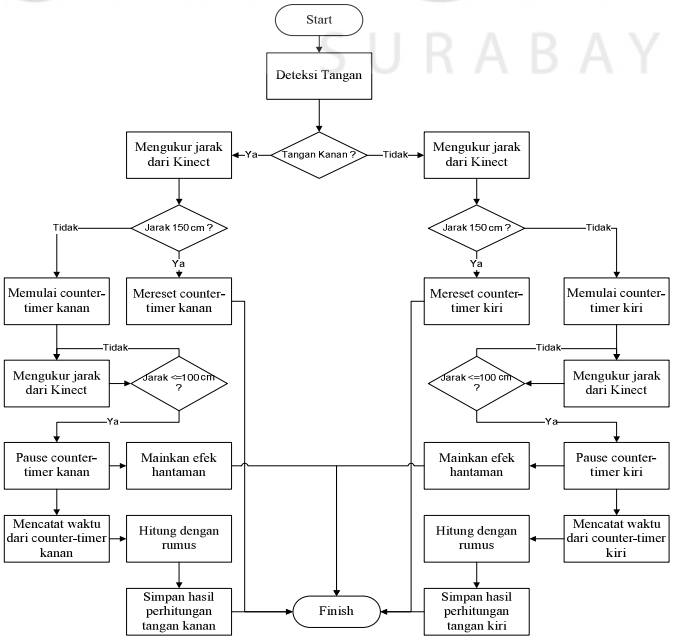
- User berdiri tegak didepan Kinect pada jarak antara 150 - 155 cm.
- User melambaikan tangan kearah Kinect agar Kinect dapat memberikan ID pada masing-masing tangan.

B. Proses Pendeteksian Tangan

- Tangan harus berada di area tangkapan Kinect, setelah tangan terdeteksi maka diberikan ID pada masing-masing tangan kemudian diberikan sebuah kursor berupa sarung tangan.

C. Proses Pendeteksian Kecepatan Pukulan

- Tangan yang sudah terdeteksi akan diukur jaraknya.
- Setiap tangan memiliki ID dan counter-timer masing-masing.
- Jika tangan terdeteksi pada jarak 150 cm maka counter-timer akan diatur ke posisi 00.00.00.000.
- Jika tangan bergerak maju mendekati Kinect sehingga jaraknya kurang dari 150 cm atau sama dengan 149 cm maka counter-timer akan di Mulai.
- Apabila tangan sudah dekat dengan Kinect dengan jarak tangan kurang dari 100 cm atau sama dengan 100 cm maka counter-timer akan dihentikan, mainkan efek hantaman lalu dicatat waktu yang dihasilkan oleh counter-timer tersebut.
- Hitung waktu yang didapat dari counter-timer dengan cara yang telah ditentukan dibagian analisis, kemudian catat hasilnya. Hasil tersebut merupakan kecepatan tangan. Masing-masing nilai dibedakan sesuai ID tangan tersebut.



Gambar 3.6 Flowchart pendeteksian kecepatan dan hantaman

3.1.6 Menampilkan *Video Stream* dan *Depth Stream*

Prosedur yang dipakai untuk menampilkan *video stream* dan *depth stream* pada layar adalah sebagai berikut:

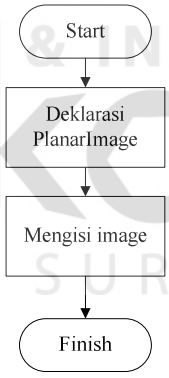
A. Prosedur *ColorImageReady* digunakan untuk menampilkan *video stream* yang ditangkap Kinect ke layar.

Proses yang terjadi pada prosedur *ColorImageReady* antara lain :

A.1 Proses deklarasi variabel bertipe *planarImage* berperan menampung *video output* dari kamera *RGB* Kinect.

A.2 Proses mengisi komponen *image* pada WPF-*form* berperan mengisi *image* dengan *video output* dari kamera *RGB*.

Proses yang terjadi pada prosedur *ColorImageReady* ini dapat dilihat pada Gambar 3.7



Gambar 3.7 *Flowchart ColorImageReady*

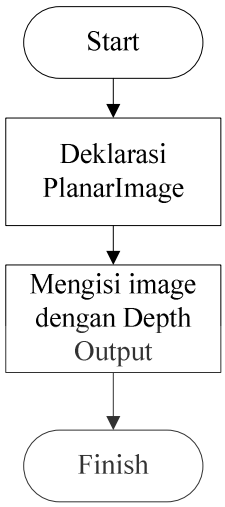
B. Prosedur *DepthImageReady* digunakan untuk menampilkan *depth stream* yang ditangkap Kinect ke layar.

Proses yang terjadi pada prosedur *DepthImageReady* antara lain :

B.1 Proses deklarasi variabel bertipe *planarImage* berperan untuk menampung *depth output* dari *depth* sensor Kinect.

B.2 Proses mengisi komponen *image* pada WPF-*form* berperan mengisi *image* dengan *depth output* dari *depth* sensor.

Proses yang terjadi pada prosedur *DepthImageReady* ini dapat dilihat pada Gambar 3.8.



Gambar 3.8 *Flowchart DepthImageReady*

3.1.7 Melakukan Pengaturan *User Joint*

Prosedur pengaturan *user joint* digunakan supaya *joint* yang dideteksi sesuai dengan posisi dan ukuran *user*.

A. Prosedur *ScaleJoint* digunakan untuk mengukur *joint* pada *user*.

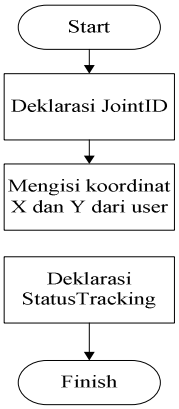
Proses yang terjadi pada prosedur *ScaleJoint* adalah sebagai berikut :

A.1 Proses deklarasi *JointID* yang berfungsi menampung ID dari *joint* yang dideteksi.

A.2 Proses mengisi koordinat X dan Y dengan posisi *user* berperan untuk menentukan letak koordinat dan posisi *user*.

A.3 Deklarasi *StatusTracking* berperan untuk menandai apakah *user* telah terdeteksi atau belum.

Proses dari prosedur *ScaleJoint* ini dapat dilihat pada Gambar 3.9.



Gambar 3.9 Flowchart Scale*Joint*

B. Prosedur *ScaledPosition* digunakan untuk mengukur posisi *user* dari layar.

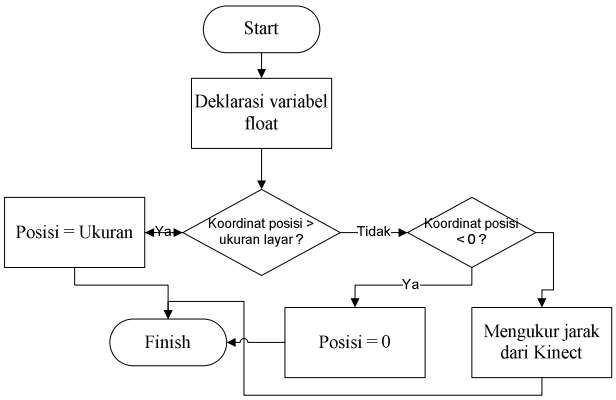
Proses yang terjadi pada prosedur *ScaledPosition* adalah sebagai berikut :

B.1 Proses deklarasi variabel bertipe *float* yang berperan untuk menampung titik koordinat dari posisi *user*.

B.2 Jika nilai posisi *user* lebih besar dari ukuran *canvas* maka posisi dianggap sama dengan ukuran canvas.

B.3 Jika nilai posisi *user* kurang dari nol maka posisi dianggap sama dengan nol. Selain itu nilai *float* posisi *user* dianggap sama dengan posisi *user* saat itu.

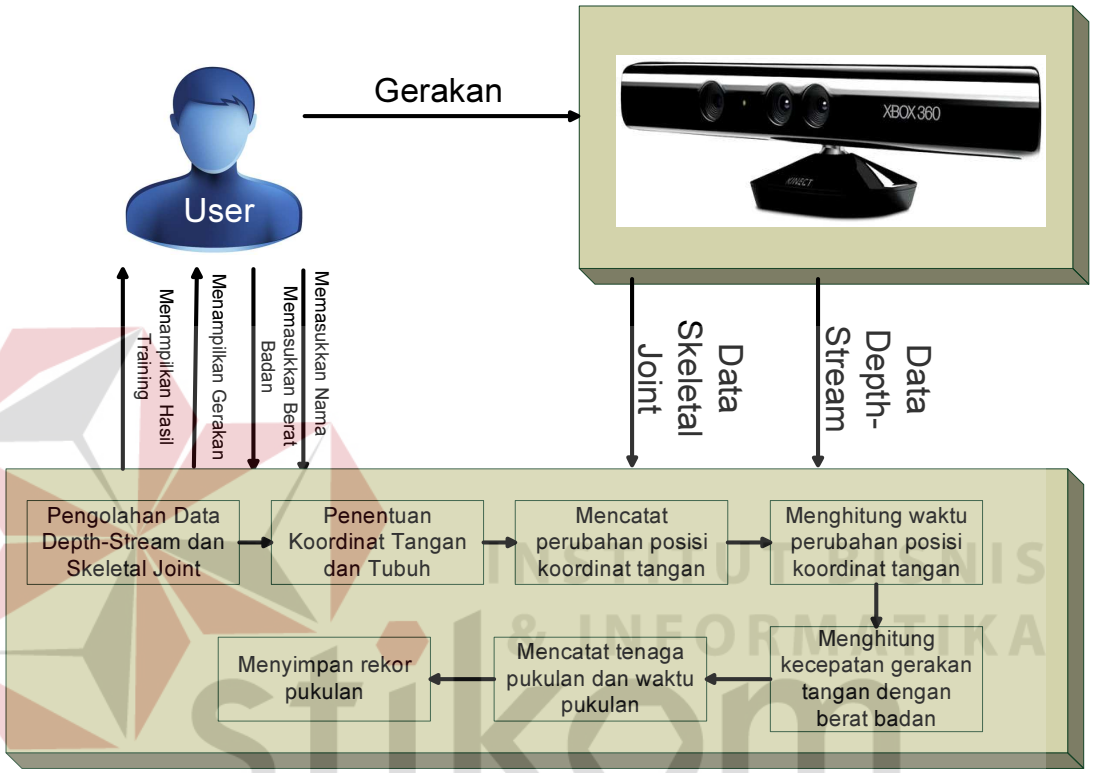
Proses pada prosedur *ScaledPosition* dapat dilihat pada Gambar 3.10.



Gambar 3.10 *Flowchart ScaledPosition*

3.2 Model Pengembangan Sistem

Berdasarkan hasil analisis tersebut, maka akan dibuat sebuah pengembangan aplikasi, dimana proses yang terjadi dalam aplikasi *Virtual Punch Training* dapat digambarkan sebagai berikut:



Gambar 3.11 Blok Diagram Aplikasi

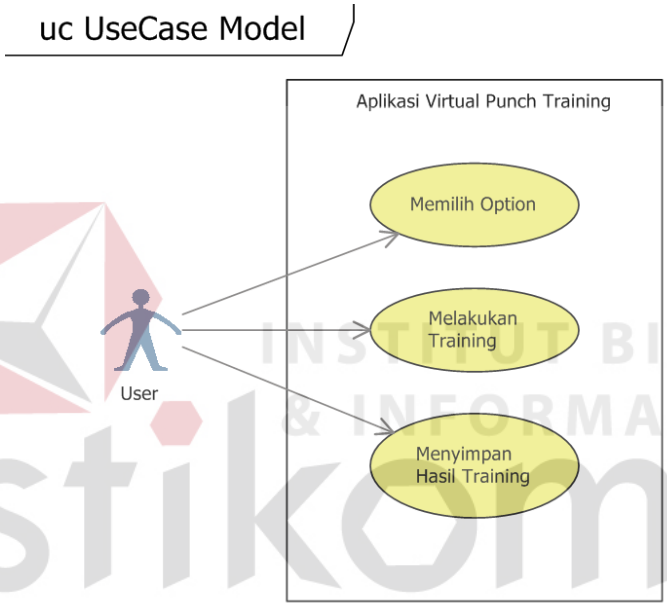
Berikut adalah cara kerja dari aplikasi *Virtual Punch Training* menggunakan Microsoft Xbox Kinect:

1. *User* melakukan interaksi terhadap aplikasi dengan menggunakan gerakan tangan didepan Kinect, dan memilih menu yang tersedia. *User* juga dapat berinteraksi secara manual dengan mouse dan keyboard.
2. Gerakan interaksi *User* ditangkap oleh Kinect kemudian data *Depth-Stream* dan data *Skeletal-Joint* dimasukkan kedalam komputer untuk diproses oleh aplikasi.

3. Aplikasi mengolah gerakan interaksi terhadap menu dari *user* dan memberikan respon berupa tampilan dari layar, baik berupa tampilan hasil latihan ataupun tampilan menu latihan.

3.3 *Use Case Diagram* Aplikasi *Virtual Punch Training*

Kemampuan sistem untuk berinteraksi dengan *user* digambarkan dengan *use case diagram* pada gambar dibawah:



Gambar 3.12 *Use Case Diagram* Aplikasi *Virtual Punch Training*

3.4 Activity Diagram

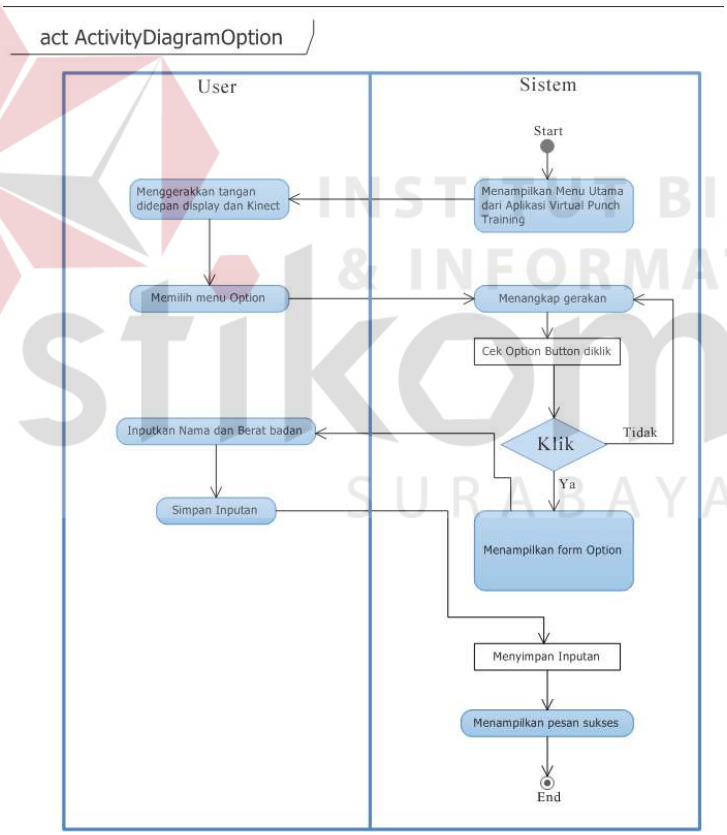
Dari *use case* yang ada dibutuhkan *Activity diagram* untuk menjelaskan proses apa saja yang dilakukan setiap *use case.*

3.4.1 *Activity Diagram* untuk *Use Case* Memilih *Options*

Activity memilih *options* dapat digambarkan sebagai berikut. Pertama-tama sistem akan menampilkan menu utama dari aplikasi *Virtual Punch Training* yang

terdiri dari beberapa pilihan, lalu *user* diwajibkan memilih menu *options* terlebih dahulu dimana menu ini berisi data penting berupa masukan berat badan *user* yang akan digunakan untuk menampilkan hasil latihan. *User* dapat menggerakkan tangannya kearah tombol *options* ataupun menggunakan mouse untuk berinteraksi, setelah tombol dipilih maka *form options* akan ditampilkan, *user* diwajibkan mengisi nama dan berat badan menggunakan keyboard dan kemudian menyimpan data tersebut. Setelah selesai *user* dapat kembali ke menu utama.

Activity Diagram dari *use case* Memilih *Options* dapat digambarkan sebagai berikut:

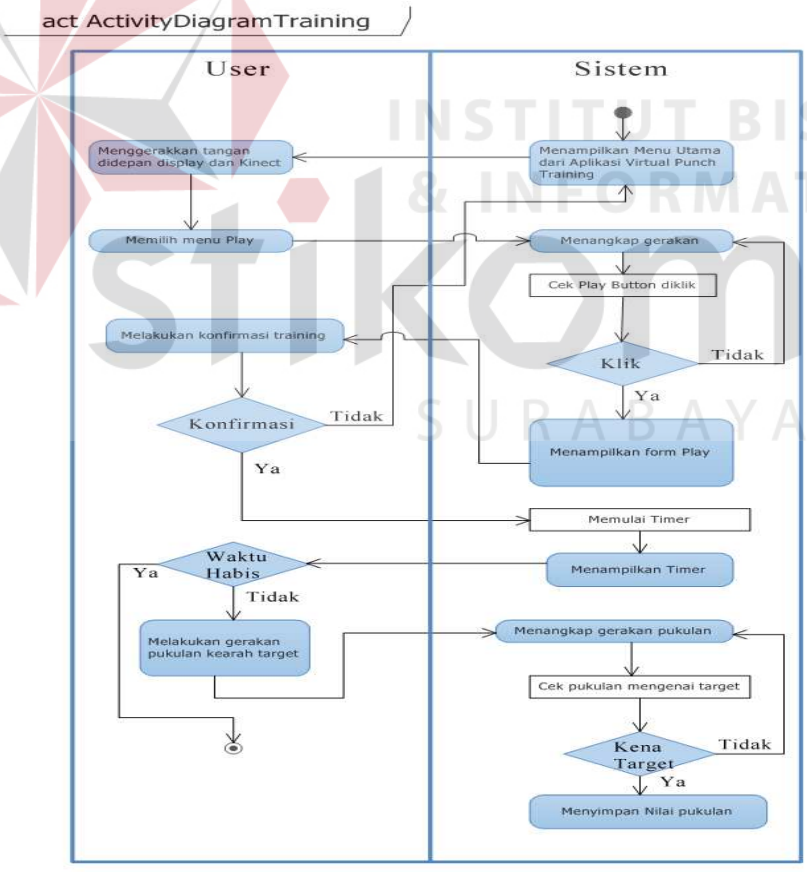


Gambar 3.13 *Activity Diagram* untuk *Use Case* Memilih *Options*

3.4.2 *Activity Diagram* untuk *Use Case* Melakukan *Training*

Activity melakukan *training* dapat digambarkan sebagai berikut. Pertama-tama sistem akan menampilkan menu utama dari aplikasi *Virtual Punch Training*, kemudian *user* memilih menu *play*, saat menu terpilih maka *form play* akan ditampilkan dan memberikan konfirmasi untuk melakukan *training* apabila *user* setuju untuk melakukan *training* maka *timer* akan berjalan dan *user* dapat mulai melakukan gerakan pukulan kearah target sampai waktu habis. Apabila *user* tidak setuju maka akan kembali ke menu utama.

Activity Diagram dari *use case* Melakukan *Training* dapat digambarkan sebagai berikut:

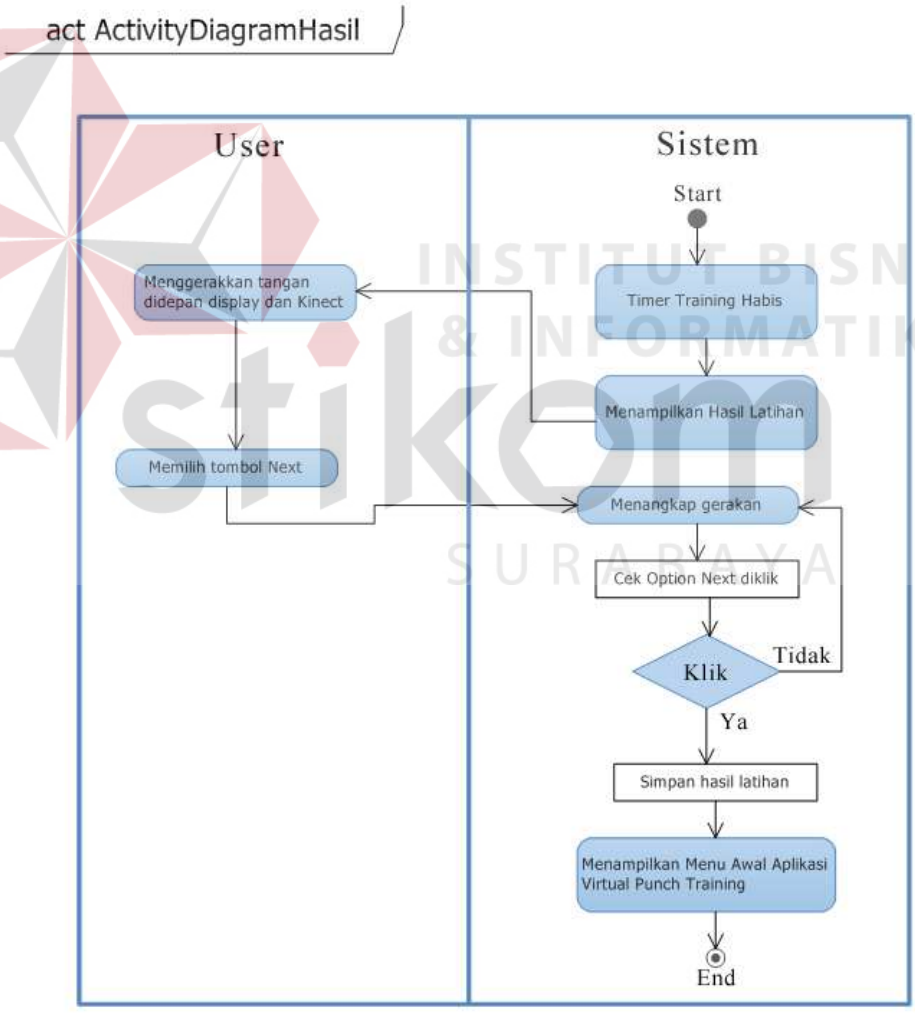


Gambar 3.14 *Activity Diagram* untuk *Use Case* Melakukan *Training*

3.4.3 *Activity Diagram* untuk *Use Case* Menyimpan Hasil *Training*

Activity menyimpan hasil *training* dapat digambarkan sebagai berikut. Setelah *user* selesai melakukan *training* maka akan segera muncul *form* hasil latihan, disini *user* dapat mereview setiap pukulan yang dihasilkan selama *timer* berjalan. *User* diwajibkan memilih tombol *next* untuk kembali ke menu awal dan hasil *training* disimpan dalam bentuk nilai yang terbaik saat melakukan *training*.

Activity Diagram dari *use case* Melakukan *Training* dapat digambarkan sebagai berikut:



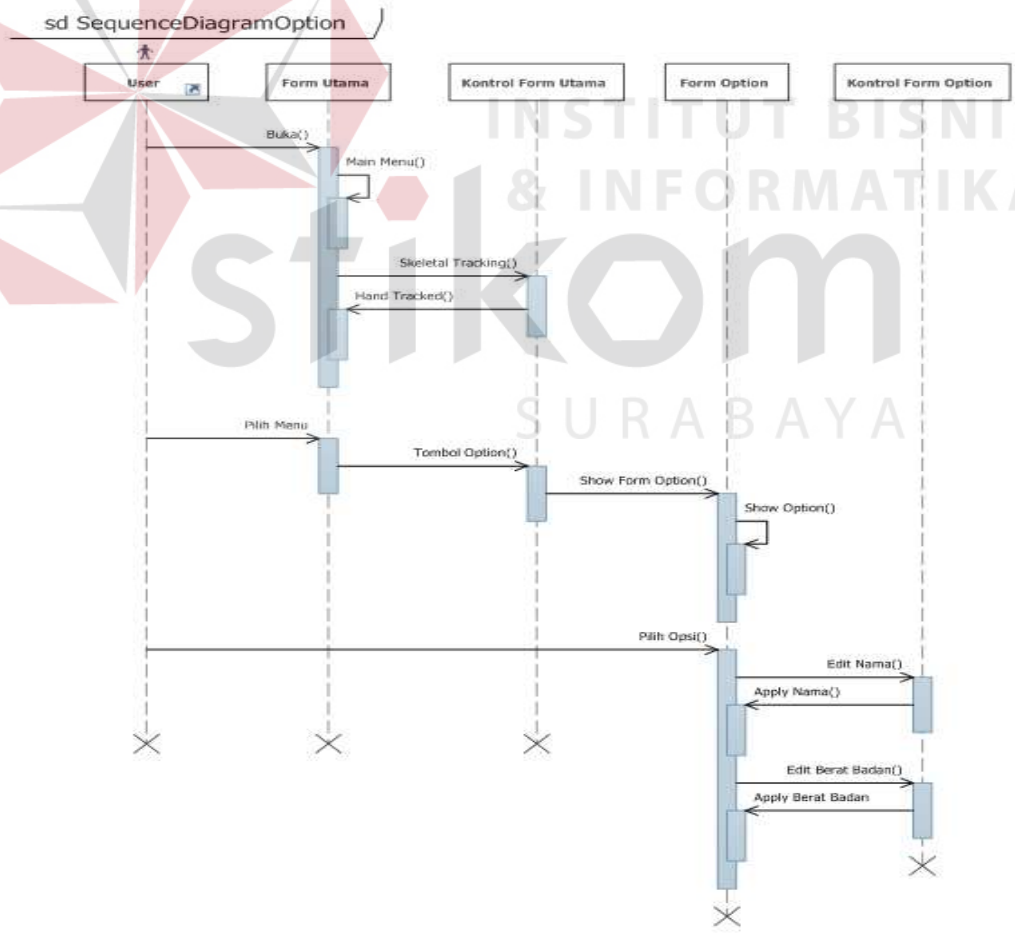
Gambar 3.15 *Activity Diagram* untuk *Use Case* Menyimpan Hasil *Training*

3.5 Sequence Diagram

Untuk memberikan penjelasan dari masing-masing *use case* dari aplikasi *Virtual Punch Training* dibutuhkan *sequence diagram* yang menggambarkan jalannya suatu proses yang melibatkan *obejct* atau *instance* dari suatu *class* pada aplikasi *Virtual Punch Training*. Untuk lebih detail akan dijelaskan sebagai berikut:

3.5.1 *Sequence Diagram* Memilih *Options*

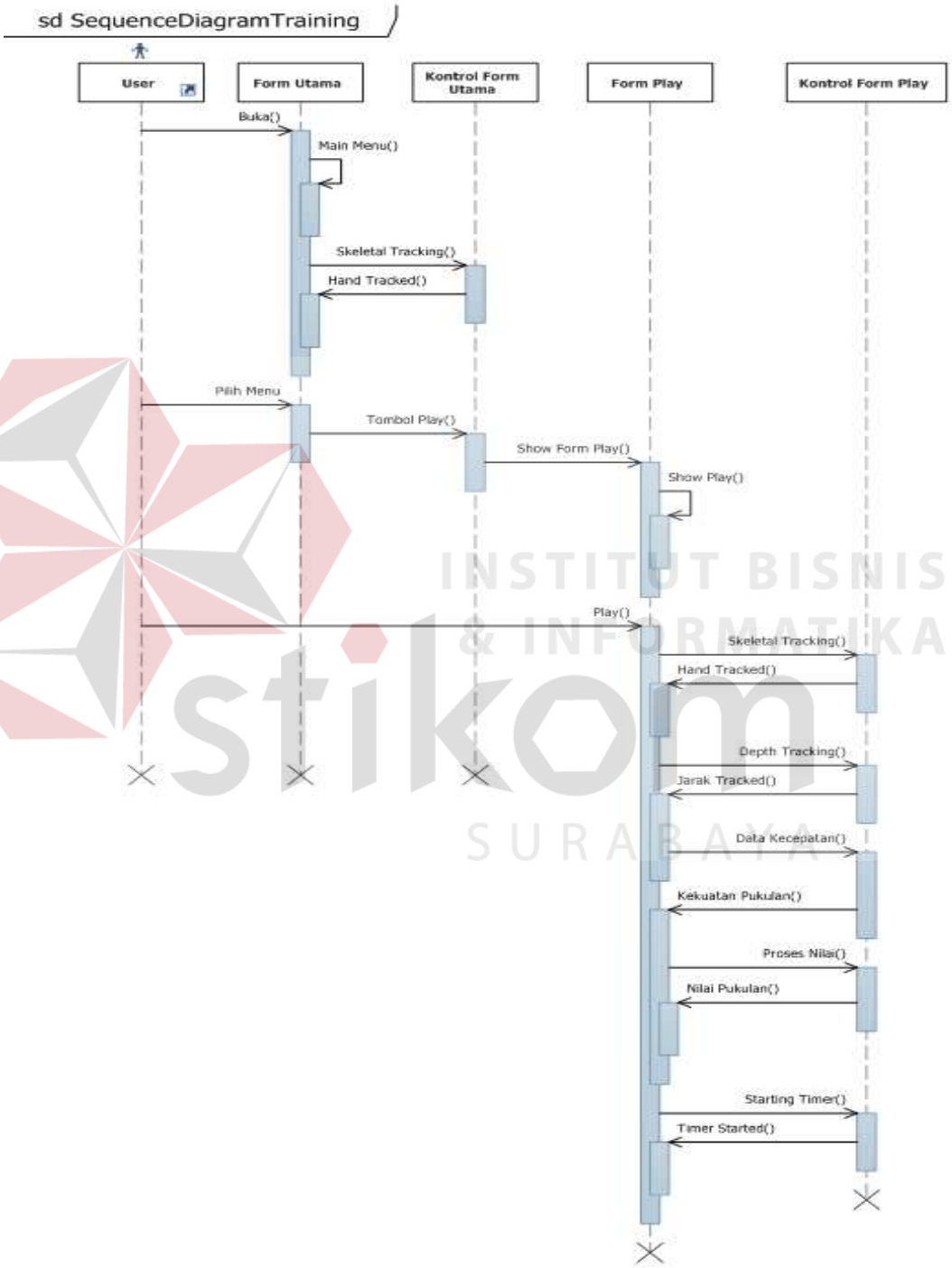
Sequence Diagram dari *Activity* Memilih *Options* digambarkan sebagai berikut:



Gambar 3.16 *Sequence Diagram* Memilih *Options*

3.5.2 *Sequence Diagram* Melakukan *Training*

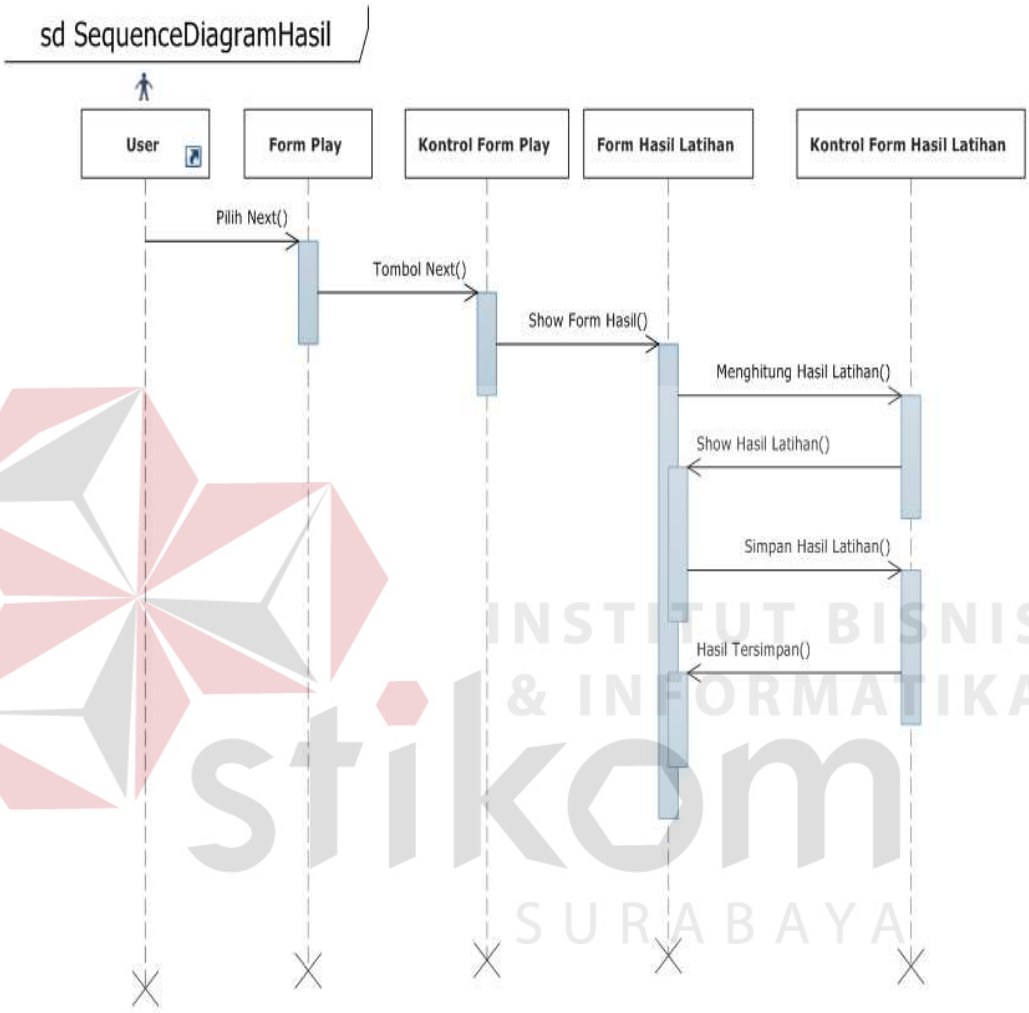
Sequence Diagram dari *Activity* Melakukan *Training* digambarkan sebagai berikut:



Gambar 3.17 *Sequence Diagram* Melakukan *Training*

3.5.3 *Sequence Diagram* Menyimpan Hasil *Training*

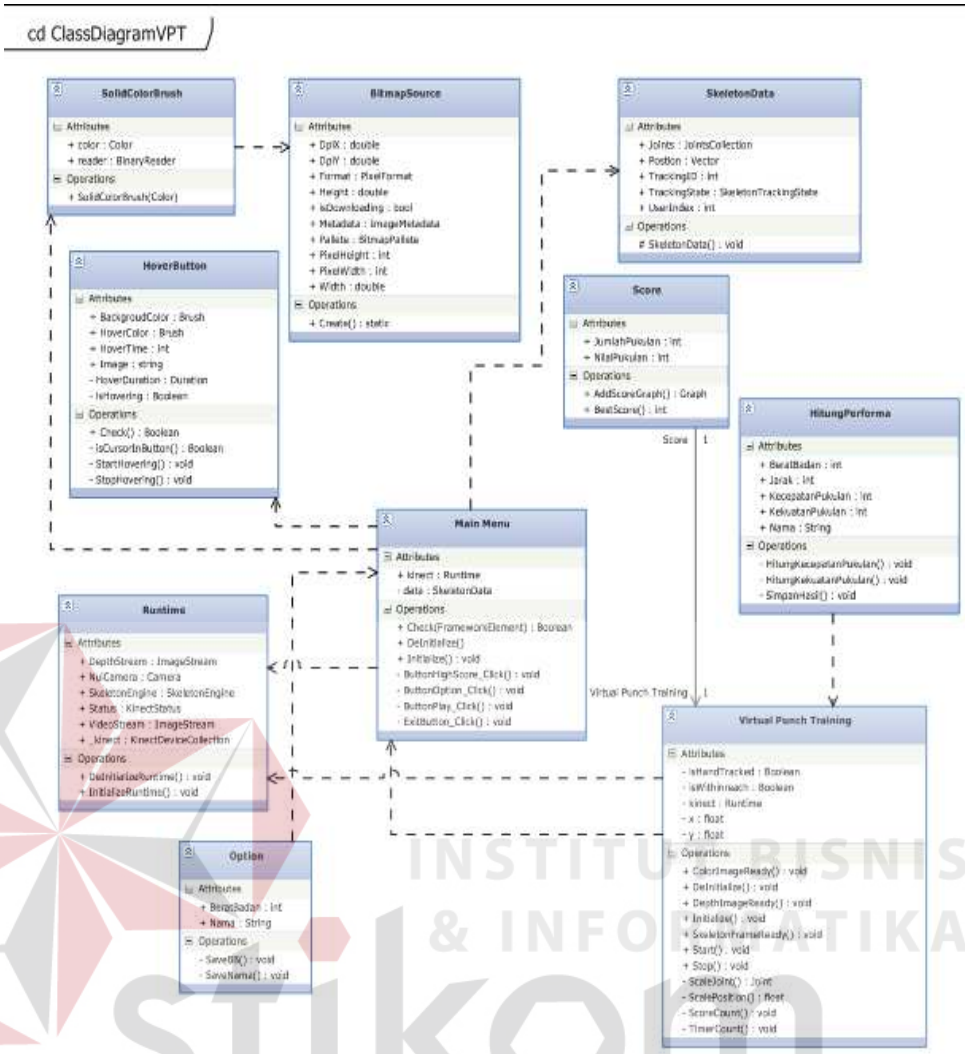
Sequence Diagram dari *Activity* Menyimpan Hasil *Training* digambarkan sebagai berikut:



Gambar 3.18 *Sequence Diagram* Menyimpan Hasil *Training*

3.6 Class Diagram

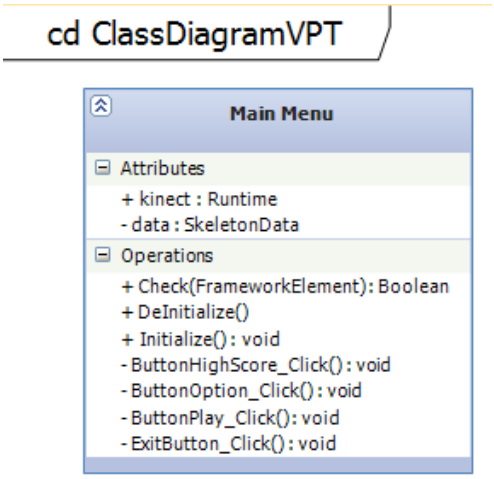
Berdasarkan perencanaan sistem *use case diagram*, dibutuhkan *class* untuk mendukung jalannya aplikasi. *Class Diagram* digunakan untuk menampilkan *class* atau paket yang ada didalam sistem serta relasi antar *class* tersebut. Hubungan antar *class* dapat digambarkan sebagai berikut:



Gambar 3.19 *Class Diagram* Aplikasi *Virtual Punch Training*

3.6.1 Class Diagram Main Menu

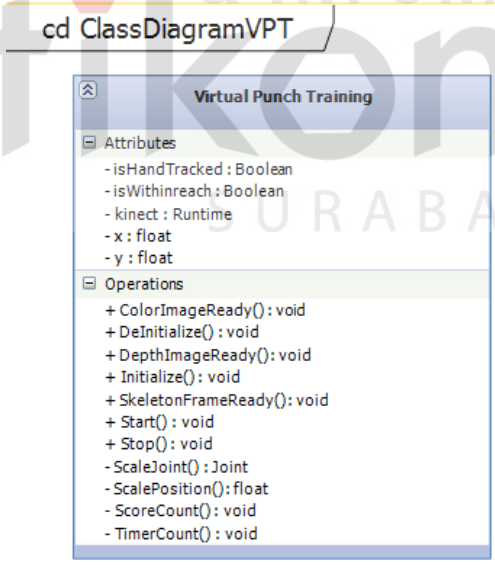
Class Main Menu merupakan class menu utama pada aplikasi *Virtual Punch Training*. Class ini mengkoordinasikan operasi dari *HoverButton*, tampilan awal aplikasi dan lainnya. Untuk lebih jelasnya lihat gambar berikut:



Gambar 3.20 *Class Diagram Main Menu*

3.6.2 *Class Diagram Virtual Punch Training*

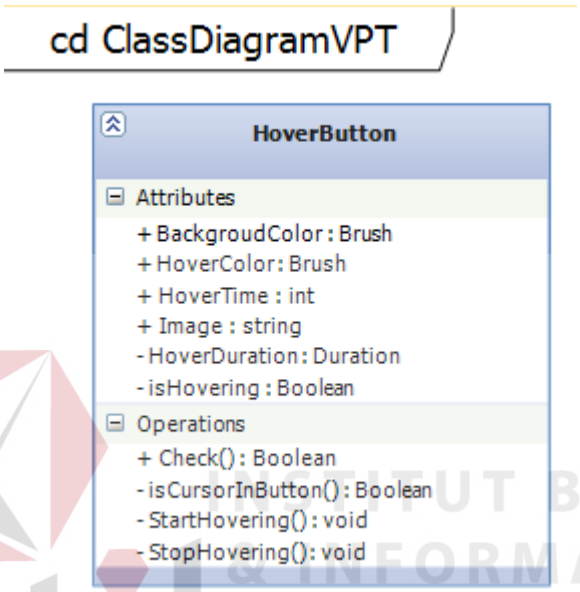
Class Virtual Punch Training merupakan *class* aplikasi utama yang mengkoordinasikan beberapa operasi. Untuk lebih detail dapat dilihat pada gambar berikut:



Gambar 3.21 *Class Diagram Virtual Punch Training*

3.6.3 Class Diagram HoverButton

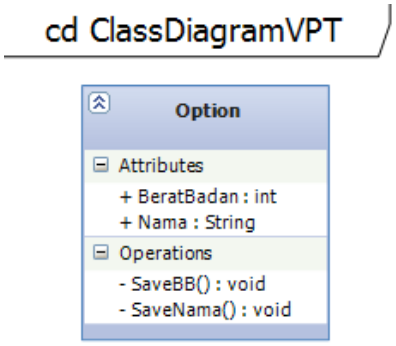
Class HoverButton digunakan untuk melakukan pengecekan apakah tangan *user* sudah menyentuh *Hover Button* pada menu utama, *class* ini juga digunakan untuk menampilkan animasi pada saat *user* melakukan *hovering* pada menu utama, serta mengatur durasi *hover*. Untuk lebih jelasnya lihat gambar berikut:



Gambar 3.22 *Class Diagram HoverButton*

3.6.4 Class Diagram Options

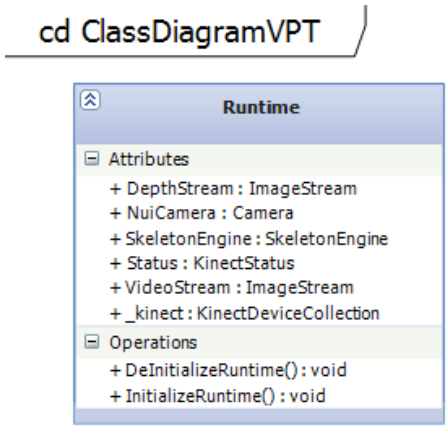
Class ini digunakan untuk menyimpan data nama *user* dan berat badan *user*. Untuk lebih jelasnya lihat gambar berikut:



Gambar 3.23 *Class Diagram Options*

3.6.5 *Class Diagram Runtime*

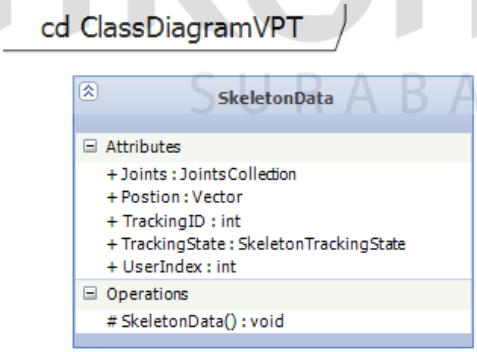
Class Runtime digunakan untuk melakukan inisialisasi awal Kinect. Untuk lebih jelasnya lihat gambar berikut:



Gambar 3.24 *Class Diagram Runtime*

3.6.6 *Class Diagram RuntimeData*

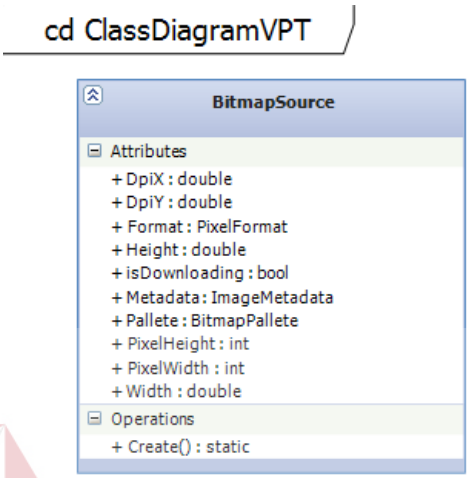
Class ini digunakan untuk menangkap data *runtime user*. Untuk lebih jelasnya lihat gambar berikut:



Gambar 3.25 *Class Diagram RuntimeData*

3.6.7 Class Diagram BitmapSource

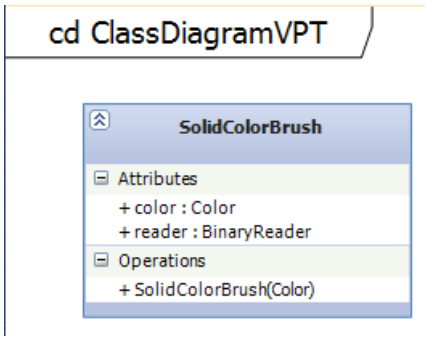
Class ini digunakan untuk menampilkan bitmap image yang digunakan pada pengambilan gambar pada kamera Kinect. Untuk lebih jelasnya lihat gambar berikut:



Gambar 3.26 *Class Diagram BitmapSource*

3.6.8 Class Diagram SolidColorBrush

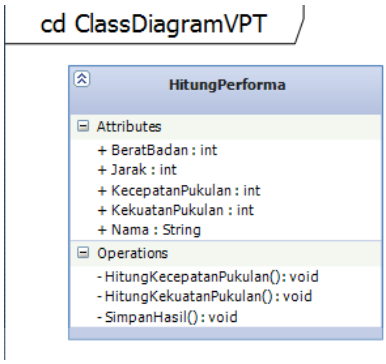
Class ini digunakan untuk mengubah warna pada suatu objek. Untuk lebih jelasnya lihat gambar berikut:



Gambar 3.27 *Class Diagram SolidColorBrush*

3.6.9 *Class Diagram* Hitung Performa

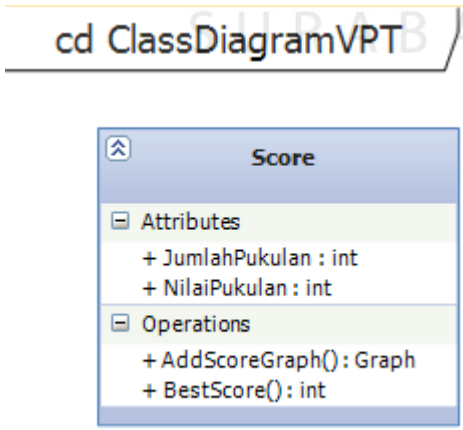
Class ini digunakan untuk menghitung kecepatan pukulan *user*, menghitung kekuatan pukulan dan menyimpan hasilnya. Untuk lebih jelasnya lihat gambar berikut:



Gambar 3.28 *Class Diagram* Hitung Performa

3.6.10 Class Diagram Score

Class ini digunakan untuk menginputkan data nilai pukulan dan jumlah pukulan kedalam grafik kemudian membuat score terbaik. Untuk lebih jelasnya lihat gambar berikut:



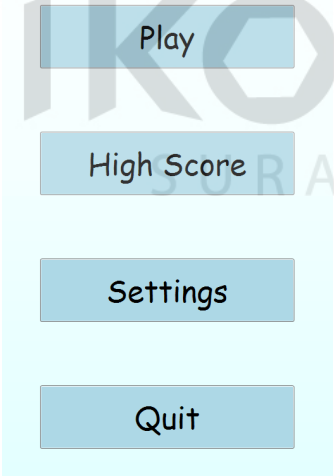
Gambar 3.29 *Class Diagram Score*

3.7 Desain *User Interface*

Pembuatan tampilan sangat diperlukan agar pengguna dapat berinteraksi dengan sistem sehingga dibutuhkan perancangan secara detail mengenai tampilan aplikasi berdasarkan informasi yang ditampilkan pada *display*.

3.7.1 Desain *User Interface Form* Menu Utama

Pilihan yang ada pada menu utama terdiri dari *Play, High Score, Settings, Quit.* Menu *Play* digunakan untuk memanggil *form Training* dimana *user* dapat memulai sesi *training*nya. Menu *High Score* digunakan agar *user* dapat melihat *track-record* dari hasil *training* terakhir, melihat berapa *score* terbaiknya saat melakukan *training*. Menu *Settings* berfungsi untuk menginputkan nama dan berat badan *user*. Menut *Quit* digunakan untuk keluar dari aplikasi. Perancangan tampilan untuk menu utama dapat dilihat pada gambar berikut.



Gambar 3.30 Desain *User Interface Form* Menu Utama

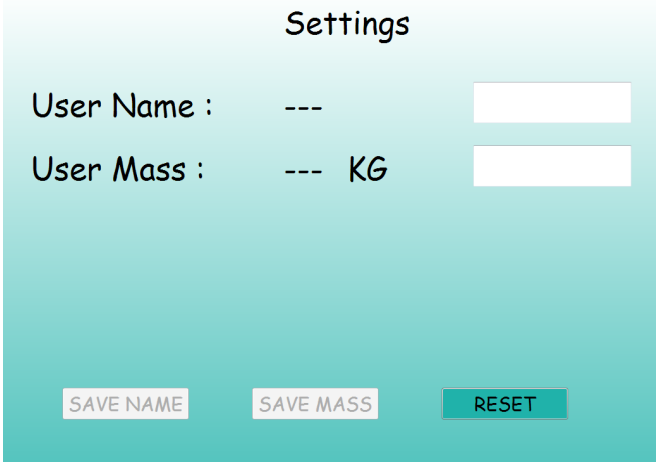
3.7.2 Desain *User Interface Form Training*

Form Training adalah *form* utama dari program *Virtual Punch Training*, dimana *user* melakukan sesi *training* dan berinteraksi dengan objek yang ada didalam sistem. *Timer* akan berjalan beberapa detik setelah *Form Training* dibuka dari Menu Utama. Berikut adalah rancangan tampilan dari *from training*.

Gambar 3.31 Desain *User Interface Form Training*

3.7.3 Desain *User Interface Form Options*

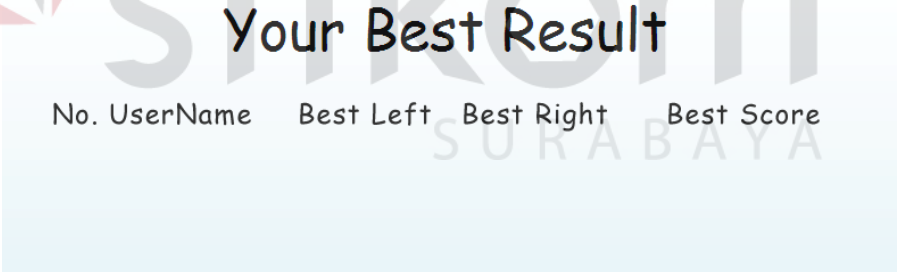
Form Options adalah *form* yang berfungsi untuk mengubah data nama *user* dan data berat badan *user* yang dibutuhkan untuk perhitungan aplikasi dan pencatatan *score* selama latihan. Terdapat beberapa tombol diantaranya *Save Name* berfungsi untuk menyimpan nama *user*. *Save Mass* berfungsi untuk menyimpan berat badan *user*. *Reset* digunakan untuk mengembalikan ke nilai *default* seperti yang ada digambar Berikut rancangan tampilan *form* option.



Gambar 3.32 Desain *User Interface Form Options*

3.7.4 Desain *User Interface Form High Score*

Pada *Form* High Score akan ditampilkan nilai terbaik saat melakukan *training*, terdapat beberapa informasi yang akan ditampilkan dalam *form* ini diantaranya nomor urut, nama *user*, pukulan terbaik dari tangan kiri dan kanan, lalu jumlah score selama latihan. Berikut rancangan tampilan *form high score*.

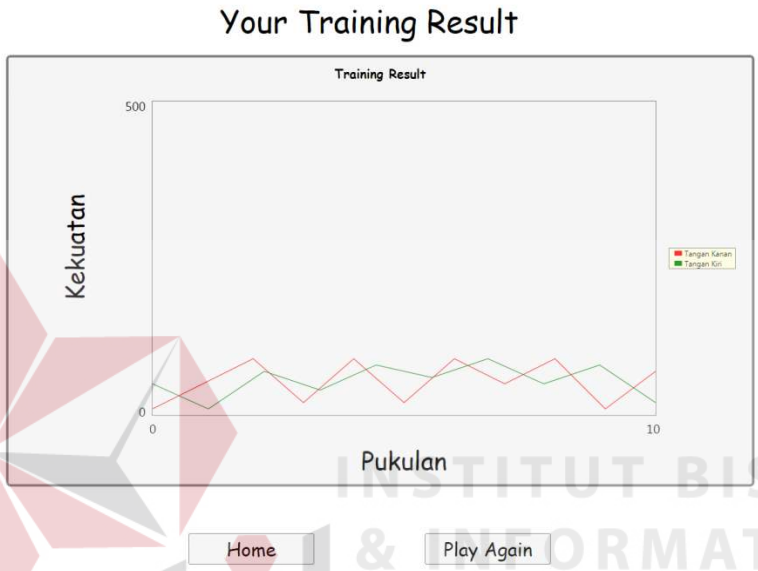


Gambar 3.33 Desain *User Interface Form High Score*

3.7.5 Desain *User Interface Form Preview Result*

Form Preview Result berisi tampilan grafik dari satu sesi *training*, *user* dapat dengan mudah melihat hasil *training*nya. Terdapat 2 tombol didalam *form* ini, diantaranya *Play Again* agar *user* bisa melanjutkan ke sesi *training*

selanjutnya dan *Home* dimana *user* akan kembali ke menu utama. Kekuatan adalah nilai dari kekuatan pukulan dan akan ditampilkan dalam 2 grafis yang berbeda untuk membedakan tangan kanan dan tangan kiri. Pukulan merupakan jumlah pukulan yang dilakukan selama satu sesi *training*. Berikut rancangan tampilan dari *form preview result*.



Gambar 3.34 Desain *User Interface Form Preview Result*