repository.ub.ac.id

PENGGABUNGAN *NATURAL USER INTERFACE* DENGAN *JOYPAD* PADA *ROLE PLAYING VIDEO GAME (RPG)*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Sandhi Wistara
NIM: 125150200111054

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

UNIVERSITAS BRAWIJAYA

PENGESAHAN

PENGGABUNGAN *NATURAL USER INTERFACE* DENGAN *JOYPAD* PADA *ROLE PLAYING VIDEO GAME*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Sandhi Wistara
NIM:125150200111054

Skripsi ini telah diuji dan dinyatakan lulus pada
25 Juli 2019
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing 2
Wibisono Sukmo Wardhono, S.T, M.T	Muhammad Aminul Akbar , S.Kom.,M.T
NIK: 201008 820404 1 001	NIK: 2016078910131001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

repository.ub.ac.id

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar referensi.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 10 July 2019

Sandhi Wistara

NIM: 125150200111054

UNIVERSITAS BRAWIJAYA

PRAKATA

Puji syukur kami panjatkan kepada Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya, sehingga penelitian yang berjudul "Penggabungan *Natural User Interface* dengan *Joypad* pada *Role Playing Game (RPG)*" dapat diselesaikan dengan baik.

Dalam penyusunan dan penelitian skripsi ini tidak lepas dari bantuan dan dukungan baik berupa yang diberikan dari berbagai pihak, sehingga penulis mengucapkan banyak terima kasih kepada:

1. Wibisono Sukmo Wardhono, S.T, M.T selaku dosen pembimbing 1 yang telah rela meluangkan waktu untuk memberikan bimbingan, masukan dan saran untuk penelitian ini.
2. Muhammad Aminul Akbar , S.Kom.,M.T selaku dosen pembimbing 2 yang telah rela meluangkan waktu untuk memberikan bimbingan, masukan dan saran untuk penelitian ini.
3. Seluruh dosen yang telah memberikan ilmu pengetahuan dan motivasi kepada penulis dalam perkuliahan.
4. Kedua orang tua, Kakak dan keluarga besar penulis atas segala dukungan, kasih sayang, dan doanya demi terselesaikannya skripsi ini.
5. Teman-teman seperjuangan dalam pengerjaan skripsi khususnya Viqi Hanada, Aldino Caturrahmanto, dan Muh. Ihsan As Sauri yang senantiasa saling berbagi ilmu dan pengalaman dalam pengerjaan skripsi ini.
6. Serta berbagai pihak yang tidak dapat disebutkan satu-persatu disini atas seluruh bantuan dan ilmu yang telah diberikan.

Penulis berharap seluruh jasa dan kebaikan mereka diberikan balasan dari Tuhan Yang Maha Esa. Penulis menyadari bahwa penelitian ini masih jauh dari kesempurnaan, sehingga penulis mengharapkan kritik dan saran untuk penelitian yang selanjutnya sebagai pengembangan dari skripsi ini. Semoga penelitian ini bermanfaat bagi pembaca

Malang, 10 Juli 2019

Penulis

sandhi_w@student.ub.ac.id

repository.ub.ac.id

ABSTRAK

Role Playing Video Game adalah *game* dimana pengguna diberikan peran dalam dunia yang dibangun. Pada *game* tipe ini menyediakan pemain dengan berbagai macam kekuatan yang terkadang memakan *input* lebih. *Game* tipe ini lebih banyak dikenal pada komputer. Meskipun beberapa *game* jenis ini dikembangkan pada perangkat konsol tetapi pengembang mengurangi beberapa fungsi yang ada. Hal ini dikarenakan *input* yang dimiliki keyboard komputer lebih banyak daripada Joypad yang biasa digunakan pada konsol. Untuk menangani kekurangan ini pada perangkat konsol atau seseorang yang lebih terbiasa menggunakan Joypad maka digabungkan sebuah *input* tambahan yang dapat digunakan bersamaan dengan joypad. Pada konsol jenis X-box memiliki perangkat bernama *kinect*. *Kinect* dikembangkan untuk menangkap gerakan pada pengguna. Perangkat ini adalah satu dari berbagai macam jenis *Natural User Interface*. *Natural user interface (NUI)* adalah sebuah salah satu jenis tatap muka pengguna yang dimanipulasi secara alami. Contoh dari NUI adalah layar sentuh, mouse, dan manipulasi dimana pengguna menggunakannya tanpa perlu latihan khusus. Penelitian konsep Penggabungan *Natural User Interface* dengan *Joypad* pada *Role Playing Video Game (RPG)* dimulai dengan meneliti bagaimana konsep menerapkan *Natural User Interface* pada *kinect*. Konsep dikembangkan dengan meneliti berbagai jenis data yang *kinect* berikan dan bagaimana data diubah menjadi *input*. *Kinect* membaca posisi dari setiap anggota badan, bentuk tangan yaitu lasso, close, open dan unknown. Dari posisi tersebut *kinect* membuat bentuk rangka dari pengguna yang saat itu memasuki kamera dari *kinect*. *Input* berhasil dikembangkan menjadi delapan *input*. Hal yang digunakan adalah jarak antar tangan untuk mengatur antara kondisi *input* atau tidak. Kondisi tangan lasso digunakan sebagai pusat dari *input*. Dari lasso dihitung jarak X dan Y untuk menentukan *input* mana yang dieksekusi.

Kata kunci: Natural User Interface, *Role Playing Video Game*, *kinect*, *Input*.

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

ABSTRACT

Role Playing Video Game is type of *game* where player given role in world that build based on setting that made.In this type of *game* player given many power that use up so many *input* to execute. This type of *game* is more common in computer device then consol. Even thou this type of *game* is sometimes develop or ported in consol device, the *input* it's accommodate is not much to handle the *input* the *game* provide and in return some *input* is not implemented. This is caused by the number of *input* Keyboard can handle is more then Joypad on consol could. To tackle this problem a new *input* needed to be combined with Joypad. On x-box consol device an extra device called *Kinect*is used *Kinect* is developed for catching user body motion of the user that currently in frame. This device is one of many kind of *Natural User Interface type device*. *Natural user interface (NUI)* is a form of interface that let user manipulate the interface in natural manner. The example of NUI is mouse device in computer, touch screen, and any device that handled without any need of special training. Research of Penggabungan *Natural User Interface* dengan *Joypad* pada *Role Playing Video Game (RPG)* is start by analize the concept of Natural User Interface that is implemented by *Kinect*. The concept is develop by what kind of data *kinect* give and how to implement it into *Input*. *Kinect* read position of body part, the form of hand is form that is lasso, Close, open and Unknown. From this position *Kinect* form a frame of user body that is catched by *kinect* camera. From this research eight kind of *input* is made. *Input* is formed by the distance between hand to decide whenever or not user use an *input*. The hand form lasso is used as the center of *input* to decide what kind of *input* the hand want. From center lasso the position of X and Y is counted between center and new position to decide what *input* the hand do.

Keyword : Natural User Interface, *Role Playing Video Game*, *kinect*, *Input*.

UNIVERSITAS BRAWIJAYA

DAFTAR ISI

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Role Play Video Game (RPG) adalah sebuah permainan dengan pemain menggerakkan sebuah karakter untuk memainkan peran yang telah didesain. Pada permainan ini karakter memiliki berbagai macam aksi yang dapat dilakukan. Karakter memiliki status, sihir dan berbagai macam hal yang kompleks. *Game* tipe ini memiliki *input* yang banyak dan seringkali *input* dari Joypad yang digunakan oleh mesin konsol seperti X-box tidak mencukupi. (Adams and Rollings 2007)

Dengan keterbatasan tersebut maka diperlukan suatu *input* tambahan selain joypad. Dengan *Joypad* sebagai batasannya maka dipilihlah sebuah pendekatan dengan menggunakan *Natural User Interface* (NUI). *NUI* adalah sebuah pendekatan interaksi antar muka dengan menggunakan manipulasi yang terasa seperti alami. Manipulasi yang terasa alami bisa dicontohkan sebagai interaksi yang dilakukan oleh manusia seperti sentuhan, gerakan badan, dan suara. Contoh dari NUI adalah *Touch Screen* pada *Smartphone, Motion Detection, Motion Capture, Motion manipulation,* dan *voice command*. (Mortensen 2018)

Kinect adalah sebuah perangkat tambahan untuk Xbox yang dikembangkan oleh *Microsoft*. *Kinect* memiliki beberapa komponen yaitu *Red Green Blue Camera* (RGB-Camera), *Depth Sensor, Multiarray Microphone,* dan *Custom Processor*. *RGB-Camera* di gunakan untuk memisahkan setiap object yang ditangkap oleh kamera menjadi tiga warna. *Depth sensor* berguna melihat ruangan dalam dimensi ketiga (3D). *Multyarray microphone* untuk mendeteksi suara dan membedakan gema suara agar dapat digunakan sebagai mikrofon. *Custom processor* digunakan untuk mengolah gambar yang diterima menjadi kerangka yang kemudian diberikan sebagai output (Microsoft 2012).

Kinect membaca *input* dari gambar yang ditangkap oleh kamera kemudian diproses dan dirubah oleh *Kinect* menjadi data kerangka. *Kinect* membaca gerakan tubuh pengguna secara *real time*. Setiap gerakan dari pengguna akan selalu dibaca sebagai *input* oleh *Kinect*. Dengan Perangkat ini dapat diperuntukkan untuk memberikan *input macro* tambahan sebagai alternatif dari permainan *Role Playing Video game*.

Pada *game* Final Fantasy empat belas berbasis *RPG* pemain memiliki *input* sebanyak dua puluh empat. Pada perangkat komputer *input* dapat ditampung oleh keyboard tanpa perlu fungsi lain. Sedangkan pada perangkat konsol *game* melakukan alternatif dengan memisahkan enam belas *input* menjadi dua. Delapan *input* dengan menekan tombol bumper kiri pada joypad ditambahkan dengan menekan salah satu tombol arah dan tombol salah satu tombol simbol pada joypad. Delapan *input* selanjutnya dengan menekan tombol bumper kanan pada joypad ditambahkan dengan menekan salah satu tombol arah dan tombol salah satu tombol simbol pada joypad. Untuk menambahkan empat *input* sisanya dapat dilakukan dengan mengganti susunan *input* yang ada dengan menekan

tombol trigger dan memilih salah satu tombol arah atau tombol simbol pada joypad. Akan tetapi pengguna dapat melakukan kesalahan dikarenakan *input* yang dimasukkan terganti dengan *input* yang diinginkan dikarenakan pengguna lupa mengganti susunan *input*. (Square Enix 2010)

Dari Latar belakang tersebut, maka disusun sebuah skripsi yang berjudul Evaluasi Penggabungan *Natural User Interface* Dengan *Joypad* Pada *Game RPG*. Penelitian ini diharapkan untuk memberi layanan pada pemain *role playing video game* yang menggunakan *Xbox* agar dapat menambahkan *input* yang dibutuhkan dari berbagai macam skill atau sihir yang sering digunakan pada *role playing video game*. Dengan menggunakan *hand gesture* maka diharapkan *Kinect* dapat menangkap *input* yang diinginkan oleh pemain.

1.2 Rumusan masalah

Dari latar belakang tersebut maka dirumuskan beberapa permasalahan sebagai berikut :

1. Bagaimana mengatasi keterbatasan *input* dari *joypad* pada *game RPG ?*

2. Bagaimana pengaruh dari penambahan *Natural User Interface* sebagai *input macro* dalam permainan *game RPG* ?

1.3 Tujuan

Tujuan dari penelitian ini untuk mengatasi keterbatasan *input* dari joypad pada *game RPG* menggunakan pendekatan *NUI.* Mengevaluasi penambahan *NUI* sebagai *input macro* dari *game* bertema *Role Playing Video Game.*

1.4 Manfaat

Manfaat yang diharapkan dalam penelitian ini adalah :

1. Memahami cara merancang sebuah sistem untuk diimplementasikan dalam *kinect*.
2. Memberikan gambaran dari potensi yang dapat dikembangkan dari *kinect*.
3. Memberikan menambahkan pilihan *input macro* yang dapat digunakan dalam melakukan *input* pada *game* digital.
4. Penelitian ini dapat dikembangkan sebagai manipulasi *Augmented reality* atau *Virtual reality*.

1.5 Batasan masalah

Penelitian ini akan dibatasi oleh beberapa hal pokok untuk mencegah terjadinya pelebaran topik pembahasan. Hal-hal pokok tersebut yaitu :

1. Penggunaan *Joypad* untuk mensimulasikan posisi pengguna dalam permainan.

2. Sistem hanya perlu menjalankan fungsi untuk menerima *input* dari satu orang pemain *(Singleplayer).*
3. Implementasi sistem menggunakan perangkat lunak *Microsoft Visual Studio*.
4. Sistem dirancang pada *platform* PC dengan menggunakan sistem operasi Windows 8.1 keatas.
5. Pengembangan yang akan dilakukan hingga pembuatan prototype yang mampu menjalankan layanan hand gesture recognition.
6. Pengembangan menggunakan metode water fall hingga tahap evaluasi.
7. Pengujian menggunakan metode Black box.

1.6 Sistematika Pembahasan

1. Bab I Pendahuluan

Memuat latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat dan sistematika penulisan.

2. Bab II Dasar Teori

Memaparkan teori dasar dan teori pendukung yang berhubungan dengan Pengembangan *Hand Gesture Recognition* Menggunakan *Kinect* Sebagai *Input Macro* Pada *Game RPG*.

3. Bab III Metodologi

Berisi tentang langkah-langkah dalam melakukan observasi, pengumpulan, dan pengolahan data. Data yang dicari terkait dengan evaluasi penggabungan *Natural User Interface* Sebagai *Input Macro* Pada *Game* RPG.

4. Bab IV Pengembangan dan Evaluasi

Membahas mengenai langkah-langkah pengembangan sistem *input macro* berdasarkan analisa data yang telah dikumpulkan. Setelah itu mengevaluasi hasil perancangan dalam konsep.

5. Bab V Penutupan

Memuat kesimpulan yang diperoleh dari hasil penelitian beserta saran untuk pengembangan penelitian yang selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 *Natural User Interface*

Natural user interface (NUI) adalah pendekatan antarmuka yang interaksi secara langsung dan konsisten dengan gerakan *natural* yang kita lakukan. Penggunaan kata *natural* memiliki arti bahwa pendekataan ini mempelajari antar muka yang dapat digunakan tanpa perlu pelatihan yang rumit. Dengan *NUI* maka pengguna dapat mengoperasikan sistem tanpa perlu pelatihan khusus dan dapat terbiasa menggunakannya.

Pendekatan ini digunakan untuk mengembangkan teknologi seperti *touch screen* pada komputer, atau smarthphone. Penggunaan *Touch Screen* tidak diperlukan pembelajaran yang rumit dan lama. Hanya memerlukan menit dan pengguna dapat mengoperasikan perangkat tersebut.

Dari berbagai metode yang ada *Hand gesture recognition* digunakan pada penelitian ini. *Hand gesture recognition* adalah pendekatan pengenalan gerakan tangan untuk diterjemahkan kedalam output pada perangkat. Penelitian dalam bidang ini dapat dibedakan menjadi tiga level yaitu *Static hand gesture recognition, dynamic gesture recognition*, dan Dimensi Tiga (3D) hand gesture recognition. (Yuhai, Jing and Zhaojie 2016)

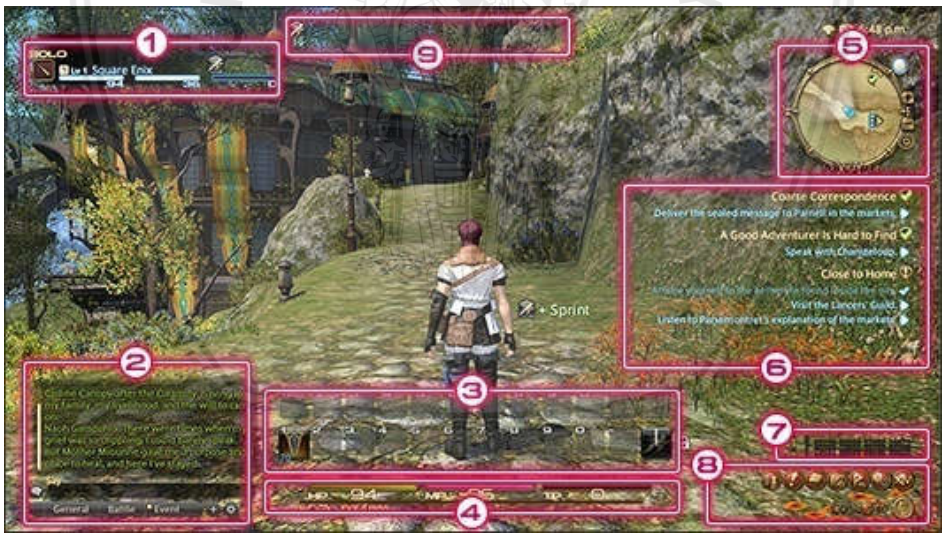
Static dan dynamic pada dasarnya menggunakan gambar warna dua Dimensi sebagai *input* yang didapat dari RGB camera tetapi menghiraukan informasi kepadatan yang didapat oleh depth sensor. Berbeda dengan ke dua level sebelumnya 3D hand gesture menggunakan dasar 3D, dimana informasi kedalaman yang didapat dari depth sensor digunakan secara sepenuhnya agar didapat data yang lebih efektif untuk digunakan oleh hand gesture recognition. Interaksi *NUI* dimana perangkat kamera bernama *kinect* mengambil gambar berupa gerakan yang kemudian diterjemahkan menjadi *input* seperti contoh dibawah ini :

Gambar 2.1 *NUI* berupa gerakan pada *kinect*.

2.2 *Role Playing Video game*

Role playing video game (RPG) adalah sebuah genre dimana pemain menjalankan seorang atau beberapa karakter untuk menjalankan narasi yang dari dunia yang telah diciptakan. *Game* jenis ini terinspirasi dari permainan papan atau lebih dikenal sebagai table top *game* seperti Warhammer dan dungeon & dragon. *Game* tipe ini menggunakan penamaan istilah, seting dan mekanika permainan yang sama dengan permainan papan yang dinarasikan. Kesamaan lain dari *game* tipe ini dalah dengan adanya narasi, perkembangan karakter dan kompleksitasnya. Keunggulan dari media digital adalah dengan menghilangkan *game*master dimana semua sudah diatur oleh sistem (Adams and Rollings 2007).

Role Playing Video Game memberikan pilihan kepada pemain untuk menjalankan karakter dengan seting yang akan menjalankan karakter secara langsung ataupun tidak langsung. Pilihan dari pemain akan dijalankan oleh karakter sesuai dengan desain dari skill tersebut. keberhasilan dari aksi tersebut ditentukan dari beberapa atribut dari karakter. beberapa dari *game* jenis ini memiliki pengguna sihir dimana sihir tersebut dapat digunakan untuk beberbagai macam hal seperti menyerang, bertahan, dan melakukan dukungan kepada karakter lain. Berikut adalah contoh dari RPG yang akan di teliti :



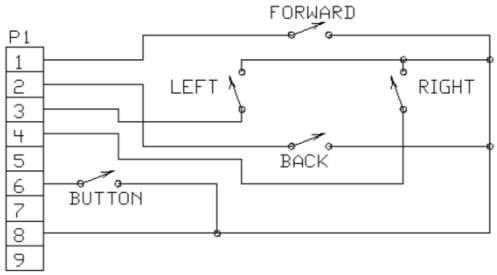
Gambar 2.2 Final Fantasy XIV (Square Enix 2010)

2.3 Joypad

Joypad adalah instrument yang digunakan oleh pengguna untuk berinteraksi dengan *game*. Perangkat keras ini merupakan alat yang sering berinteraksi dengan pengguna. Joypad memiliki ciri khas tombol dengan arah panah dan beberapa tombol *input*. (Lu 2003)

Joypad memberikan pengalaman yang imersif dikarenakan joypad adalah perwakilan dari kegiatan pengguna dalam kegiatan yang disediakan pada *game*.

Karenanya Joypad dapat membuat pengalaman tersebut menjadi frustasi, dikarenakan *input* yang digunakan oleh pengguna tidak sesuai dengan hasil yang diinginkan pengguna.



Gambar 2.3 *Contoh Diagram Sirkuit sederhana dari Joypad* (Lu 2003)

2.4 *Kinect*

Kinect adalah suatu perangkat tambahan yang ada pada *platform Xbox 360, Xbox one* dan Komputer. *Kinect* dipergunakan untuk menangkap gambar dari user yang kemudian akan diproses menjadi gambar infra merah untuk membedakan antara *player* dan *environment*. Perangkat ini menggabungkan empat macam feature yaitu depth *sensor, RGB camera, multiarray microphone*, dan *custom processor* (Microsoft 2012).

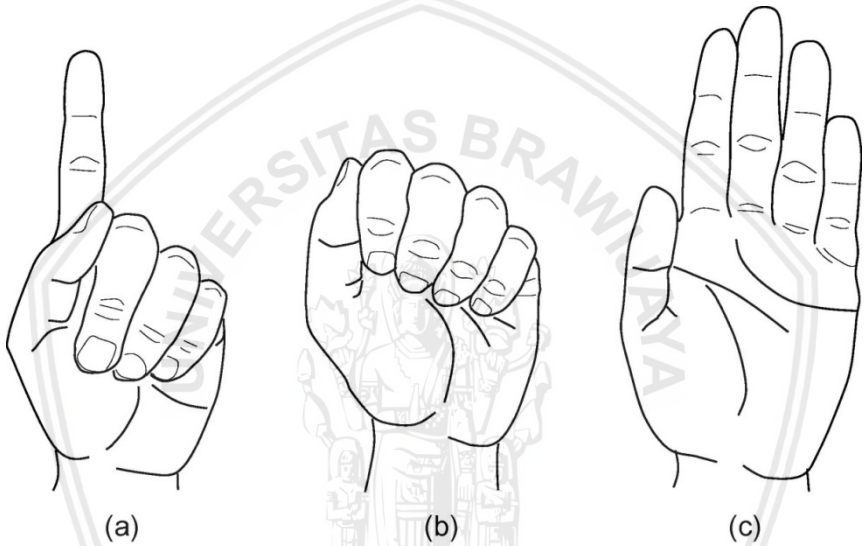
RGB camera merupakan kamera yang memberikan tiga warna dasar yang digunakan untuk membedakan setiap benda yang direkam. *Depth sensor* digunakan untuk melihat ruang lingkup dalam 3D dengan mengkombinasikan sensor monochrome CMOS dalam berbagai kondisi cahaya. *Multi array microphone* digunakan untuk mendeteksi lokasi suara dan mengekstrak gema dari suara. Custom processor digunakan untuk membedakan bagian tubuh manusia dan benda lain.

Kinect telah digunakan oleh berbagai macam penelitian dalam bidang visi komputer. *Kinect* mendeteksi informasi kedalaman secara *Time of flight* dan menyediakan berbagai macam kualitas data yang bagus seperti warna, infra merah, kedalaman, rangka, dan Solid SDK.

Gambar 2.4 *Contoh Testing pada kinect* (Yuhai, Jing and Zhaojie 2016)

Pada dokumentasi *kinect* terdapat beberapa informasi mengenai beberapa data yang data ditangkap oleh *kinect*, data tersebut dapat dibagi menjadi berikut :

1. Posisi User
2. Bentuk Rangka dari user
3. Posisi tangan yang direpresentasikan menjadi
   a. Posisi X
   b. Posisi Y
4. Simbol Tangan yang berada di dalam *Library* ditunjukkan pada gambar 5
   a. Lasso
   b. Closed
   c. Open



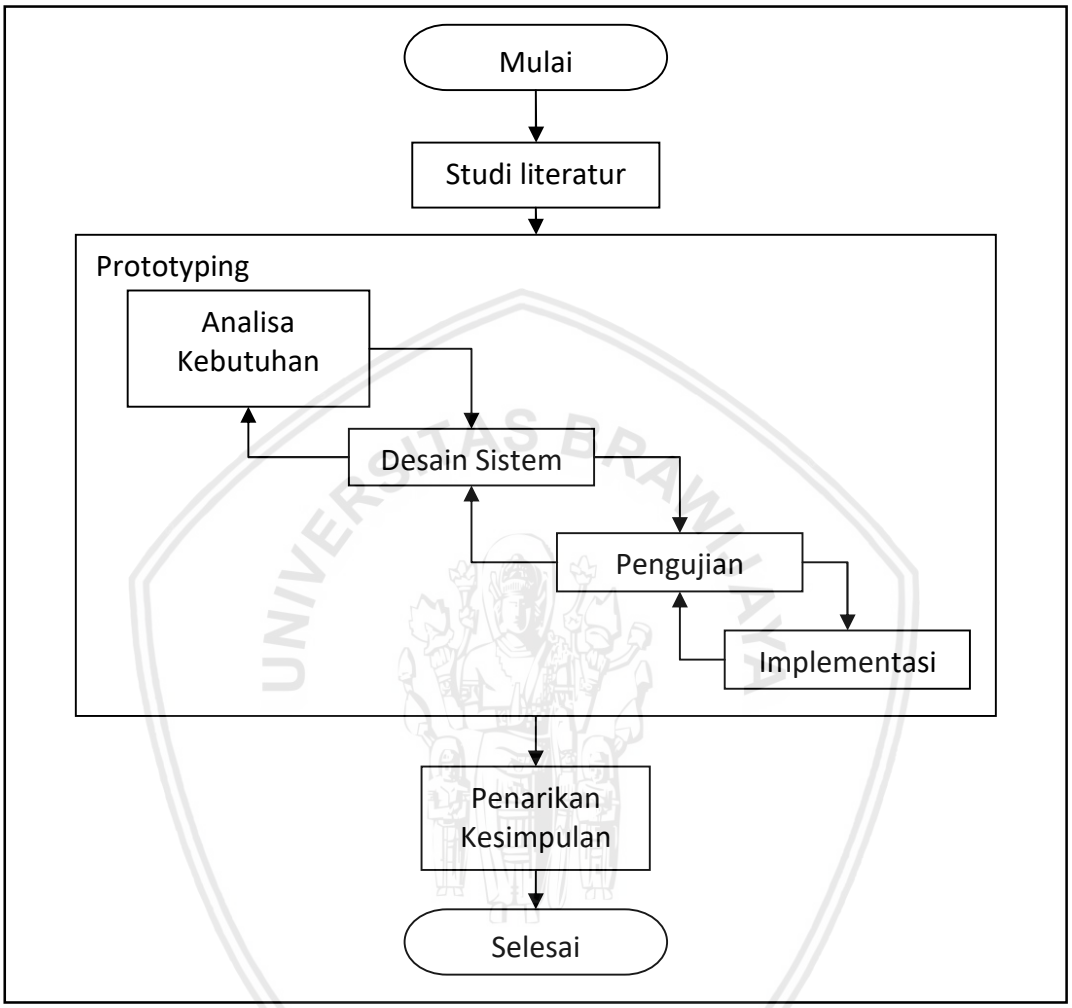
Gambar 2.5 Bentuk dari tangan yang dibaca oleh *Kinect* (E.H. Shortliffe 2017)

2.5 *Prototyping*

Prototyping adalah metode penelitian dengan memahami apa yang dibutuhkan pada desain dan diterjemahkan pada bentuk model uji atau *prototype.* Proses prototyping dimulai dengan melakukan analisa bagaimana penelitian menginginkan hasil yang dikeluarkan. Selanjutnya dilakukan desain pada sistem untuk mendeskripsikan apa yang perlu sistem lakukan untuk memenuhi kebutuhan tersebut. Setelah sistem terdesain dilakukan pengujian apakah sistem dapat diimplementasikan, langkah ini dilakukan berulang ulang hingga dapat diciptakan prototype yang siap diuji. Setelah sistem terbentuk maka dilakukan implementasi pada tahap ini dilakukan pengujian apakah sistem siap dioperasikan atau tidak.

BAB 3 METODOLOGI

Tahapan pengembangan dari penelitian ini dijabarkan dengan diagram seperti dibawah ini :



Gambar 3.1 Diagram Pengembangan Sistem

3.1 Studi Literatur

Studi Literatur adalah tahap pengumpulan bahan dan informasi yang menjadi dasar teori dalam penelitian ini. Dasar teori tersebut ditelusuri dari berbagai sumber, diantaranya dalam jurnal, buku, artikel, dan konsultasi dengan para ahli. Informasi pendukung dalam penelitian ini meliputi *Natural User Interface ,joypad,*dan *Role Playing Video Game*.

3.1.1 Natural User Interface

Natural User interface adalah sebutan dari sebuah sebutan untuk tatap muka yang menggunakan gerakan alami dari pengguna. Berbagai contoh dari *Natural User Interface* adalah layar sentuh pada layar sentuh, mouse pada computer, *Joypad* pada *Nintendo Wii*, dan sensor gerak pada *Kinect.*

Perangkat *kinect* digunakan untuk mengakomodasi dari *Natural User Interface* penelitian ini. Dari beberapa sumber telah dipelajari maka didapatkan beberapa data yang dibaca oleh *kinect* yaitu Posisi Tangan, bentuk yang dibuat tangan dan jari, serta bentuk tubuh.

3.1.2 Kinect

Perangkat *Natural User Interface* yang terhubung dengan *game* konsol dan juga computer adalah *Kinect. Kinect* adalah perangkat yang dikembangkan *Microsoft* untuk memberikan *input* body gesture kepada perangkat konsol *X-box*. Perangkat ini dikembangkan untuk menangkap gerakan pengguna yang kemudian diterjemahkan menjadi *input* kepada *game*.

3.1.3 Joypad

Joypad adalah istilah untuk perangkat yang digunakan oleh user sebagai media untuk memanipulasi permainan. Pada perkembangannya Joypad memiliki berbagai iterasi mulai dari berbentuk control pesawat, antar muka kepada pengguna dalam bentuk papan dengan tombol hingga yang berbentuk lebih pada simulasi seperti pedal gas kemudi mobil yang diperuntukkan untuk mensimulasikan pengendalian kendaraan. Joypad diteliti sebagai perangkat acuan untuk meneliti bagaimana mengimplementasikan metode yang dikembangkan.

3.1.4 Role Playing Video Game

Role Playing adalah sebuah kegiatan dimana seseorang mengambil sebuah identitas dari dunia fantasi untuk bermain peran dalam cerita atau permainan fantasy. Konsep ini dikembangkan menjadi sebuah Video *game* untuk mensimulasikan pengunaan konsep fantasy seperti sihir, binatang fantasy, dan kemampuan yang tidak dapat disimulasikan didunia nyata.

Pada perkembangannya *game* bertema ini memakan banyak tombol. Beberapa judul pada *game* ini menggunakan beberapa alternative akan tetapi hal tersebut mengubah beberapa cara kerja dari permainan hingga perbedaan dengan versi computer yang besar. Dikarenakan kekurangan yang didapat pada joypad meskipun beberapa hal dapat di akomodasi akan tetapi hal itu dapat mengurangi kecepatan akses dari permainan. Dari studi ini maka keluarlah ide untuk menambahkan *Natural User Interface* sebagai *input* tambahan untuk mengakomodasi kekurangan dari *input* pada *game* bertema *Role Play.*

3.2 Pengembangan Konsep

Metodologi pengembangan Konsep ini dilakukan dengan cara meneliti dasar dari pemrograman *kinect*. Dari studi literatur yang dilakukan untuk menjalankan *kinect* maka dirancang konsep untuk menggunakan *kinect* dengan joypad secara bersamaan tanpa terjadi konflik *input*. Untuk mendapatkan hasil yang sesuai dengan yang diinginkan dilakukan bebrapa tahap perancangan.

Pengembangan dari konsep ini dibagi menjadi beberapa subbab yaitu analisa kebutuhan, pengembangan, implementasi, dan evaluasi. Analisa kebutuhan dilakukan sebagai acuan apa saja yang diperlukan untuk mengembangkan metode. Pengembangan dilakukan dari hasil analisa kebutuhan yang telah didapat. Setelah pengembangan telah didapat maka dilakukan implementasi yang dijabarkan pada babnya. Setelah implementasi selesai dilakukan evaluasi untuk meneliti apakah ada kesalahan atau hasil yang didapat dari implementasi sudah sesuai atau tidak.

3.2.1 Analisa Kebutuhan

Analisa kebutuhan dilakukan untuk menentukan apa saja yang dibutuhkan untuk membangun system *macro*. Berikut adalah kebutuhan fungsional yang dibutuhkan :

1. Fungsi pembacaan lokasi tangan
2. Fungsi pembacaan Hand Gesture
3. Fungsi pembacaan tanda yang dibuat

3.2.2 Desain Sistem

Desain Sistem dilakukan untuk mendapatkan konsep sistem *macro.* Dari berbagai literatur yang telah dipelajari maka akan dibangun sistem *macro*. Sistem dikembangkan dengan tujuan untuk dapat membedakan apakah user sedang melakukan *input* atau tidak.

Sistem dikembangkan untuk dapat digunakan bersamaan dengan *Joypad* saat memainkan *game* dan sistem tidak mengganggu pengguna dalam berinteraksi dengan *joypad.*

Pada pengembangan ini digunakan *source code* yang telah disediakan oleh website pterneas (Pterneas, pterneas 2014). Pada *source code* telah ada *library* yang disediakan untuk dapat memanggil beberapa data yang diterima dalam *kinect.* Beberapa data yang dimanipulasi adalah pemanggilan bentuk tubuh, keadaan dari jari yang tangan bentuk, dan posisi dimana tangan berada.

3.2.3 Pengujian Sistem

Pengujian sistem dilakukan setelah desain dari sistem selesai. Dari sistem tersebut dilakukan uji berkali-kali untuk mengetahui kesesuaian *input* yang dilakukan oleh responden dengan output yang diberikan. Untuk mendapatkan data dari program maka dilakukan pengujian dengan metode fungsional (blackbox – testing) dan playtesting yang dilakukan oleh beberapa orang responden secara acak. Responden tersebut akan diberikan kuisioner yang berisi pertanyaan terkait play testing yang telah mereka lakukan terhadap sistem Hand gesture.

Hasil dari pengujian *game* akan dievaluasi, hal tersebut bertujuan untuk menghasilkan suatu kesimpulan. Kesimpulan tersebut dapat memicu saran yang

dapat digunakan untuk memperbaiki kesalahan dan pertimbangan untuk mengembangkan Sistem Hand Gesture yang lebih sempurna kedepannya.

3.2.4 Implementasi

Pengimplementasian metode ke dalam bentuk perangkat lunak akan dilakukan setelah pengembangan. Tahap ini akan melalui aktivitas-aktivitas berikut :

1. **Implementasi Gesture :** Penerapan gesture terhadap *kinect* untuk mempelajari gerakan tangan yang akan digunakan atau dipakai pada sistem.
2. **Implementasi Match *Input* :** Melakukan pengecekan ulang apakah *input* yang diberikan sama dengan output yang diberikan pada pengujian.

3.3 Pengambilan Kesimpulan dan Saran

Dari Tahap perancangan maka dilakukan penarikan kesimpulan dan pengumpulan saran yang didapat dari para tester. Dengan berbagai macam saran dan data yang didapat dari Evaluasi maka akan ditarik kesimpulan untuk pengembangan di kedepannya.

BAB 4 TAHAP PENGGABUNGAN *NUI* DENGAN *JOYPAD*

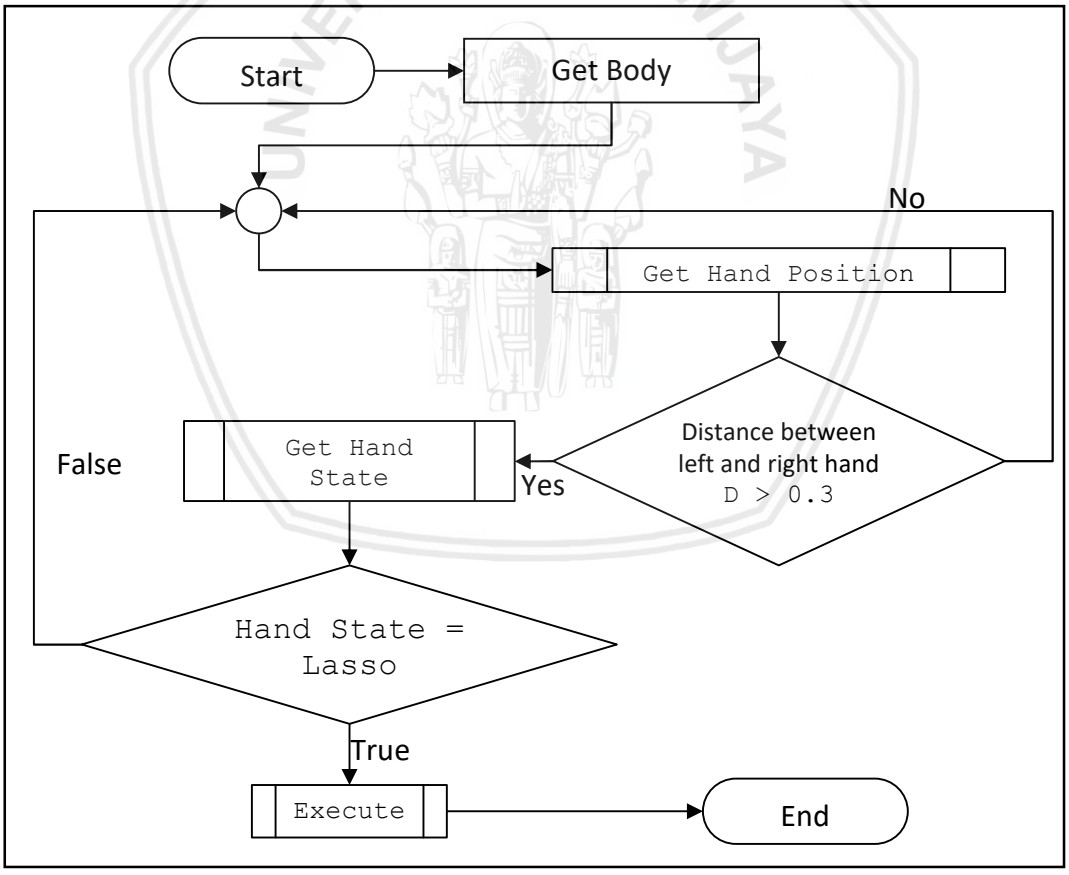
4.1 Perancangan

Dari Studi literature didapatkan metode-metode yang diperlukan untuk membangun konsep *Macro*.

4.1.1 Pembuatan Konsep Utama

Pembuatan konsep utama adalah penjelasan bagaimana metode berjalan mulai dari *kinect* mendapatkan *input* data hingga *kinect* memberikan output. Didalam konsep ini hanya akan memberikan gambaran bagaimana *input* didapat dan *input* keluar.

Didalam konsep akan diperjelaskan lagi kedalam berbagai subbab yang akan dijelaskan lebih detail seperti bagaimana sistem mendapatkan posisi tangan dan bagaimana system dapat membedakan antara pengguna melakukan *input* atau sedang menggunakan Joypad secara keseluruhan. Proses ini dapat ditunjukkan pada gambar 7.



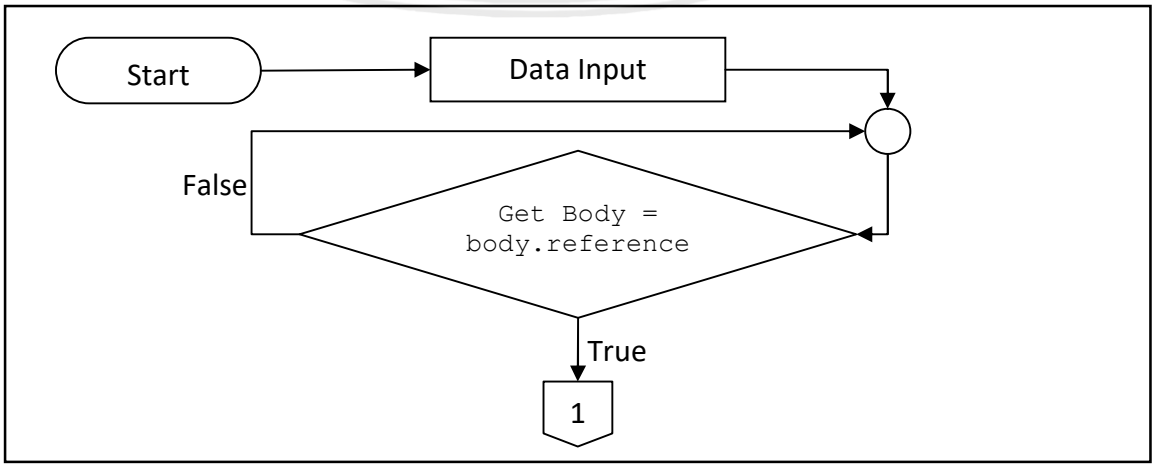
Gambar 4.1 Flow Chart dari Konsep Penambahan *Input*

Pada gambar 4.1 menggambarkan bahwa proses pembuatan konsep utama adalah sebagai berikut:

1. Memanggil Body agar sistem dapat menangkap bentuk tubuh dan nama bagian tubuh pengguna.
2. Sistem memanggil data tangan dari data yang diterima body, dan bentuk tubuh yang diterima oleh *kinect*.
3. Dari data yang diterima dipanggil posisi tangan kanan dan kiri.
4. Dari data kedua tangan yang diterima diperlukan sebuah batasan atau threshold agar dapat dibedakan antara pengguna melakukan input atau tidak.
5. Threshold berupa perhitungan pitagoras jarak dari kedua posisi tangan.
6. Hasil jarak antar kedua tangan didapatkan dari perhitungan dengan menggunakan rumus pitagoras antara posisi X dari Tangan kanan (ka) dan tangan kiri (ki) seperti berikut : D = $\sqrt{(Xka - Xki) + (Yka - Yki)}$.
7. Apabila Jarak yang antara kedua tangan lebih dari 0.3 meter maka system akan mulai membaca bentuk tangan yang dibentuk dari kedua tangan.
8. Sistem akan melakukan respon apabila *kinect* memberikan data Lasso kepada tangan, apabila linect memberikan data lain atau tidak dapat menentukan apa yang dibentuk maka system tidak akan merespon dan kembali pada pencarian posisi tangan.
9. Apabila tangan membaca *kinect* dengan symbol lasso maka akan dieksekusi perintah yang telah di tentukan oleh system sesuai dengan gerakan yang telah ada di system

4.1.2 Fungsi Untuk Mendapatkan Bentuk Tubuh Pengguna

Sebelum *Kinect* menggunakan *input* yang diterima menjadi data yang dapat dikerjakan *Kinect* memerlukan data berupa bentuk tubuh dari pengguna. Maka dipanggil fungsi untuk memanggil Rangka untuk menggambarkan gerakan tubuh atau bentuk tubuh pengguna. Berikut adalah alur dari Fungsi ini :



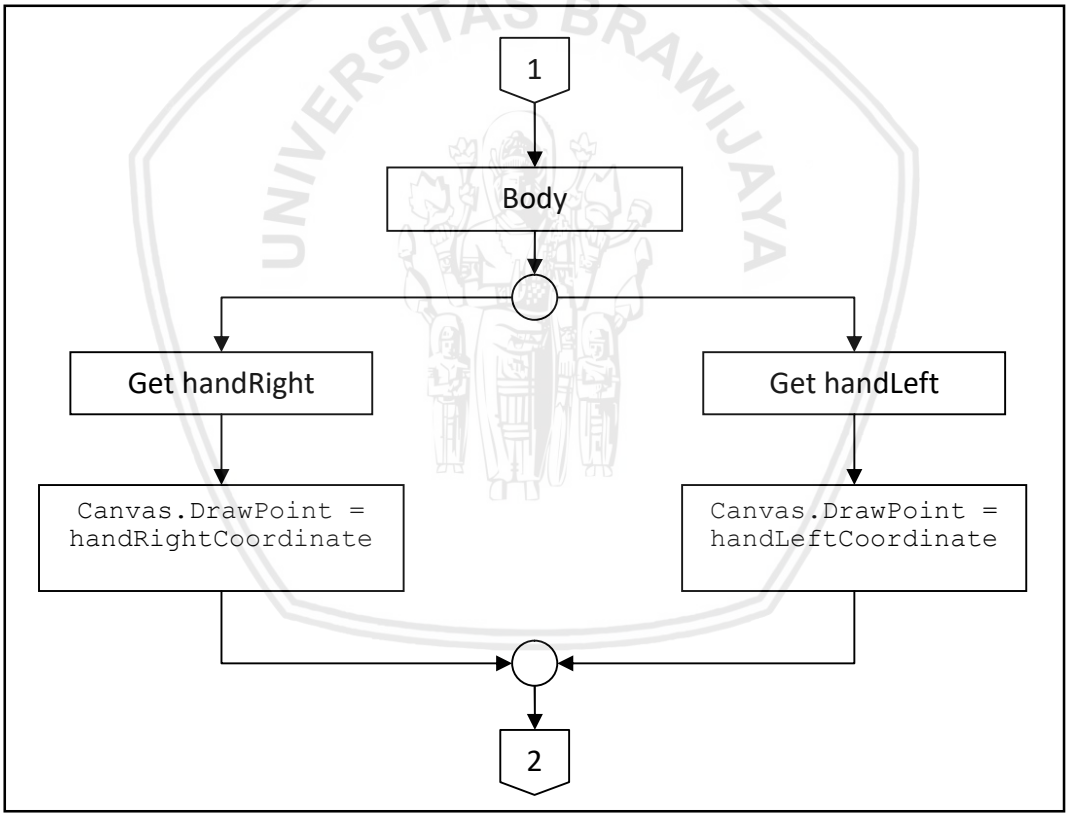
Gambar 4.2 Alur Get Body.

Pada gambar 4.3 adalah proses yang dilakukan untuk membedakan data yang diberikan kepada *kinect* menjadi *input* yang dapat diproses :

1. Data *input* yang diterima oleh *kinect* diproses dalam program.
2. Dari berbagai data yang didapat library yang telah disediakan akan memilah data tersebut hingga menjadi body.
3. Apabila dari data yang diterima ditemukan bentuk tubuh maka akan dibaca sebagai Body yang kemudian diterjemahkan oleh program untuk output.

4.1.3 Fungsi `HandCoordinate`

Setelah Proses pembentukan data Body selesai maka dilakukan pencarian posisi tangan dari pengguna yang tertangkap oleh *kinect*. Proses ini memanggil setiap tangan yang dapat ditunjukkan sebagai berikut :



Gambar 4.3 Alur dari penggambaran posisi tangan.

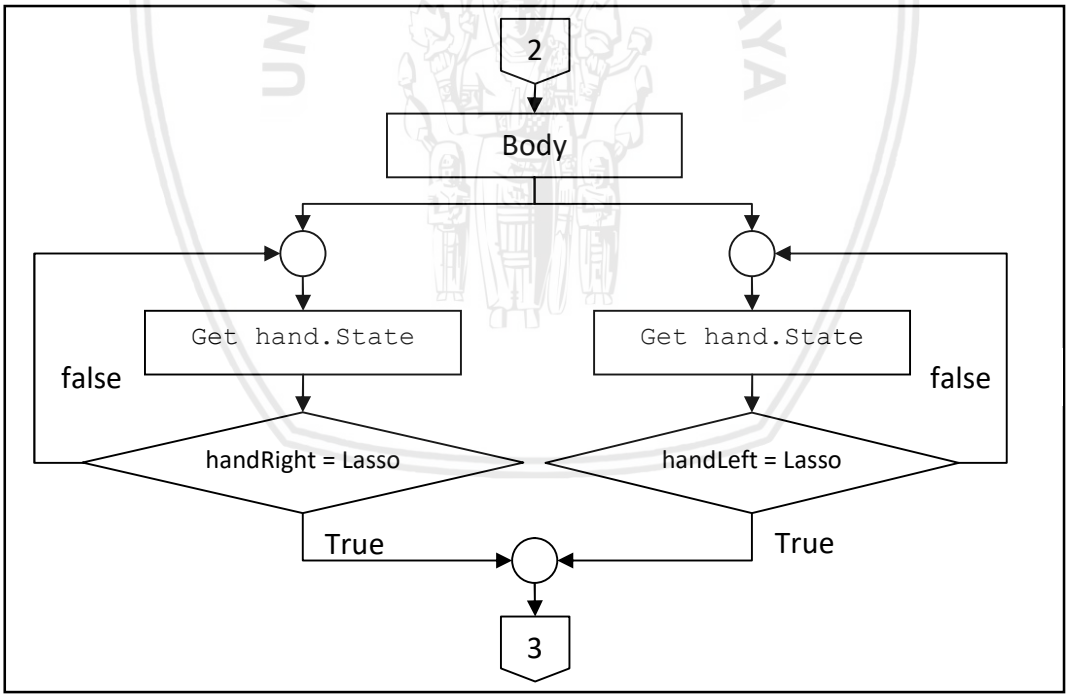
Data yang didapatkan dari body dipilih handRight dan handLeft untuk kemudian diproses dan digambar secara visual untuk mengetahui posisi dari data tangan yang terbaca. Dari gambar 4.4 dijelaskan sebagai berikut :

1. Data dari Body didapat dan diproses
2. Kemudian dipilih handRight dan handLeft.
3. Dari data yang didapat dibedakan menjadi dua proses sesuai dengan nama kedua tangan.

4. Data yang didapatkan dari handRight kemudian diproses.
5. Pada handRight diambil posisinya dalam gambar yang ditampilkan oleh *kinect*.
6. Kemudian dari data tersebut ditunjukkan dimana letak kordinat tersebut dengan `Canvas.Drawpoint` untuk menggambarkannya dalam bentuk titik.
7. Data yang didapatkan dari handLeft kemudian diproses.
8. Pada handLeft diambil posisinya dalam gambar yang ditampilkan oleh *kinect*.
9. Kemudian dari data tersebut ditunjukkan dimana letak kordinat tersebut dengan `Canvas.Drawpoint` untuk menggambarkannya dalam bentuk titik.
10. Dari kedua data tersebut kemudian digabungkan dan ditampilkan kedalam gambar yang ditampilkan oleh *kinect*.

4.1.4 Fungsi `Get Handstate`

Dari data posisi tangan dilanjutkan kepada penggunaan fungsi status tangan apabila jarak dari kedua tangan terpenuhi yaitu jarak 0.3 dari kedua object. Pada fungsi ini dilihat data dari handRight dan handLeft status dari keua data tersebut yang ditunjukkan sebagai berikut :



Gambar 4.4 Alur Pencarian Status Tangan

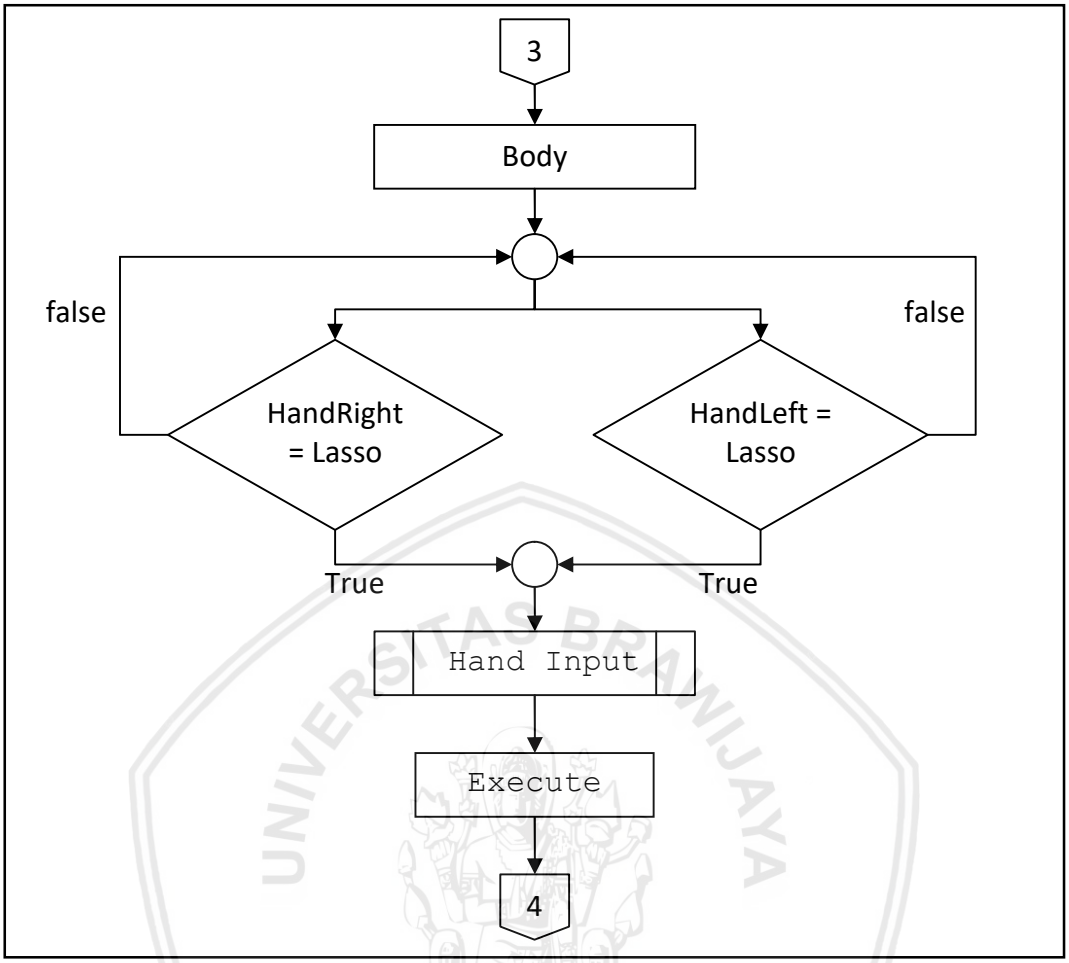
Sebelum *macro* dapat di-eksekusi perlu dilakukan cek apakah pengguna mau melakukan *macro* atau tidak dengan melakukan cek bentuk tangan pengguna yang ditunjukkan oleh gambar 4.5 dijelaskan sebagai berikut :

1. Data dari Body dipanggil untuk mendapatkan data handState
2. Dari data yang diberikan oleh Body dibaca apakah data dari setiap tangan.

3. Setelah data didapat maka dibagi kembali menjadi dua proses lagi untuk membaca data dari tangan kanan dan kiri.
4. Tangan kanan dibaca untuk melihat apakah tangan ini melakukan *input* atau tidak.
5. Dari bacaan tangan kanan dicocokkan dengan status *Lasso* pada tangan kanan.
6. Apabila kondisi ini salah atau `False` maka kembali kepembacaan tangan.
7. Tangan kiri dibaca untuk melihat apakah tangan ini melakukan *input* atau tidak.
8. Dari bacaan tangan kiri dicocokkan dengan status *Lasso* pada tangan kanan.
9. Apabila kondisi ini salah atau `False` maka kembali kepembacaan tangan.
10. Dari kedua kondisi ini maka akan dieksekusi apabila ada yang memiliki kondisi True.

4.1.5 Fungsi `Execute`

Setelah semua kondisi telah terpenuhi maka dilanjutkan dengan melakukan eksekusi terhadap *input* yang diinginkan. Untuk penelitian ini digunakan delapan *input* yang dibagi menjadi empat yang diwakili oleh setiap tangan. Langkah dari setiap *input* ditunjukkan sebagai berikut :



Gambar 4.5 Alur Pengeksekusian *Input* oleh tangan kanan

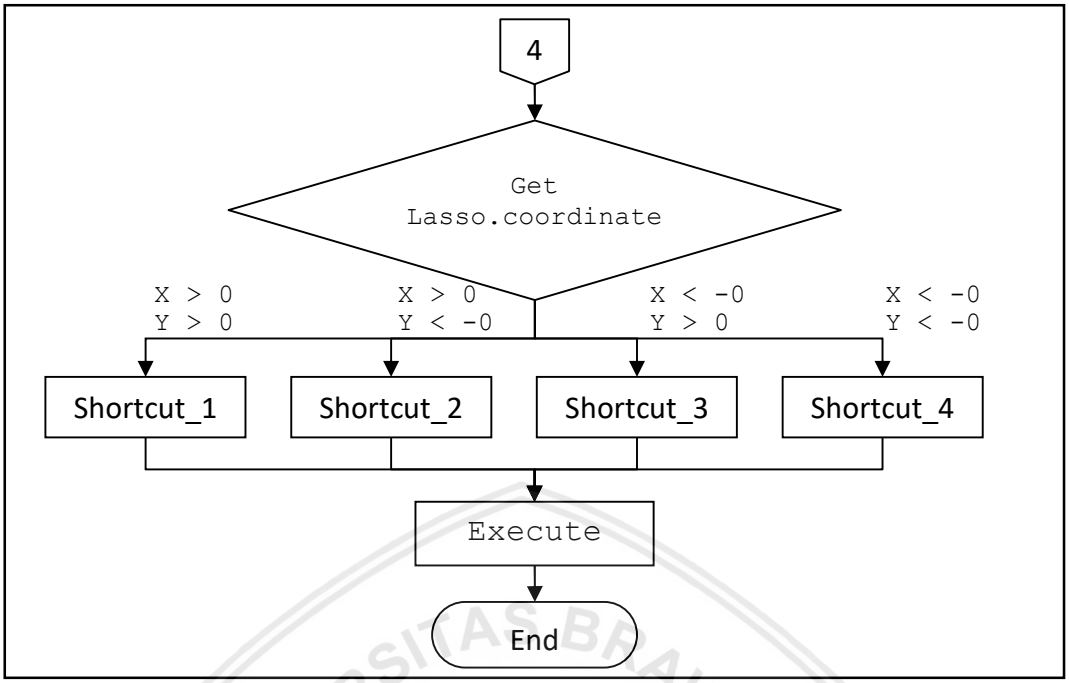
Dari Gambar 4.6 ditunjukkan bahwa pengeksekusian *input* dilakukan dengan cara sebagai berikut :

1. Dari fungsi terakhir dilihat bagian tangan manakah yang melakukan status lasso.
2. Status dicari dan disamakan dari data yang library dari Body berikan.
3. Proses dibagi menjadi dua yaitu proses tangan Kanan dan tangan Kiri.
4. *Input* akan dieksekusi berdasarkan dari kedua tangan yang mengeluarkan status Lasso.

Dari fungsi ini akan dijalankan berdasarkan tangan mana yang melakukan lasso pertama kali. Dari setiap tangan memiliki *input* tersendiri yang dijelaskan pada fungsi berikutnya.

4.1.6 Kode `Hand` ***`Input`***

Pada tangan kanan setelah tangan kana terdeteksi sebagai lasso maka akan dihitung kordinat X dan Y dari posisi awal lasso terbentuk keposisi tangan kanan yang baru untuk mengeksekusi *input*. Alur dari Right Hand *Input* menjadi flowchart pada gambar 4.7 seperti berikut ini :



Gambar 4.6 Alur Pengeksekusian *Input* oleh tangan kanan

Berikut adalah penjelasan pengeksekusian yang dilakukan oleh tangan kanan pada gambar 4.7:

1. Setelah tangan melakukan lasso maka coordinat tangan direset kembali menjadi 0.0 yang dilambangkan dengan Lasso Coordinate.
2. Dari Lasso.Coordinate maka akan dihitung lagi jarak X dan Y.
3. Dari hasil yang dikeluarkan Lasso.coordinate dibagi menjadi 4 output.
4. Shortcut_1 dijalankan apabila nilai dari X > 0 dan Y > 0.
5. Shortcut_2 dijalankan apabila nilai dari X > 0 dan Y < -0.
6. Shortcut_3 dijalankan apabila nilai dari X < -0 dan Y > 0.
7. Shortcut_4 dijalankan apabila nilai dari X < -0 dan Y < -0.
8. Dari ke-empat *input* tersebut akan diambil satu yang dilakukan oleh user.

4.2 Implementasi

4.2.1 Fungsi Get Body

```
1   IList<Body> _bodies;
2   using (var frame =
3   reference.BodyFrameReference.AcquireFrame())
4   {
5   if (frame != null)
6   {
7   canvas.Children.Clear();
8   _bodies = new Body[frame.BodyFrameSource.BodyCount];
9
10  frame.GetAndRefreshBodyData(_bodies);
11
12  foreach (var body in _bodies)
13  {
14  if (body != null)
15  {
16  if (body.IsTracked)
17  {
```

Gambar 4.7 Kode untuk memunculkan data tubuh pengguna.

Kode berikut ini digunakan untuk mengambil gambar yang ditangkap oleh *kinect* menjadi body joint yang dapat dimanipulasi menjadi data *input*. Berikut adalah penjelasan untuk kode :

1. Baris satu adalah Library yang disediakan oleh Visual Studio untuk memanggil bentuk tubuh dari data yang dideteksi *kinect*.
2. Baris Tiga Hingga sebelas digunakan untuk mendeteksi apakah ada tubuh yang terdeteksi.
3. Pada baris tiga digunakan untuk mengambil data body pada frame, baris ke tiga digunakan untuk memanggil referensi tubuh kedalam frame.
4. Apabila frame tidak ada *input* maka baris tujuh akan dieksekus dan semua data akan dibersihkan dari layar tampilan.
5. Baris tiga belas digunakan untuk merubah setiap bagian dari _bodies diwakilkan dengan kata _bodies sebagai body
6. Baris lima belas dan tujuh belas digunakan untuk status terdeteksinya tubuh.

4.2.2 Fungsi `Get Hand Position`

```
1  Joint handRight = body.Joints[JointType.HandRight];
2  Joint handLeft = body.Joints[JointType.HandLeft];
3  Joint head = body.Joints[JointType.Head];
4
5  canvas.DrawPoint(handRight, _sensor.CoordinateMapper);
6  canvas.DrawPoint(handLeft, _sensor.CoordinateMapper);
7  canvas.DrawPoint(head, _sensor.CoordinateMapper);
8
9  float posisiTanganKananX = handRight.Position.X -
10 head.Position.X;
11 string tanganKananX = posisiTanganKananX.ToString("0.##");
12 float posisiTanganKananY = handRight.Position.Y -
13 head.Position.Y;
14 string tanganKananY = posisiTanganKananY.ToString("0.##");
15 float posisiTanganKananZ = handRight.Position.Z -
16 head.Position.Z;
17 string tanganKananZ = posisiTanganKananZ.ToString("0.##");
18
19 float posisiTanganKiriX = handLeft.Position.X -
20 head.Position.X;
21 string tanganKiriX = posisiTanganKiriX.ToString("0.##");
22 float posisiTanganKiriY = handLeft.Position.Y -
23 head.Position.Y;
24 string tanganKiriY = posisiTanganKiriY.ToString("0.##");
25 float posisiTanganKiriZ = handLeft.Position.Z -
26 head.Position.Z;
27 string tanganKiriZ = posisiTanganKiriZ.ToString("0.##");
28
29 float handDistance = Distance_From_To(posisiTanganKananX,
30 posisiTanganKananY, posisiTanganKiriX, posisiTanganKiriY);
```

Gambar 4.8 Code Untuk menggambarkan posisi tangan

Pada gambar 4.9 digunakan untuk mengambil data tentang tangan dan menampilkan data tersebut kedalam frame. Berikut adalah penjelasan mengenai kode tersebut :

1. Pada baris satu hingga baris tiga digunakan untuk mendapatkan referensi data kepala, tangan kanan, dan tangan kiri dari body.
2. Baris lima sampai 7 digunakan untuk menggambarkan titik pada frame untuk menggambarkan posisi dari tangan kanan, tangan kiri, dan kepala berdasarkan kordinat yang diberikan oleh body.
3. Baris Sembilan hingga baris tujuh belas digunakan untuk menghitung posisi x,y dan z dari tangan kanan.
4. Posisi dari tangan kanan dihitung dari mengurangi posisi kepala sebagai acuan dan posisi tangan kanan menggunakan variable `float`.
5. Hasil dari `float` kemudian nilainya ditampilkan dengan variable String.
6. Untuk baris Sembilan belas hingga baris dua puluh tujuh digunakan untuk menghitung posisi x,y dan z dari tangan kiri.

7. Posisi dari tangan kiri dihitung dari mengurangi posisi kepala sebagai acuan dan posisi tangan kanan menggunakan variable `float`.
8. Hasil dari `float` kemudian nilainya ditampilkan dengan variable String.
9. Dari hasil kedua tangan yang didapat maka dilakukan pengukuran jarak antara kedua tangan dengan menggunakan variable `float` yang ditulis pada baris dua puluh Sembilan dan tiga puluh.

4.2.3 Fungsi `Get Hand State`

```
1   float handDistance = Distance_From_To(posisiTanganKananX,
2   posisiTanganKananY, posisiTanganKiriX, posisiTanganKiriY);
3   if (!Waiting_Input && !Reset_Input)
4   {
5   if (handDistance > 0.3)
6   {
7   if (posisiTanganKananX > 0.3)
8   {
9   switch (body.HandRightState)
10  {
11  case HandState.Lasso:
12  Console.WriteLine("kanan lasso");
13  Start_Input("Kanan", "Lasso", posisiTanganKananX,
14  posisiTanganKananY);
15  break;
16  if (posisiTanganKiriX < -0.3)
17  {
18  switch (body.HandLeftState)
19  {
20  case HandState.Lasso:
21  Console.WriteLine("kiri lasso");
22  Start_Input("Kiri", "Lasso", posisiTanganKiriX,
23  posisiTanganKiriY);
24  break;
25  if (inputHand == "Kanan")
26  {
27  if (Get_Distance_From_Trigger_Position(posisiTanganKananX,
28  posisiTanganKananY))
29  {
30  Console.WriteLine("Get distance from trigger passed, right");
31  tblAction.Text = Show_Result(handState,
32  Input_Position(posisiTanganKananX, posisiTanganKananY),
33  inputHand);
34  }
35  switch (body.HandRightState)
36  {
37  case HandState.Closed:
38  tblAction.Text = Show_Result2();
39  break;
40  default:
41  break;
42  }
43  }
44  if (inputHand == "Kiri")
45  {
46  if (Get_Distance_From_Trigger_Position(posisiTanganKiriX,
47  posisiTanganKiriY))
```

```
48  {
49  Console.WriteLine("Get distance from trigger passed, left");
50  tblAction.Text = Show_Result(handState,
51  Input_Position(posisiTanganKiriX, posisiTanganKiriY),
52  inputHand);
53  }
54  switch (body.HandLeftState)
55  {
56  case HandState.Closed:
57  tblAction.Text = Show_Result2();
58  break;
59  default:
60  break;
```

Gambar 4.9 Kode untuk mendapatkan kondisi tangan Lasso

Gambar 4.10 digunakan untuk mendapatkan kondisi tangan lasso sebagai syarat *input* dimasukkan. Kode tersebut dijelaskan sebagai berikut :

1. Baris Satu hingga tiga digunakan untuk syarat dari variable hand distance.
2. Sebelum jarak antara kedua tangan mencapai 0.3 dari satu sama lain maka frame akan di loop hingga didapat jarak sesuai syarat.
3. Apabila jarak dari HandDistance lebih dari 0.3 maka akan dijalankan kode berikutnya sesuai dengan syarat variable `If` dari baris lima.
4. Jika hand distance sudah terpenuhi maka dijalankan baris ke tujuh atau enam belas
5. Apabila jarak variable posisisTanganKanan lebih dari 0.3 maka kode pada baris Sembilan hingga lima belas dijalankan.
6. Kode pada baris sembilan hingga lima belas akan membaca kondisi dari tangan kanan.
7. Apabila kondisi dari tangan kanan dibaca sebagai lasso maka tangan kanan akan dianggap sebagai *input*.
8. Apabila syarat pada baris enam belas terpenuhi maka kode baris delapan belas hingga dua puluh empat akan dijalankan.
9. Apabila kondisi dari tangan kiri dibaca sebagai lasso maka tangan kanan akan dianggap sebagai *input*.
10. Setelah Lasso terdeteksi dan terbaca maka akan dilakukan pengecekan sebagai trigger.
11. Apabila Lasso terbaca sebagai posisi pada kanan maka kode baris dua puluh lima akan dijalankan.
12. Setelah itu dilakukan cek apakah tangan telah keluar dari dari posisi trigger untuk dijadikan *input* yang ditulis pada baris dua puluh tujuh hingga empat puluh satu.
13. Apabila Lasso terbaca sebagai posisi pada kiri maka kode baris empat puluh empat akan dijalankan.
14. Setelah itu dilakukan cek apakah tangan telah keluar dari dari posisi trigger untuk dijadikan *input* yang ditulis pada baris empat puluh enam hingga enam puluh.

4.2.4 Fungsi `Get Hand` *`Input`*

```
1   string Input_Position(float inputX, float inputY)
2   {
3   string result = "?";
4   if (inputX < regTanganX)
5   {
6   if (inputY < regTanganY)
7   {
8   result = "Kiri Bawah";
9   }
10  else
11  {
12  result = "Kiri Atas";
13  }
14  }
15  else
16  {
17  if (inputY < regTanganY)
18  {
19  result = "Kanan Bawah";
20  }
21  else
22  {
23  result = "Kanan Atas";
24  }
25  }
```

Gambar 4.10 Kode untuk melambangkan posisi dari tangan.

Sebelum *Input* dibedakan menjadi tangan kanan dan kiri maka dibuat sebuah variabel yang bisa dipanggil untuk membedakan empat *input* utama. Berikut adalah penjelasan yang dilakukan untuk gambar 4.11 :

1. Pada baris satu dilakukan pemanggilan hasil X dan Y yang diwakili dengan variabel *Input*_Position.
2. *Input* utama dibagikan menjadi empat *input* utama.
3. Pada baris empat digunakan variabel `IF` sebagai syarat dari X untuk menunjukkan *input* yang akan digunakan.
4. Koordinat dianggap antara Kanan dari apabila hasil dari *input* ada diatas regTanganX yang didapat dari hasil pendeteksian lasso pertama.
5. Dan apabila hasilnya dibawah dari regTanganX maka hasil akan dijadikan kiri sesuai dengan perintah pada variabel else pada baris kelima belas.
6. Persyaratan berikutnya adalah melihat apabila nilai dari Y melebihi dari hasil regTanganY yang dideskripsikan oleh baris enam hingga dua belas.
7. Apabila hasil lebih besar dari regTanganY maka *input* dianggap “Kiri Atas”.
8. Jika hasil kurang dari regTanganY maka *input* dianggap “Kiri Bawah”.
9. Dari variabel `IF` diatas jika tidak ada kondisi yang terpenuhi maka akan dimasukkan lagi kedalam variabel `IF` selanjutnya.

10. Pada baris ketujuh belas ditulis apabila Y memiliki nilai lebih besar dari hasil dari regTanganY maka *input* menjadi “Kanan Atas”
11. Untuk kondisi lain maka akan dijadikan “Kanan Bawah”

Kode ini digunakan untuk menjadi referensi *input* agar tidak dilakukan pembuatan kode berkali kali.

4.2.4.1 Get Hand *Input*

```
1   if (handState == "Lasso")
2   {
3   switch (input)
4   {
5   case "Kanan Atas":
6   output = "Shortcut 1 " + inputHand;
7   Reset_Input_Flag();
8   break;
9   case "Kanan Bawah":
10  output = "Shortcut 2 " + inputHand;
11  Reset_Input_Flag();
12  break;
13  case "Kiri Atas":
14  output = "Shortcut 3 " + inputHand;
15  Reset_Input_Flag();
16  break;
17  case "Kiri Bawah":
18  output = "Shortcut 4 " + inputHand;
19  Reset_Input_Flag();
20  break;
21  }
22  }
```

Gambar 4.11 Kode untuk melakukan eksekusi dari shortcut.

Setelah kode untuk referensi *input* masuk maka dibuat kode untuk melakukan implementasi output. Berikut adalah penjelasan dari gambar 4.12:

1. Baris satu digunakan untuk memastikan handState sebagai Lasso.
2. Dari *input* maka dilakukan penggantian pada baris tiga.
3. Apabila pada frmae terdeteksi sebagai Kanan Atas maka akan dilakukan Shortcut 1 beserta inpt yang dimasukkan.
4. Apabila pada frame terdeteksi sebagai Kanan Bawah maka akan dilakukan Shortcut 2 beserta inpt yang dimasukkan.
5. Apabila pada frame terdeteksi sebagai Kiri Atas maka akan dilakukan Shortcut 3 beserta inpt yang dimasukkan.
6. Apabila pada frame terdeteksi sebagai Kiri bawah maka akan dilakukan Shortcut 4 beserta inpt yang dimasukkan.

4.3 Evaluasi

Dari kode dan penelitian yang dilakukan maka dievaluasi kembali apakah percobaan telah bekerja dengan benar. Beberapa test yang dilakukan adalah sebagai berikut :

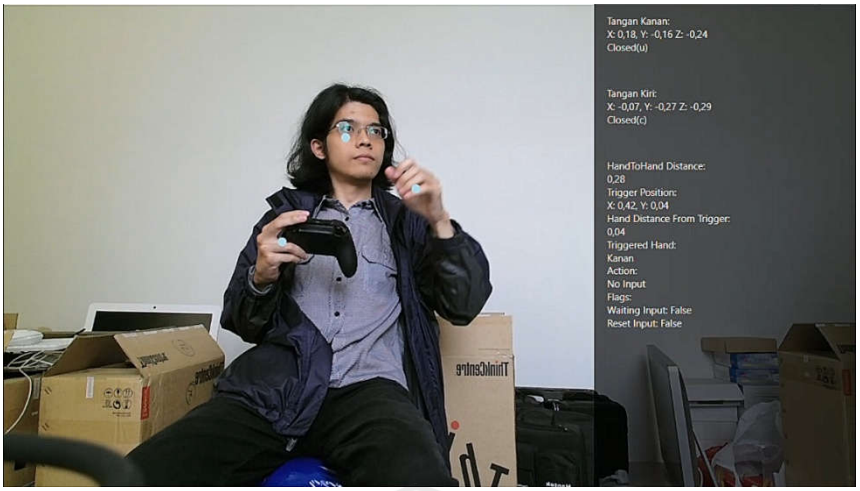
1. Evaluasi Body.
2. Evaluasi Get Hand State.
3. Evaluasi Get Lasso Position.
4. Evaluasi Get *Input*.

4.3.1 Evaluasi Body

Pengujian ini dimaksudkan untuk memastikan apakah data yang didapatkan dari *kinect* dapat mendeteksi bentuk tubuh untuk mengetahui posisi dari tangan dan kepala. Fungsi yang diuji pada subbab ini adalah Get Body, Get Hand Position yang ditunjukkan pada gambar 4.13 :

Gambar 4.12 Pengujian deteksi posisi kepala dan tangan.

Sesuai pada gambar data yang dikeluarkan dari *kinect* mendeteksi posisi dari kepala dan kedua tangan pengguna. Posisi dapat diketahui dari beberapa titik yang ada pada gambar. Untuk memperjelas data yang didapat maka dibuat debug yang di tunjukkan seperti berikut :



Gambar 4.13 Bentuk debug untuk menunjukkan hasil dari *kinect*.

Hasil dari data yang didapatkan untuk pembacaan tubuh yang diperlukan untuk penelitian ini adalah data posisi dari tangan kanan, Tangan kiri, dan kepala. Dari gambar 4.14 dikeluakan posisi dalam bentuk kordinat X, Y, dan Z, posisi dari Trigger, *input*, jarak antar kedua tangan.

Tabel 4.1 Tabel pengujian `Body`

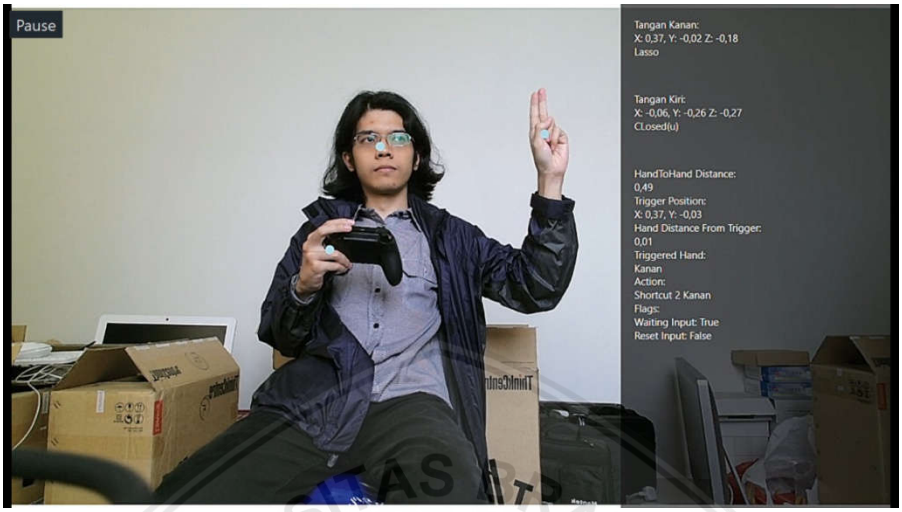
No	Fungsi Yang DiUji	Hasil	Deskripsi
1	Posisi dari Kepala	Benar	Posisi dari kepala pengguna pada gambar yang ditangkap *kinect*.
2	`DrawPoint.head`	Benar	Simbol titik menunjukkan posisi kepala pengguna saat itu
3	Posisi dari Tangan Kanan	Benar	Posisi dari tangan kanan pengguna pada gambar yang ditangkap *kinect*.
4	`DrawPoint.handRight`	Benar	Simbol titik menunjukkan posisi tangan kanan pengguna saat itu
5	Posisi dari Tangan Kiri	Benar	Posisi dari tangan kiri pengguna pada gambar yang ditangkap *kinect*.
6	`DrawPoint.handLeft`	Benar	Simbol titik menunjukkan posisi tangan kiri pengguna saat itu.

4.3.2 Evaluasi `Get State`

Pengujian ini dilakukan untuk mendeteksi apakah kode dapat mendeteksi data bentuk dari tangan yang yang dibentuk. Dari library yang dimiliki oleh *kinect* terdapat tiga kondisi yaitu Open, Close, Lasso. Dibawah ini adalah test yang dilakukan untuk mengetahui status dari tangan:

4.3.2.1 State Lasso

Evaluasi berikut digunakan untuk mendeteksi apakah bentuk tangan Lasso terbaca.



Gambar 4.14 Pengujian Status lasso.

Sesuai dengan Gambar 4.15 pada debug yang ditampilkan oleh *User Interface* dibawah tampilan kordinat X, Y, Z menampilkan status dari tangan sebagai Lasso.

Tabel 4.2 Tabel pengujian status `Lasso`

No	Fungsi Yang DiUji	Hasil	Deskripsi
1	Status `Lasso` pada tangan kanan	Benar	Fungsi untuk melakukan pengecekan bahwa tangan kanan melakukan kondisi `lasso` pada saat itu.
2	Tampilan pada debug untuk `Lasso`	Benar	Tulisan status yang dikeluarkan oleh debug apakah sama dengan status yang pengguna lakukan.

4.3.2.2 Fungsi `State Closed`

Evaluasi berikut digunakan untuk mendeteksi apakah bentuk tangan Closed terbaca.

Gambar 4.15 Pengujian Status Closed.

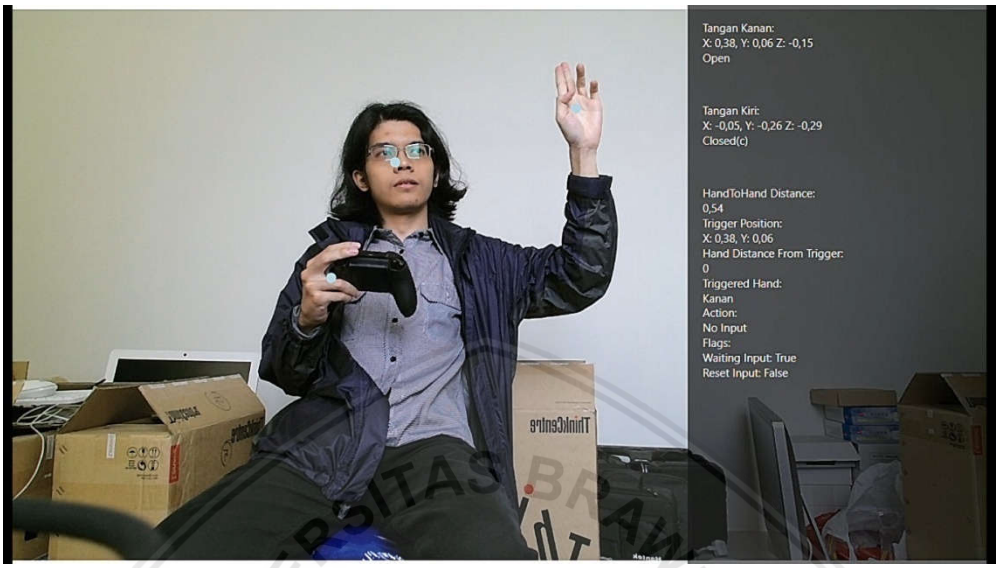
Sesuai dengan Gambar 4.16 pada debug yang ditampilkan oleh *User Interface* dibawah tampilan kordinat X, Y, Z menampilkan status dari tangan sebagai Closed.

Tabel 4.3 Pengujian Status `Closed`

No	Fungsi Yang DiUji	Hasil	Deskripsi
1	Status tangan Close	Benar	Aksi dari pengguna melakukan aksi Close pada *kinect*.
2	Hasil yang dituliskan	Benar	Tulisan status yang dikeluarkan oleh debug apakah sama dengan status yang pengguna lakukan.

4.3.2.3 Fungsi `State Open`

Evaluasi berikut digunakan untuk mendeteksi apakah bentuk tangan *Open* terbaca.



Gambar 4.16 Pengujian Status Open.

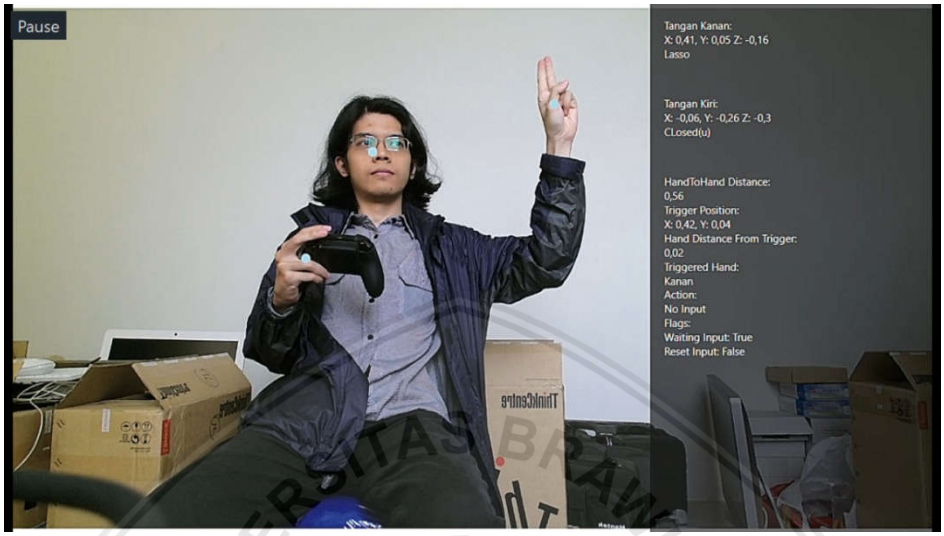
Sesuai dengan Gambar 4.17 pada debug yang ditampilkan oleh *User Interface* dibawah tampilan kordinat X, Y, Z menampilkan status dari tangan sebagai Open.

Tabel 4.4 Pengujian Status Tangan `Open`

No	Fungsi Yang DiUji	Hasil	Deskripsi
1	Status `Open` pada tangan kanan	Benar	Fungsi untuk melakukan pengecekan bahwa tangan kanan melakukan kondisi `Open` pada saat itu.
2	Tampilan pada debug untuk `Open`	Benar	Tulisan status yang dikeluarkan oleh debug apakah sama dengan status yang pengguna lakukan.

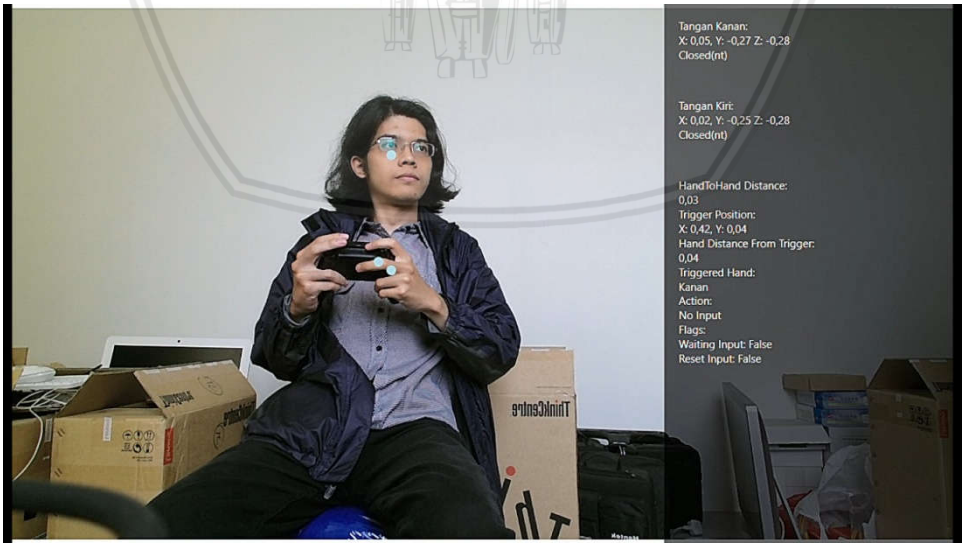
4.3.3 Evaluasi `Lasso Position`

Berikut adalah evaluasi yang dilakukan untuk melakukan Test pada posisi lasso yang digunakan untuk melakukan *input*. Hasil dari evaluasi dari subbab ini ditunjukkan dengan gambar 4.18.



Gambar 4.17 Posisi dari Trigger.

Dari gambar 4.18 adalah data kordinat dari trigger dituliskan pada debug dengan Trigger Position yang ditunjukkan posisi X dan Y. Data yang digunakan hanya data X dan Y dikarenakan penelitian hanya memerlukan kordinat dua dimensi yang dibaca. Untuk menguji apakah trigger tidak terpengaruh meskipun pengguna melakukan gerakan setelah trigger mendapatkan posisinya maka dilakukan gerakan kembali ke tangan.



Gambar 4.18 Posisi trigger tidak berubah.

Pada gambar 4.19 menggambarkan bahwa posisi X dan Y tidak berubah meskipun tangan telah digerakan. Posisi dari trigger teralhir akan tersimpan hingga *kinect* mendapatkan data trigger baru.

Tabel 4.5 Pengujian fungsi `Trigger`

No	Fungsi Yang DiUji	Hasil	Deskripsi
1	Status `Lasso` pada tangan kanan	Benar	Fungsi untuk melakukan pengecekan bahwa tangan kanan melakukan kondisi `lasso` pada saat itu.
2	Posisi `Trigger`	Benar	Melakukan cek pada tangan untuk dijadikan posisi trigger apabila tangan berubah menjadi `Lasso` untuk pertama kali deteksi terjadi.
3	Kondisi tetap Trigger	Benar	Fungsi ini dilakukan untuk mengetahui apakah posisi dari trigger berubah atau tidak dari trigger pertama dilakukan.

4.3.4 Evaluasi `Get` *`Input`*

Evaluasi ini untuk menguji bagaimana *input* yang telah didesain telah bekerja sesuai dengan desain. Beberapa *input* yang dirancang sebanyak empat *input* dari masing - masing tangan. Berikut adalah daftar dari *Input* :

1. *Input* Tangan Kanan
    a. Shortcut 1 Kanan
    b. Shortcut 2 Kanan
    c. Shortcut 3 Kanan
    d. Shortcut 4 Kanan
2. *Input* Tangan Kiri
    a. Shortcut 1 Kiri
    b. Shortcut 2 Kiri
    c. Shortcut 3 Kiri
    d. Shortcut 4 Kiri

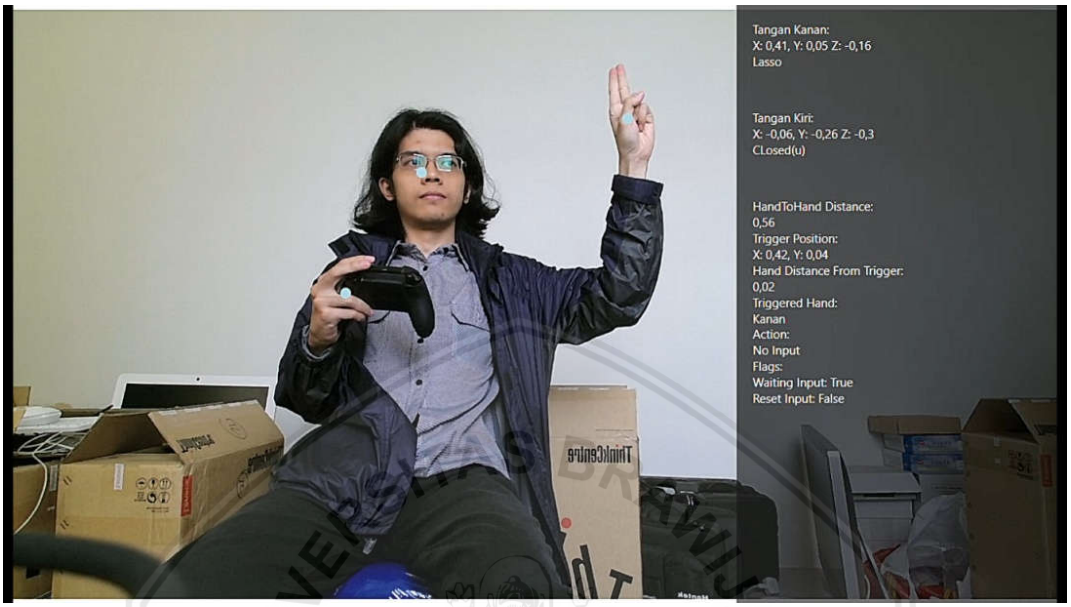
Daftar tersebut diuji untuk memastikan apakah data yang dimasukkan oleh pengguna cocok dengan data yang dikeluarkan oleh program.

4.3.4.1 *Input* Tangan Kanan

Data dari tangan kanan terdapat empat *input* yaitu shortcut 1 kanan, shortcut 2 kanan, shortcut 3 kanan, dan shortcut 4 kanan. Posisi tangan dilakukan bentuk trigger sebelum dilakukan *input* kemudian dilakukan gerakan pada posisi diluar jangkauan active yaitu 0.2 dari posisi trigger.

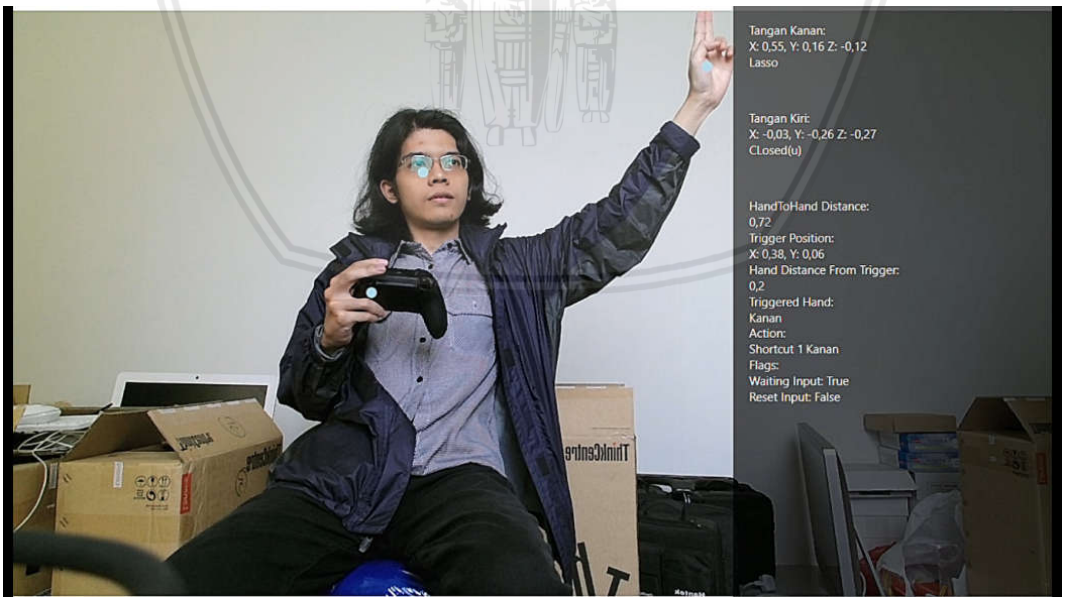
1. Shortcut 1 Kanan

Sebelum *Input* terjadi maka tangan diposisikan pada kondisi trigger yang ditunjukkan pada gambar 4.20.



Gambar 4.19 Posisi tangan pada bentuk lasso untuk posisi trigger.

Setelah posisi dari Trigger terbaca maka dilakukan gerakan yang melebihi posisi X dan Y bernilai diatas 0. Gambar 4.21 adalah ilustrasi untuk melakukan *input*.



Gambar 4.20 Posisi Tangan setelah melakukan Shortcut 1 Kanan.

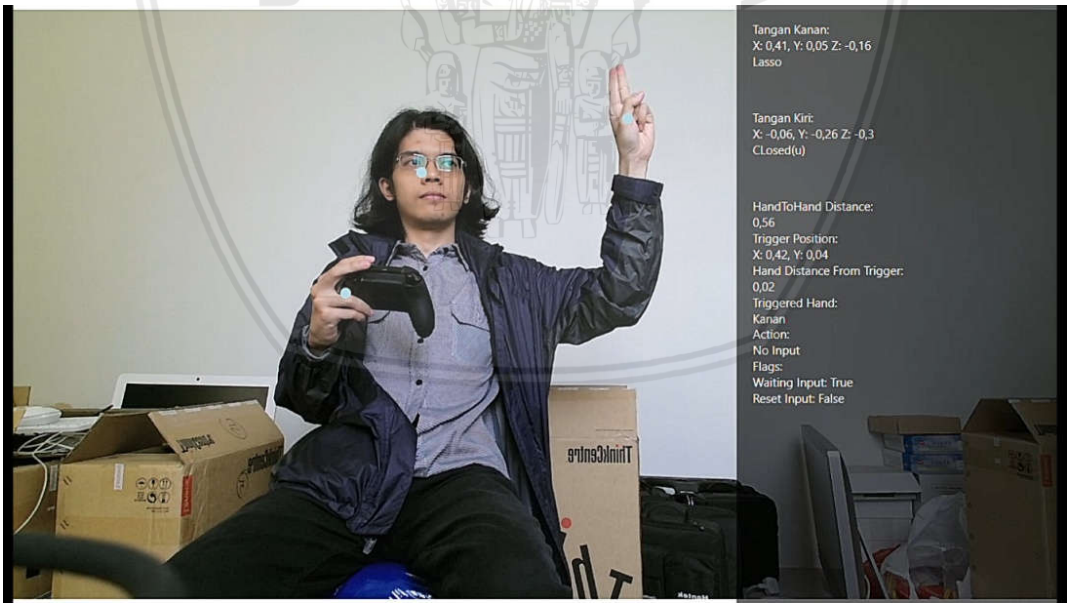
Dari gambar pengujian pada gambar 4.21 didapatkan beberapa data pengujian yang dijelaskan pada table dibawah ini.

Tabel 4.6 Pengujian untuk *input* Shortcut 1 Kanan

No	Fungsi Yang DiUji	Hasil	Deskripsi
1	Hasil *Input*	Benar	Fungsi untuk melakukan pengecekan bahwa tangan kanan melakukan kondisi `lasso` pada saat itu.
2	Posisi `Trigger`	Benar	Melakukan cek pada tangan untuk dijadikan posisi trigger apabila tangan berubah menjadi `Lasso` untuk pertama kali deteksi terjadi.
3	Kondisi tetap Trigger	Benar	Fungsi ini dilakukan untuk mengetahui apakah posisi dari trigger berubah atau tidak dari trigger pertama dilakukan.
4	Kondisi tangan *input*	Benar	Pengujian dari apakah syarat untuk melakukan shortcut 1 kanan terjadi.

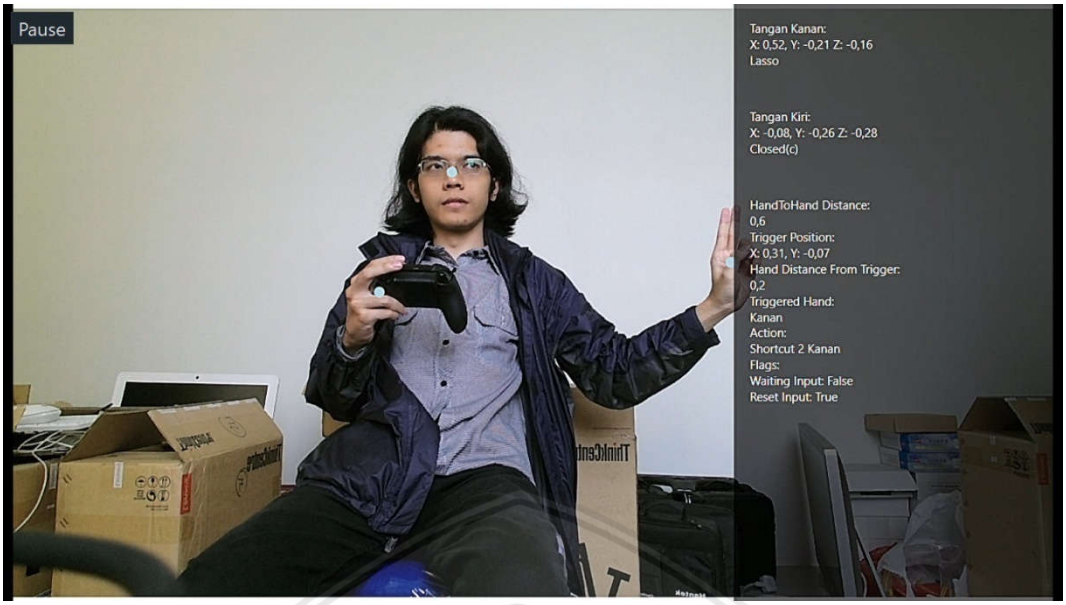
2. Shortcut 2 Kanan

Sebelum *Input* terjadi maka tangan diposisikan pada kondisi trigger yang ditunjukkan pada gambar 4.22.



Gambar 4.21 Posisi tangan pada bentuk lasso untuk posisi trigger.

Setelah posisi dari Trigger terbaca maka dilakukan gerakan yang melebihi posisi X diatas 0 dan posisi Y dibawah -0. Gambar 4.23 adalah ilustrasi untuk melakukan *input*.



Gambar 4.22 Posisi Tangan setelah melakukan Shortcut 2 Kanan.

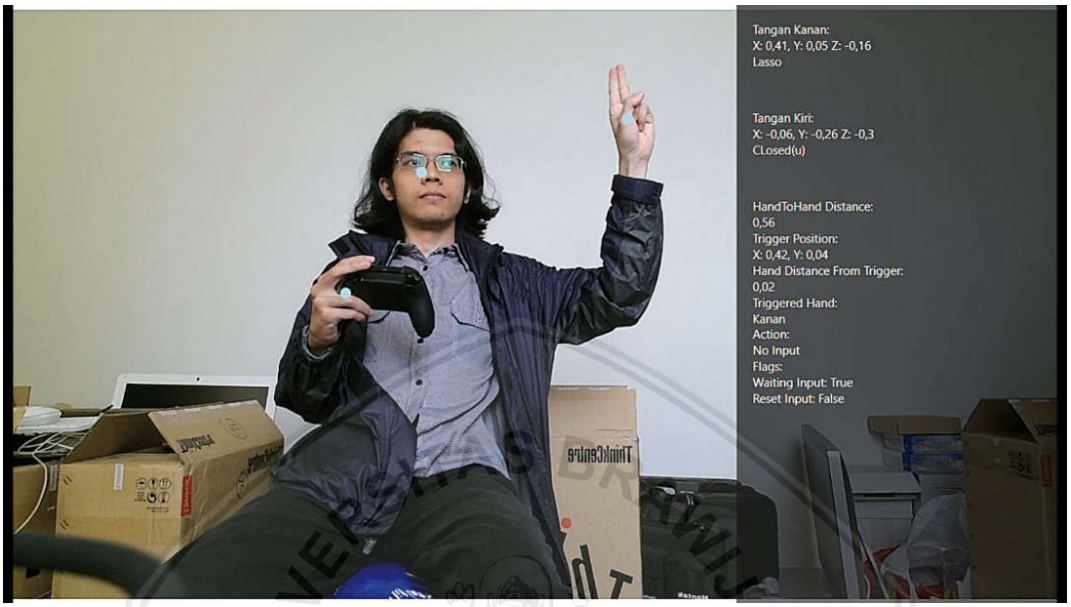
Dari gambar 4.23 pengujian didapatkan beberapa data pengujian yang dijelaskan pada table dibawah ini.

Tabel 4.7 Pengujian untuk *input* Shortcut 2 Kanan

No	Fungsi Yang DiUji	Hasil	Deskripsi
1	Hasil *Input*	Benar	Fungsi untuk melakukan pengecekan bahwa tangan kanan melakukan kondisi `lasso` pada saat itu.
2	Posisi `Trigger`	Benar	Melakukan cek pada tangan untuk dijadikan posisi trigger apabila tangan berubah menjadi `Lasso` untuk pertama kali deteksi terjadi.
3	Kondisi tetap Trigger	Benar	Fungsi ini dilakukan untuk mengetahui apakah posisi dari trigger berubah atau tidak dari trigger pertama dilakukan.
4	Kondisi tangan *input*	Benar	Pengujian dari apakah syarat untuk melakukan Shortcut 2 Kanan terjadi.

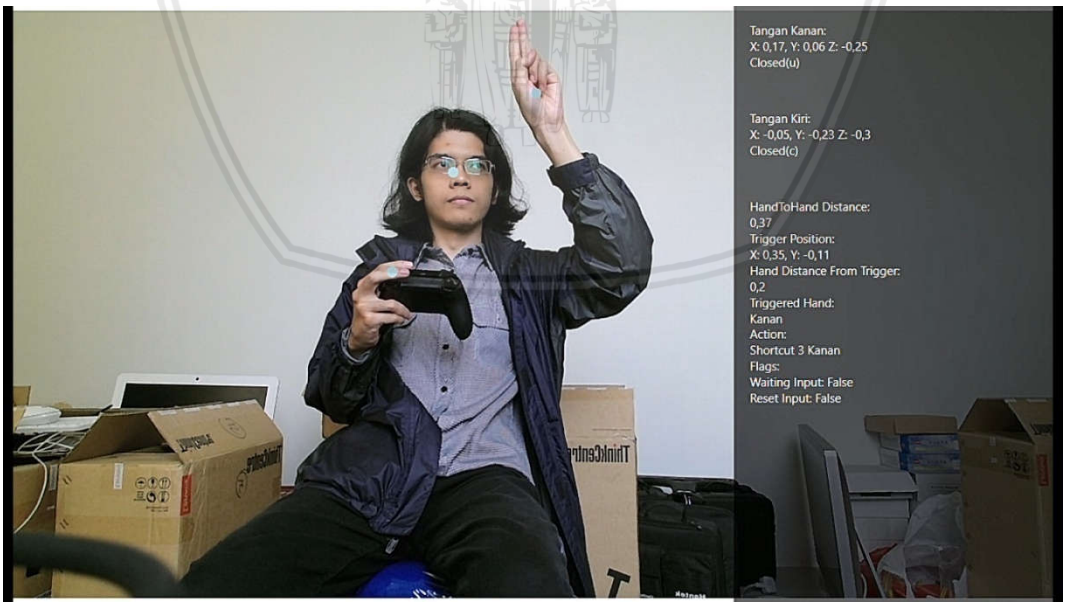
3. Shortcut 3 Kanan

Sebelum *Input* terjadi maka tangan diposisikan pada kondisi trigger yang ditunjukkan pada gambar 4.24.



Gambar 4.23 Posisi tangan pada bentuk lasso untuk posisi trigger.

Setelah posisi dari Trigger terbaca maka dilakukan gerakan yang melebihi posisi X dibawah -0 dan posisi Y diatas 0. Gambar 4.25 adalah ilustrasi untuk melakukan *input*.



Gambar 4.24 Posisi Tangan setelah melakukan Shortcut 3 Kanan.

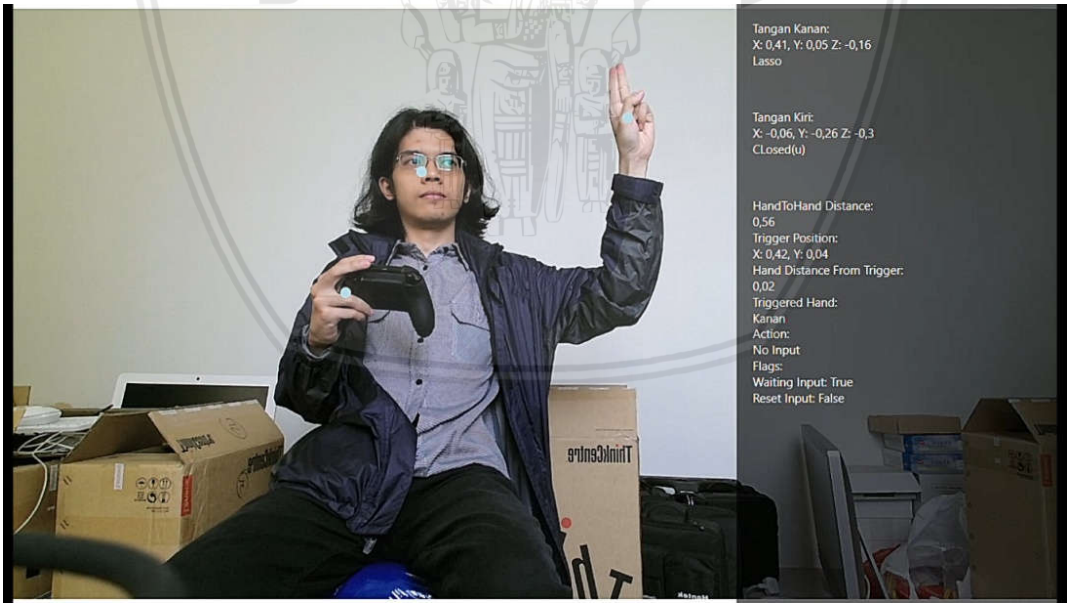
Dari gambar 4.25 pengujian didapatkan beberapa data pengujian yang dijelaskan pada table dibawah ini.

Tabel 4.8 Pengujian untuk *input* Shortcut 3 Kanan

No	Fungsi Yang DiUji	Hasil	Deskripsi
1	Hasil *Input*	Benar	Fungsi untuk melakukan pengecekan bahwa tangan kanan melakukan kondisi `lasso` pada saat itu.
2	Posisi `Trigger`	Benar	Melakukan cek pada tangan untuk dijadikan posisi trigger apabila tangan berubah menjadi `Lasso` untuk pertama kali deteksi terjadi.
3	Kondisi tetap Trigger	Benar	Fungsi ini dilakukan untuk mengetahui apakah posisi dari trigger berubah atau tidak dari trigger pertama dilakukan.
4	Kondisi tangan *input*	Benar	Pengujian dari apakah syarat untuk melakukan Shortcut 3 Kanan terjadi.

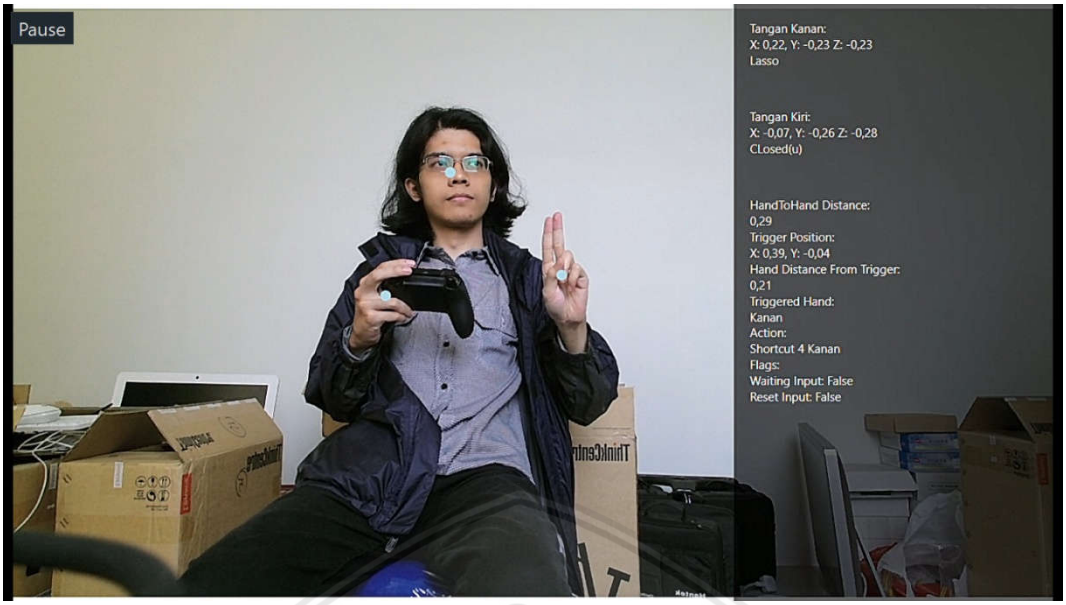
4. Shortcut 4 Kanan

Sebelum *Input* terjadi maka tangan diposisikan pada kondisi trigger yang ditunjukkan pada gambar 4.26.



Gambar 4.25 Posisi tangan pada bentuk lasso untuk posisi trigger.

Setelah posisi dari Trigger terbaca maka dilakukan gerakan yang melebihi posisi X dibawah -0 dan posisi Y dibawah -0. Gambar 4.27 adalah ilustrasi untuk melakukan *input*.



Gambar 4.26 Posisi Tangan setelah melakukan Shortcut 4 Kanan.

Dari gambar 4.27 pengujian didapatkan beberapa data pengujian yang dijelaskan pada table dibawah ini.

Tabel 4.9 Pengujian untuk *input* Shortcut 4 Kanan

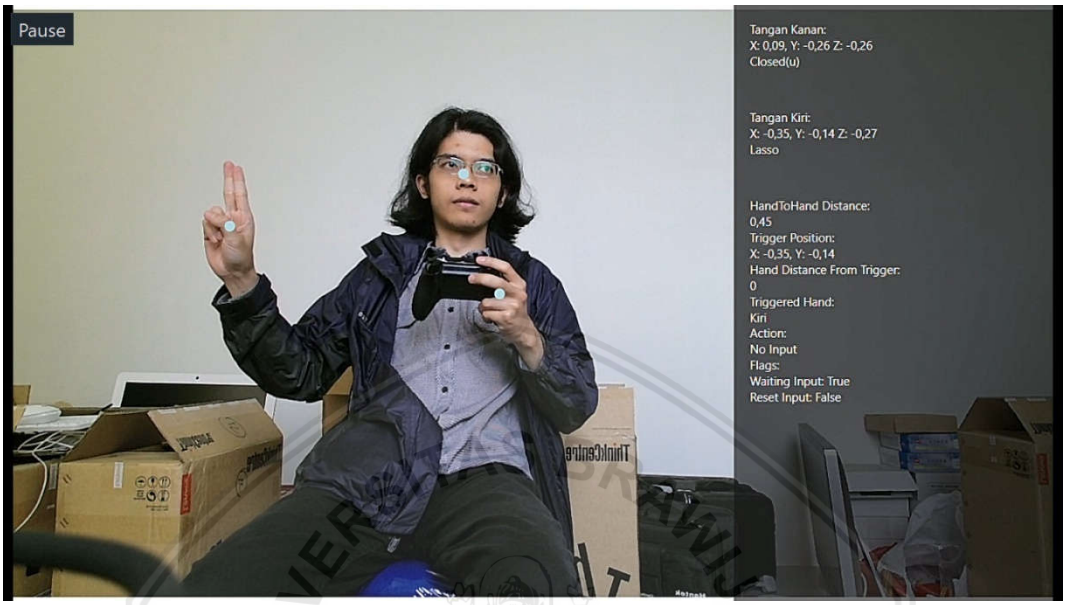
No	Fungsi Yang DiUji	Hasil	Deskripsi
1	Hasil *Input*	Benar	Fungsi untuk melakukan pengecekan bahwa tangan kanan melakukan kondisi `lasso` pada saat itu.
2	Posisi `Trigger`	Benar	Melakukan cek pada tangan untuk dijadikan posisi trigger apabila tangan berubah menjadi `Lasso` untuk pertama kali deteksi terjadi.
3	Kondisi tetap Trigger	Benar	Fungsi ini dilakukan untuk mengetahui apakah posisi dari trigger berubah atau tidak dari trigger pertama dilakukan.
4	Kondisi tangan *input*	Benar	Pengujian dari apakah syarat untuk melakukan Shortcut 4 Kanan terjadi.

4.3.4.2 *Input* Tangan Kiri

Data dari tangan kiri juag terdapat empat *input* yaitu shortcut 1 kiri, shortcut 2 kiri, shortcut 3 kiri, dan shortcut 4 kiri. Posisi tangan dilakukan bentuk trigger sebelum dilakukan *input* kemudian dilakukan gerakan pada posisi diluar jangkauan active yaitu 0 dari posisi trigger.

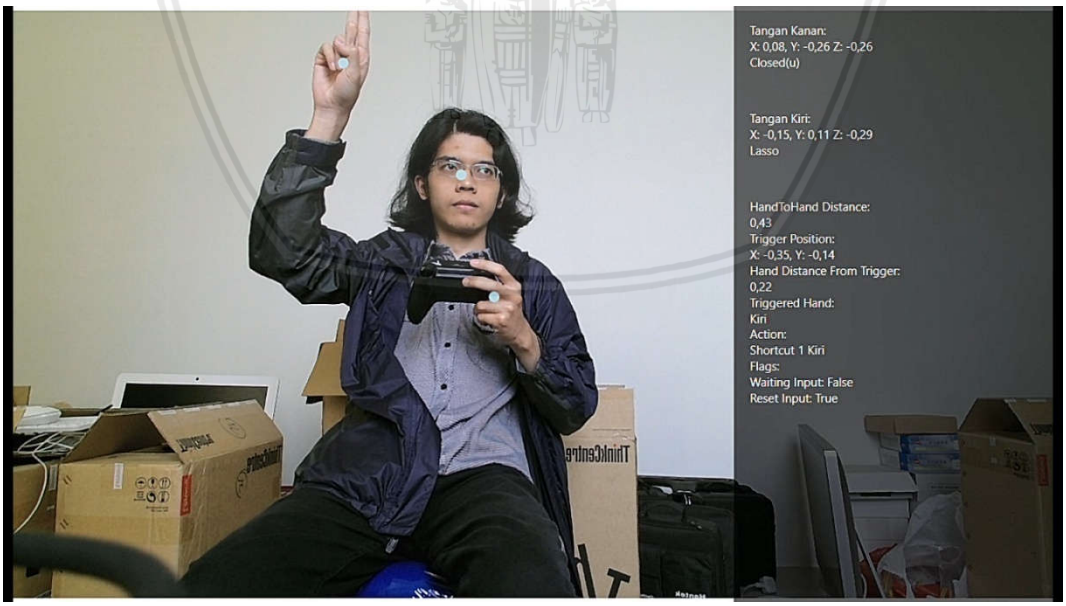
1. Shortcut 1 Kiri

Sebelum *Input* terjadi maka tangan diposisikan pada kondisi trigger yang ditunjukkan pada gambar 4.28.



Gambar 4.27 Posisi tangan pada bentuk lasso untuk posisi trigger.

Setelah posisi dari Trigger terbaca maka dilakukan gerakan yang melebihi posisi X diatas 0 dan Y diatas 0. Gambar 4.29 adalah ilustrasi untuk melakukan *input*.



Gambar 4.28 Posisi Tangan setelah melakukan Shortcut 1 Kiri.

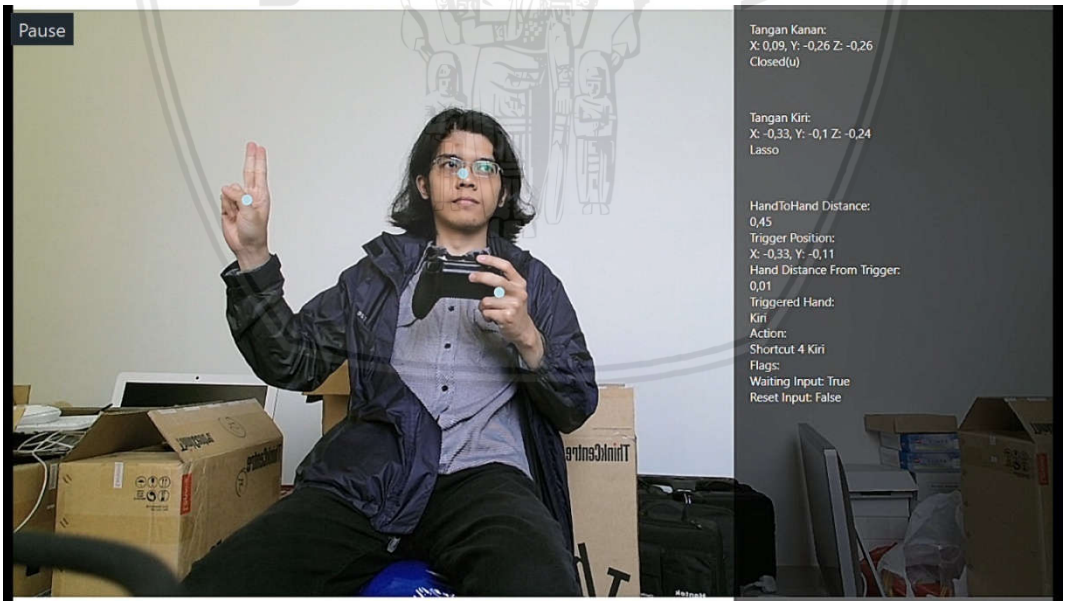
Dari gambar 4.29 pengujian didapatkan beberapa data pengujian yang dijelaskan pada table dibawah ini.

Tabel 4.10 Pengujian untuk *input* Shortcut 1 Kiri

No	Fungsi Yang DiUji	Hasil	Deskripsi
1	Hasil *Input*	Benar	Fungsi untuk melakukan pengecekan bahwa tangan kiri melakukan kondisi `lasso` pada saat itu.
2	Posisi `Trigger`	Benar	Melakukan cek pada tangan untuk dijadikan posisi trigger apabila tangan berubah menjadi `Lasso` untuk pertama kali deteksi terjadi.
3	Kondisi tetap Trigger	Benar	Fungsi ini dilakukan untuk mengetahui apakah posisi dari trigger berubah atau tidak dari trigger pertama dilakukan.
4	Kondisi tangan *input*	Benar	Pengujian dari apakah syarat untuk melakukan shortcut 1 kiri terjadi.

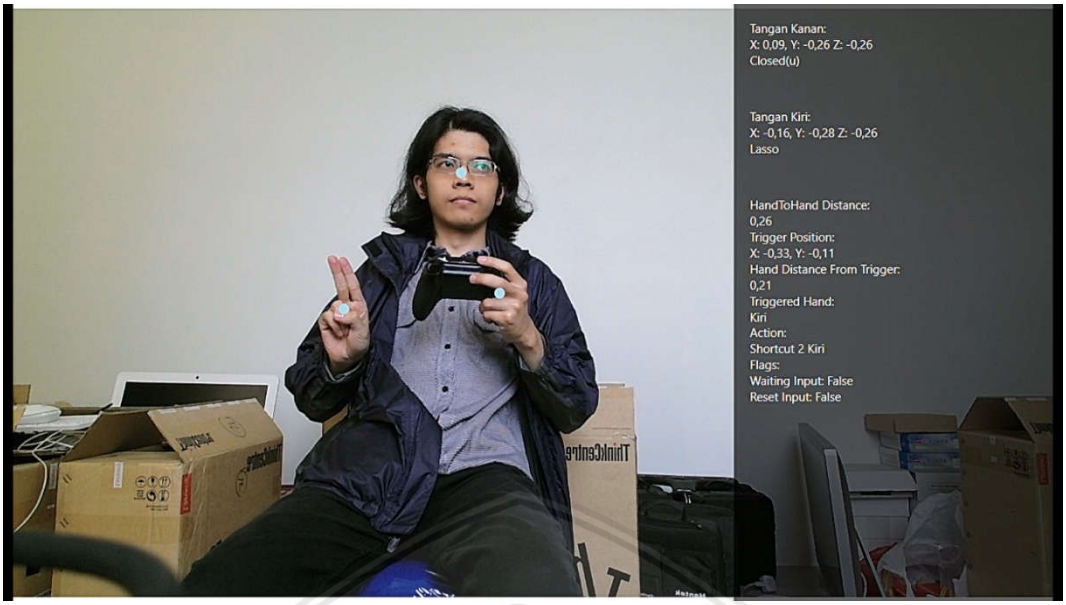
2. Shortcut 2 Kiri

Sebelum *Input* terjadi maka tangan diposisikan pada kondisi trigger yang ditunjukkan pada gambar 4.30.



Gambar 4.29 Posisi tangan pada bentuk lasso untuk posisi trigger.

Setelah posisi dari Trigger terbaca maka dilakukan gerakan yang melebihi posisi posisi X diatas 0 dan Y dibawah -0. Gambar 4.31 adalah ilustrasi untuk melakukan *input*.



Gambar 4.30 Posisi Tangan setelah melakukan Shortcut 2 Kiri.

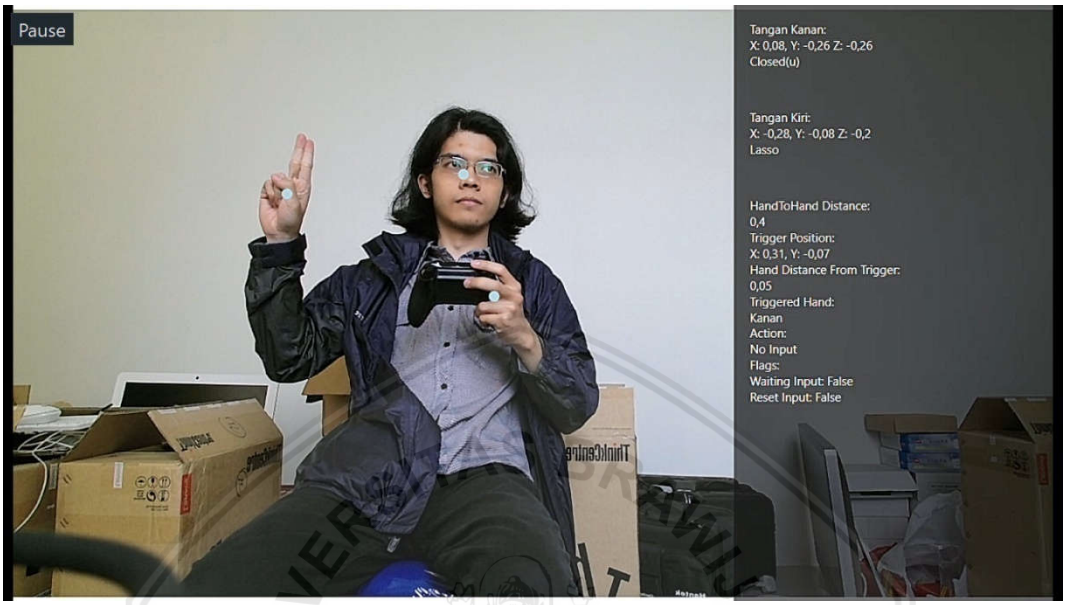
Dari gambar 4.31 pengujian didapatkan beberapa data pengujian yang dijelaskan pada table dibawah ini.

Tabel 4.11 Pengujian untuk *input* Shortcut 2 Kiri.

No	Fungsi Yang DiUji	Hasil	Deskripsi
1	Hasil *Input*	Benar	Fungsi untuk melakukan pengecekan bahwa tangan kiri melakukan kondisi `lasso` pada saat itu.
2	Posisi `Trigger`	Benar	Melakukan cek pada tangan untuk dijadikan posisi trigger apabila tangan berubah menjadi `Lasso` untuk pertama kali deteksi terjadi.
3	Kondisi tetap Trigger	Benar	Fungsi ini dilakukan untuk mengetahui apakah posisi dari trigger berubah atau tidak dari trigger pertama dilakukan.
4	Kondisi tangan *input*	Benar	Pengujian dari apakah syarat untuk melakukan Shortcut 2 Kiri terjadi.

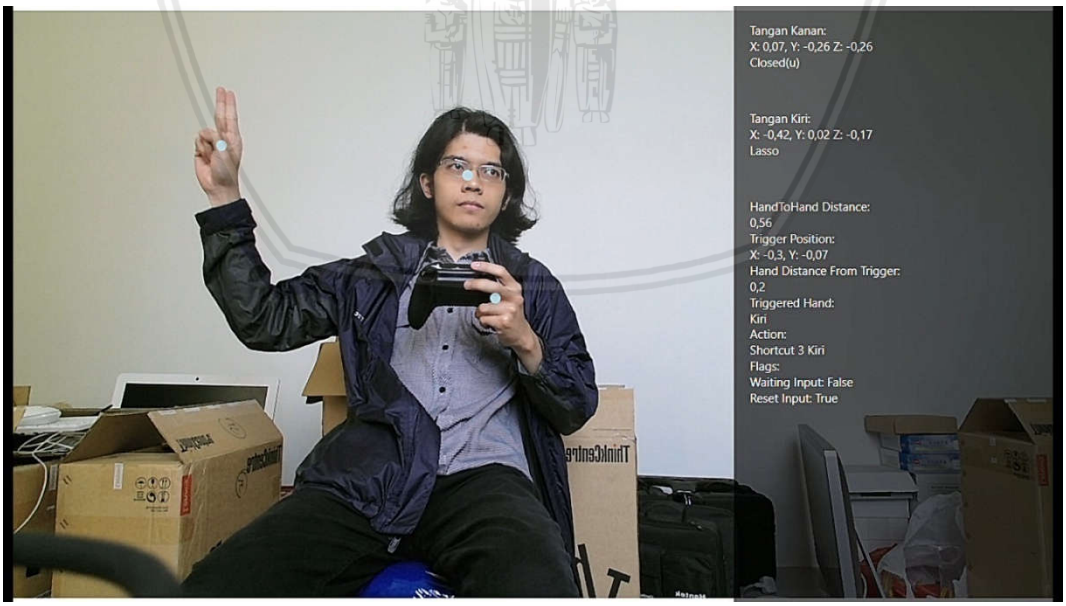
3. Shortcut 3 Kiri

Sebelum *Input* terjadi maka tangan diposisikan pada kondisi trigger yang ditunjukkan pada gambar 4.32.



Gambar 4.31 Posisi tangan pada bentuk lasso untuk posisi trigger.

Setelah posisi dari Trigger terbaca maka dilakukan gerakan yang melebihi posisi X dibawah 0 dan posisi Y diatas 0. Gambar 4.33 adalah ilustrasi untuk melakukan *input*.



Gambar 4.32 Posisi Tangan setelah melakukan Shortcut 3 Kiri.

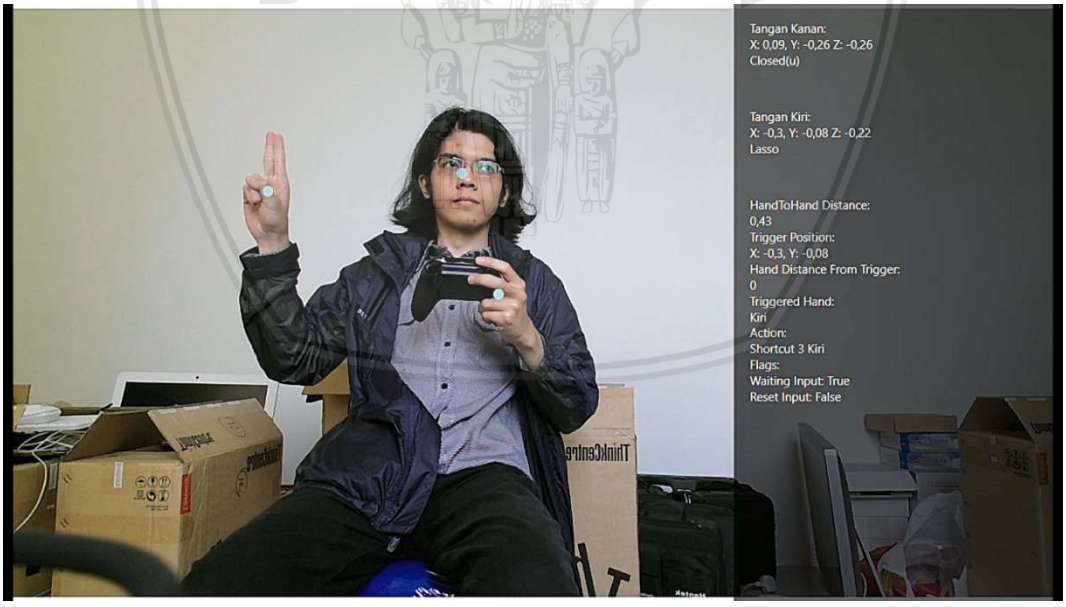
Dari gambar 4.33 pengujian didapatkan beberapa data pengujian yang dijelaskan pada table dibawah ini.

Tabel 4.12 Pengujian untuk *input* Shortcut 3 Kiri

No	Fungsi Yang DiUji	Hasil	Deskripsi
1	Hasil *Input*	Benar	Fungsi untuk melakukan pengecekan bahwa tangan kiri melakukan kondisi `lasso` pada saat itu.
2	Posisi `Trigger`	Benar	Melakukan cek pada tangan untuk dijadikan posisi trigger apabila tangan berubah menjadi `Lasso` untuk pertama kali deteksi terjadi.
3	Kondisi tetap Trigger	Benar	Fungsi ini dilakukan untuk mengetahui apakah posisi dari trigger berubah atau tidak dari trigger pertama dilakukan.
4	Kondisi tangan *input*	Benar	Pengujian dari apakah syarat untuk melakukan Shortcut 3 Kiri terjadi.

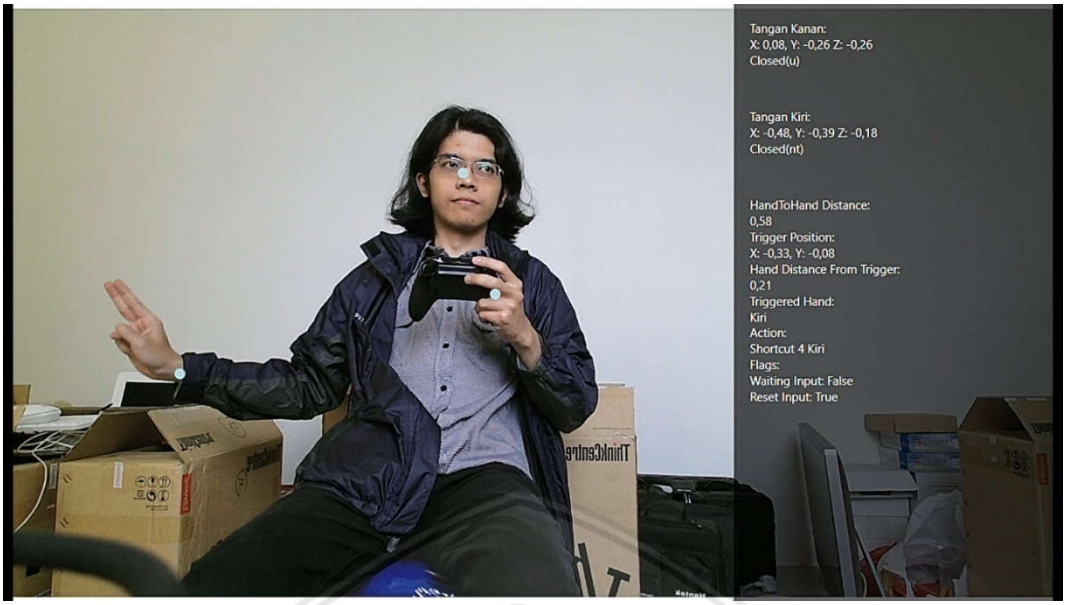
4. Shortcut 4 Kiri

Sebelum *Input* terjadi maka tangan diposisikan pada kondisi trigger yang ditunjukkan pada gambar 4.34.



Gambar 4.33 Posisi tangan pada bentuk lasso untuk posisi trigger.

Setelah posisi dari Trigger terbaca maka dilakukan gerakan yang melebihi posisi X dibawah 0 dan posisi Y dibawah 0. Gambar 4.35 adalah ilustrasi untuk melakukan *input*.



Gambar 4.34 Posisi Tangan setelah melakukan Shortcut 4 Kiri.

Dari gambar 4.35 pengujian didapatkan beberapa data pengujian yang dijelaskan pada table dibawah ini.

Tabel 4.13 Pengujian untuk *input* Shortcut 4 Kiri

No	Fungsi Yang DiUji	Hasil	Deskripsi
1	Hasil *Input*	Benar	Fungsi untuk melakukan pengecekan bahwa tangan iri melakukan kondisi `lasso` pada saat itu.
2	Posisi `Trigger`	Benar	Melakukan cek pada tangan untuk dijadikan posisi trigger apabila tangan berubah menjadi `Lasso` untuk pertama kali deteksi terjadi.
3	Kondisi tetap Trigger	Benar	Fungsi ini dilakukan untuk mengetahui apakah posisi dari trigger berubah atau tidak dari trigger pertama dilakukan.
4	Kondisi tangan *input*	Benar	Pengujian dari apakah syarat untuk melakukan Shortcut 4 kiri terjadi.

BAB 5 PENUTUPAN

Bab penutup membahas tentang kesimpulan dan saran yang diperoleh dari penelitian Penggabungan *Natural User Interface* dengan *Joypad* pada *Role Play Video Game.*

5.1 Kesimpulan

Berdasarkan pada pengerjaan dan pengujian yang telah dilakukan ditarik beberapa kesimpulan sebagai berikut :

1. Untuk mengatasi permasalahan kekurangan *input* pada Joypad digunakan *Natural User Interface* yang ditambahkan.
2. Pengaruh menambahkan *input macro* pada permainan adalah semakin cepatnya akses pada skill tambahan yang tidak dapat ditampung oleh Joypad.

5.2 Saran

Berikut adalah saran yang dapat digunakan untuk melanjutkan penelitian Penggabungan *Natural User Interface* dengan *Joypad* pada *Role Playing Video Game* ini :

1. Gesture dapat dikembangkan lagi dengan menambahkan beberapa bentuk tangan yang dapat dibuat dari awal.
2. *Macro* dapat dikembangkan agar tidak terjadi perubahan secara

DAFTAR PUSTAKA

Adams, E. & Rollings, A., 2007. Fundamental of Game Design. Dalam: *Game design and development.* California: Pearson Prentice Hall, p. 669. [Diakses 17 juli 2018].

Microsoft., 2008. *Visual Studio.* [Online] tersedia di: <https://visualstudio.microsoft.com/> [Diakses 26 09 2018].

Oxford., 2004. *https://en.oxforddictionaries.com/definition/joypad.* [Online] tersedia di: <https://en.oxforddictionaries.com/definition/joypad> [Diakses 17 juli 2018].

Anon., t.thn. *Unite User eXperience.* [Online] tersedia di: https://uniteux.com/blog/apa-itu-natural-user-interface [Diakses 17 juli 2018].

E.H. Shortliffe, M., 2017. *Elsevier.* [Online] tersedia di: <https://www.sciencedirect.com/science/article/pii/S1532046417301656> [Diakses 20 September 2018].

Lu, W., 2003. *Stanford University.* [Online] tersedia di: <web.stanford.edu> [Diakses 12 08 2018].

Wilson, Jeffrey L., 2010. Microsoft Kinect for Xbox 360. [Online] tersedia di: <https://www.pcmag.com/review/256482/microsoft-kinect-for-xbox-360> [Diakses 12 08 2018].

Mortensen, D., 2018. *Interaction Design.* [Online] tersedia di: <https://www.interaction-design.org/literature/article/natural-user-interfaces-what-are-they-and-how-do-you-design-user-interfaces-that-feel-natural> [Diakses 18 Oktober 2018].

Oxagile, 2014. *Oxagile.* [Online] tersedia di: <https://www.oxagile.com/company/blog/the-waterfall-model/> [Diakses 18 Oktober 2018].

Pterneas, V., 2014. *Implementing Kinect gestures.* [Online] tersedia di: https://pterneas.com/2014/01/27/implementing-kinect-gestures/ [Diakses 12 November 2018].

Square Enix, 2010. [Online] tersedia di: < *https://www.finalfantasyxiv.com/*> [Diakses 17 Juli 2018].

Yuhai, L., Jing, L. & Zhaojie, J., 2016. Data Fusion-based Real-Time Hand Gesture Recognition with Kinect V2. pp. 307-310. [Diakses 17 juli 2018].