Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 3, No. 9, September 2019, hlm. 9105-9113 http://j-ptiik.ub.ac.id

Penggunaan *Flag* untuk Mengurangi Permasalahan *Input* yang Tidak Disengaja pada *Controller* yang Menggunakan Kinect

Viqi Hanada[1], Wibisono Sukmo Wardhono[2], Muhammad Aminul Akbar[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: 125150101111001@ub.ac.id [1], wibiwardhono @ub.ac.id[2], muhammad.aminul@ub.ac.id[3]

Abstrak

Penggunaan *Kinect* sebagai alat untuk melakukan *input* kedalam komputer dengan membaca postur dan pergerakan tubuh dari pengguna meningkat dalam beberapa tahun ini. Penggunaan *input* dari *Kinect* ini banyak juga digunakan sebagai *input controller* untuk *Role Playing Game*. Penggunaan *Kinect Gesture Recognition* sebagai sebuah tambahan *controller* pada sebuah *game* yang berjudul *Steel Battalion: Heavy Armor* telah mendapat berbagai macam response *negative* yang dapat terlihat seperti *input* yang tidak disengaja ketika pengguna sedang tidak ingin melakukan apapun. Dibutuhan untuk membatasi *input* yang dilakukan oleh pengguna, untuk membatasi *input* yang tidak dimaksudkan pada saat pengguna sedang dalam posisi relaks atau diam, dan pada saat pengguna baru saja selesai melakukan sebuah *input*. Diperlukan pembatasan pembacaan *input* yang dilakukan oleh *Kinect* dengan membuat variabel *flag* untuk memisahkan fase dari sistem *controller* menjadi tiga tahap yaitu fase *standby*, fase *input* dan fase *reset*. Berdasarkan pengujian yang telah dilakukan, didapatkan kesimpulan bahwa pemisahan fase tersebut dapat menghilangkan *input* yang tidak dimaksudkan pada saat pengguna sedang berada dalam posisi relaks atau diam, dan pada saat pengguna baru saja selesai melakukan sebuah *input*.

Kata kunci: *Kinect Gesture Recognition, Role Playing Game, Controller, Flag.*

Abstract

The usage of Kinect to do an input into a computer by reading the movement and gesture of the user have been rising by these years. Kinect have been used alot as an input controller for a Role Playing Game. The usage of Kinect Gesture Recognition as an extended controller on a game named Steel Battalion: Heavy Armor had been getting many responses that the input done using kinect is unstable, that there are some input that unintended by the user happen when the user is doing nothing, and when the user had just done doing an input. limiting the reading of an input that kinect read using flag variable to divide input phase into 3, standby phase, input phase and reset phase, resulting in clearing the problem of unintended input that happened when user is doing nothing and when user had just done doing an input.

Keywords: *Kinect Gesture Recognition, Role Playing Game, Controller, Flag.*

1. PENDAHULUAN

Role Play Video Game (RPG) adalah sebuah permainan dengan pemain menggerakkan sebuah karakter untuk memainkan peran yang telah diperuntukkan pada saat pembuatan permainan. 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 *controller* yang digunakan pada *console* cukup untuk melakukan seluruh *input* yang ada (Adams & Rollings, 2006).

Gesture adalah bentuk pergerakan (non-verbal) yang bertujuan untuk menyampaikan informasi atau untuk berinteraksi dengan sekeliling. *Gesture* dapat dibagi menjadi dua kategori yaitu statik dan dinamik. Pada statik, *gesture* dibaca berdasarkan postur yang tertangkap pada instansi waktu tertentu, seperti

sebuah *gesture* seseorang sedang duduk di kursi membaca buku. Sedangkan dinamik membaca postur secara sekuensial yang disambungkan dengan urutan postur yang disambungkan dengan motion pada waktu yang singkat (Li, 2012).

Penggunaan Kinect *Gesture* Recognition sebagai sebuah tambahan *controller* pada sebuah *game* yang berjudul Steel Battalion: Heavy Armor telah mendapat berbagai macam response yang dapat terlihat ketidak stabilan pada control yang di *input*kan oleh user, dimana sering terdapat kesalahan *input* yang tidak ingin dilakukan user pada saat itu (*AngryJoeShow*, 2012). Permasalahan ini dapat diselesaikan dengan melakukan pemisahan situasi yang membedakan antara pada saat user ingin melakukan *input* atau tidak dengan menggunakan flag.

Sebuah flag adalah sebuah nilai yang berlaku sebagai sinyal didalam sebuah proses ataupun fungsi. Nilai dari flag tersebut digunakan untuk menentukan langkah selanjutnya dari sebuah program. Flag sering digunakan dalam bentuk binary, yang berisi nilai Boolean (true atau false). Tetapi, tidak semua flag merupakan binary, yang berarti flag dapat menyimpan beberapa jenis nilai. Contoh sederhana dari penggunaan flag adalah untuk keluar dari sebuah fungsi perulangan. (https://techterms.com, 2018)

Berdasarkan pembahasan serta solusi diatas, dalam penelitian ini menggunakan flag sebagai pembatas pada pembacaan data yang dilakukan oleh kinect kedalam tiga fase, yaitu fase standby, fase *input* dan fase *reset*.

2. LANDASAN KEPUSTAKAAN

2.1 Flag

Sebuah flag adalah sebuah variable yang berlaku sebagai sinyal didalam sebuah proses ataupun fungsi. Nilai dari flag tersebut digunakan untuk menentukan langkah selanjutnya dari sebuah program. Flag sering digunakan dalam bentuk binary, yang berisi nilai Boolean (true atau false). Tetapi, tidak semua flag merupakan binary, yang berarti flag dapat menyimpan beberapa jenis nilai. Contoh sederhana dari penggunaan flag adalah pembuatan variable dijadikan syarat untuk keluar dari sebuah perulangan.

Terdapat dua jenis flag, yaitu binary flag dan non-binary flag. Sebuah binary flag hanya memerlukan satu bit, yang dapat di tentukan sebagai 0 atau 1. Tetapi, byte merupakan 8 bit, yang berarti 7 bit lainnya tidak terpakai ketika 1 byte dipakai untuk menyimpan sebuah binary flag. Programmer dapat memilih untuk menyimpan beberapa binary flag dalam satu byte. Non-binary flag menggunakan beberapa bit dan dapat memilih tidak hanya antara "true atau false" saja. Tipe flag ini membutuhkan lebih dari 1 bit, tetapi tidak pasti memerlukan 1 byte penuh. Contohnya, 2 bit dapat menghasilkan 4 pilihan. 00 = jawaban A, 01 = jawaban B, 10 = jawaban C, 11 = jawaban D (https://techterms.com, 2018).

2.2 *Gesture* Recognition

Gesture recognition adalah sebuah cara yang bertujuan untuk mengenali postur dan gerakan tubuh manusia dengan menggunakan algoritma matematika. *Gesture* tersebut dapat dibaca dari seluruh posisi atau pergerakan bagian tubuh, tetapi umumnya hanya membaca pergerakan wajah ataupun tangan. Pengguna dapat menggunakan *gesture* sederhana untuk mengontrol ataupun berinteraksi dengan alat tanpa menyentuhnya. *Gesture* recognition dapat diartikan sebagai sebuah cara bagi komputer untuk memulai mengerti body language dari manusia (Shah & Patel, 2017).

Gambar 1 Pembacaan Telapak Tangan pada *Gesture* Recognition

Terdapat beberapa jenis *hand gesture* yang telah umum dipakai, diantaranya adalah, *Open*, *Lasso* dan *Close* yang dapat dilihat pada tabel 1. *Hand Gesture* tersebut dapat kita pakai sebagai salah satu syarat yang perlu dilakukan untuk melakukan sebuah *input*.

Tabel 1. Model *Hand Gesture Open, Lasso, Close*

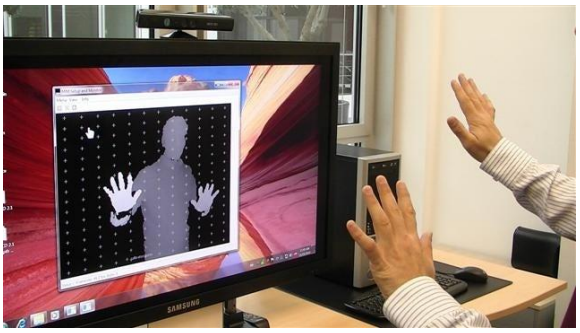
No.	Model *Hand*	Deskripsi

	Gesture	
1		*Hand Gesture*: *Open* *Hand Gesture Open* adalah posisi telapak tangan menghadap ke kamera kinect, dan kelima jari memanjang membentuk telapak tangan seperti di gambar.
2		*Hand Gesture*: *Lasso* *Hand Gesture Lasso* adalah posisi telapak tangan menghadap ke kamera kinect, dan satu hingga tiga jari memanjang membentuk telapak tangan seperti di gambar.
3		*Hand Gesture*: *Close* *Hand Gesture Close* adalah posisi telapak tangan menghadap kamera kinect, semua jari menekuk membentuk telapak tangan seperti di gambar.

2.3 Kinect

Kinect adalah sebuah *input* device yang mendeteksi serangkaian motion yang diproduksi oleh microsoft untuk Xbox 360, Xbox One dan Windows PC. Kinect, yang berbasiskan alat external yang menyerupai style webcam, yang digunakan untuk *input* data pada interaksi oleh pengguna dengan *console*/computer tanpa memerlukan *game controller*, melalui *gesture* dan perintah melalui berbicara.

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 objek yang ditangkap oleh kamera menjadi tiga warna. Depth sensor berguna melihat ruangan dalam dimensi ketiga (3D). Multiarray 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).



Gambar 2 Penggunaan sensor kinect untuk membaca bentuk tubuh *(https://newatlas.com/, 2010)*

2.4 Pengujian *Black box*

Pengujian *Black box* adalah metode pengujian yang dilakukan tanpa mengetahui struktur dalam dan desain dari sebuah program (http://softwaretestingfundamentals.com, 2019). Di dalam pengujian *black box*, kita hanya memfokuskan *input* dan *output* dari sistemnya tanpa memperhatikan apa yang terjadi di bagian dalam dari program. Metode ini digunakan untuk mencari *error* dalam kategori-kategori berikut:

- Fungsi yang hilang atau tidak benar.
- Kesalahan tampilan.
- *Error* pada struktur data atau akses database.
- *Error* pada *Behavior* atau performa.
- *Error* pada *Inisialisasi* atau *terminasi*.

Hal-hal yang perlu dilakukan untuk melakukan pengujian *black box* adalah :

- Menentukan kebutuhan dan spesifikasi dari sistem.
- Menentukan skenario valid *input* untuk mengecek apakah sistem memproses dengan benar, dan invalid *input* untuk mengecek apakah sistem dapat mendeteksi adanya *error*.
- Menentukan seluruh kemungkinan *input* yang ada.
- Membuat kasus pengujian untuk masing-

masing *input*.
- Menguji masing-masing kasus.
- Membandingkan *output* hasil pengujian dengan *output* yang diharapkan.

3. ALUR SISTEM

3.1 Analisis Kebutuhan

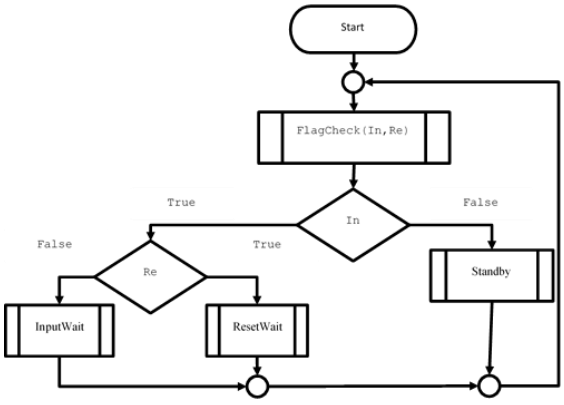
3.1.1 Kebutuhan Fungsional

Kebutuhan fungsional mendeskripsikan kebutuhan yang perlu dipenuhi oleh sistem. Adapun kebutuhan fungsional dari sistem yang saat ini dibangun dapat dilihat pada tabel 2.

Tabel 2. Kebutuhan Fungsional

No.	Kebutuhan Fungsional
1	Sistem dapat membaca posisi tangan kiri dan tangan kanan berdasarkan posisi kepala dari pengguna.
2	Sistem dapat membaca *gesture* dari telapak tangan
3	Sistem dapat membaca jarak antar kedua telapak tangan.
4	Sistem dapat menyimpan variable posisi tangan dan state tangan pada saat memulai *input*.
5	Sistem dapat mendeteksi tangan yang melakukan *input*.
6	Sistem dapat memisahkan fase antara fase standby, fase *input*, dan fase *reset*.

3.2 Skenario Alur Program



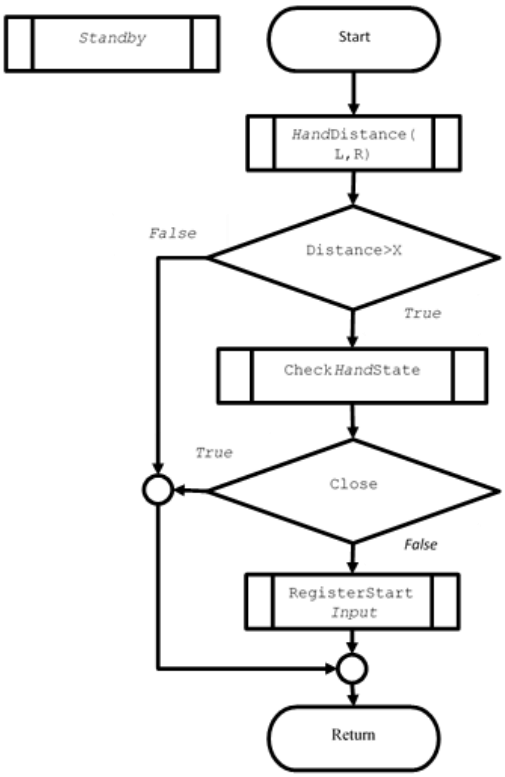
Gambar 3 Alur proses sistem

Gambar 3 menampilkan pembagian proses yang dikerjakan pada Setiap *input* data, yang dipisahkan oleh syarat yang dibutuhkan pada saat proses pengecekan *flag*. Proses-proses tersebut terbagi menjadi 3 fase, yaitu:

- Fase Standby
- Fase *Input*
- Fase *Reset*

3.2.1 Fase Standby

Fase Standby adalah adalah fase yang tidak menghasilkan *output*. Postur tubuh pengguna yang sedang tidak melakukan *input* secara sengaja, dapat diartikan sebagai posisi rileks, dimana pengguna sedang tidak memasukkan tenaga pada kedua tangan. Posisi rileks pada pengguna yang sedang duduk umumnya terlihat pada saat tangan pengguna berada diatas pangkuan, dimana posisi kedua tangan tidak lebih lebar dari bahu pengguna.



Gambar 4 Alur proses pada fase Standby

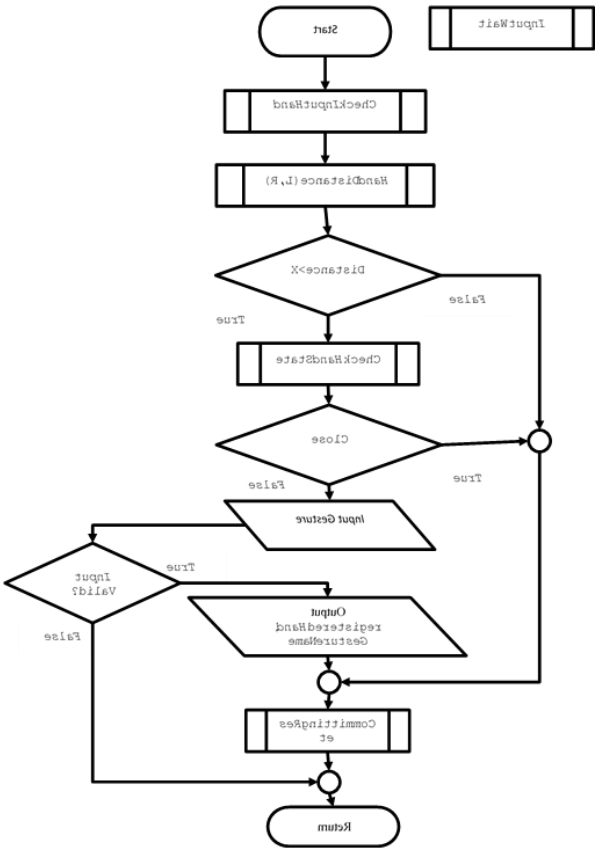
Pada gambar 4, menunjukkan alur proses dari fase Standby. Fase mengecek jarak antar tangan dari kedua user.

Apabila jarak tangan tidak melebihi nilai yang telah ditentukan oleh sistem pada variable X, sistem tidak akan melakukan aksi apapun dan kembali ke proses awal pengecekan flag. Apabila jarak tangan telah melebihi nilai variable X, maka sistem akan melanjutkan proses ke pengecekan *gesture* tangan.

Apabila tangan membentuk *gesture close*, sistem tidak akan melakukan aksi apapun dan kembali ke proses awal pengecekan flag. Apabila tangan membentuk *gesture* lain seperti

lasso atau *open*, maka sistem akan melanjutkan ke proses selanjutnya, yaitu menyimpan data tangan pada saat itu, dan mengubah flag untuk memenuhi syarat memasuki fase *input*.

3.2.2 Fase *Input*

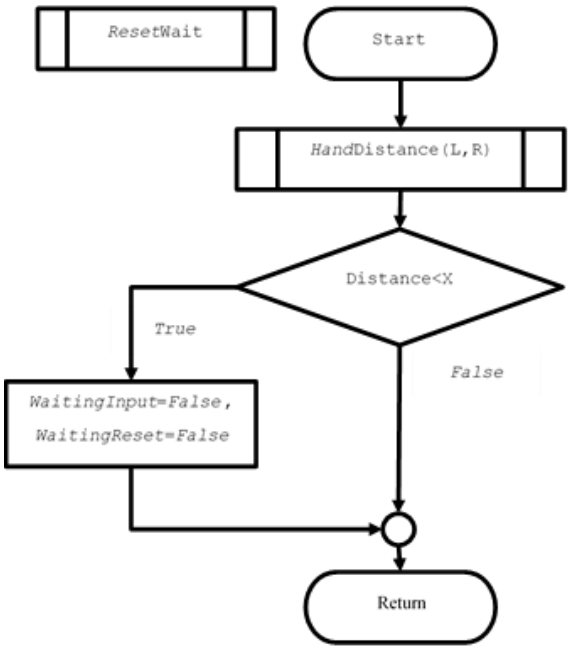


Gambar 5 Alur proses pada fase *Input*

Fase *Input* adalah adalah fase yang menghasilkan *output*. Pada fase ini, dilakukan pengecekan data dari posisi tangan dan *gesture* tangan pengguna pada fase sebelumnya. Fase ini menunggu hingga pengguna melakukan sebuah *gesture* lanjutan atau melakukan *gesture* tangan *Close*.

Apabila pengguna melakukan *gesture close*, maka sistem menganggap bahwa pengguna membatalkan *input* yang dilakukan dan mengubah flag untuk memasuki fase *reset*. Apabila *input gesture* belom valid, sistem akan kembali ke proses awal pengecekan flag. Apabila *input gesture* telah valid, sistem akan mengeluarkan *output* sesuai dengan yang telah didaftarkan dengan *gesture* yang dilakukan, dan mengubah flag untuk memasuki fase *reset*.

3.2.3 Fase *Reset*



Gambar 6 Alur proses pada fase *Reset*

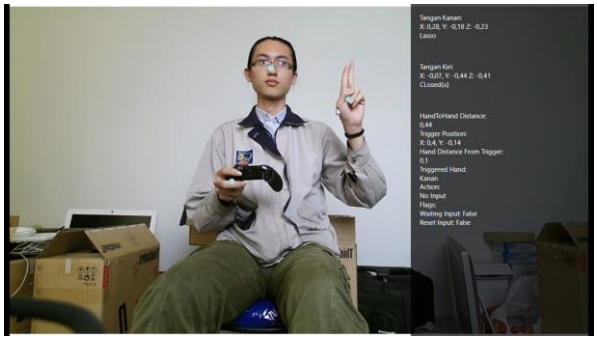
Fase *Reset* dibuat untuk menghentikan *input* yang diterima oleh kinect pada saat pengguna selesai memasukkan sebuah *input* atau membatalkan *input* dengan melakukan *gesture* tangan *Close*. Pada fase ini, data gestur tangan tidak diperdulikan, dan sistem menunggu hingga pengguna mendekatkan jarak antar tangan hingga kurang dari variable X yang dipakai sebagai salah satu syarat untuk melakukan *input* pada fase sebelumnya. Dengan mendekatkan jarak antar tangan, sistem akan mengubah nilai flag untuk memasuki fase Standby.

4. HASIL DAN PENGUJIAN

Pada bagian ini, akan dijelaskan mengenai hasil dari program dan pengujian menggunakan metode Black-Box untuk pengecekan fungsi-fungsi dari program.

4.1.1 Antarmuka

Untuk mengetahui data-data yang digunakan dan diproses di dalam sistem, ditampilkan data-data berikut ini di bagian kanan dari layar antarmuka:



Gambar 7 Antarmuka

- Posisi vector 3D dari tangan kanan dan tangan kiri dilihat berdasarkan posisi kepala. Nilai posisi dapat dilihat di bagian kanan-atas layar, di bawah nama masing-masing tangan. X merupakan nilai posisi kanan-kiri, semakin posisi berada ke-kanan, semakin besar nilai X. Y merupakan nilai posisi atas-bawah, semakin posisi ke-atas, semakin besar nilai Y. Z merupakan nilai posisi kedalaman depan-belakang, semakin menjauh dari layar, semakin besar nilai Z.
- *Gesture* tangan, yang dapat berupa *Lasso*, *Open*, dan *Close*d seperti yang dapat dilihat pada tabel 1. *Gesture* tangan yang tidak terbaca oleh sistem ikut dikategorikan sebagai *Close*d. Nama *gesture* tangan dapat terlihat pada posisi kanan-atas layar, di bawah posisi vector 3D tangan.
- Jarak antar tangan, yang di hitung menggunakan rumus pythagoras berikut ini

$$Jarak\ Antar\ Tangan = \sqrt{X^2 + Y^2}$$

  Nilai jarak antar tangan dapat dilihat di posisi kanan layar, tertulis di bawah *Hand*To*Hand* Distance.
- Trigger Position, posisi X dan Y dari tangan yang melakukan *gesture* pada saat awal memasuki fase *input*. Nilainya hanya berubah pada saat awal memasuki fase *input*.
- Triggered *Hand*, merupakan nama dari tangan yang melakukan *input*. Hanya dapat memiliki nilai "Kanan" atau "Kiri".
- Action, hasil dari *input* yang telah dilakukan. Nilainya berubah pada saat terjadi *input* yang valid pada alur yang terdapat di gambar 5. Isi dari nilai Action adalah "Nama *Input*" dan "Tangan yang melakukan *input*".
- Flag Waiting *Input* dan Flag *Reset Input*. Variable ini digunakan untuk persyaratan pemisah antara fase standby, fase *input*, dan fase *reset*.

4.1.2 Pengujian

Pada bagian ini, akan ditampilkan hasil dari pengujian *black box* pada fungsi-fungsi yang telah disebutkan pada kebutuhan fungsional.

a) Pengujian pembacaan posisi tangan kiri dan kanan berdasarkan posisi kepala

Tabel 3 Pengujian pembacaan posisi tangan

No	X	Y	Z	Deskripsi Posisi
1	0.27	-0.21	-0.24	Posisi tangan kanan terangkat di samping luar bahu dan lebih tinggi dari bahu
2	0.02	-0.43	-0.44	Posisi tangan kanan mendekati tangan kiri di depan dada

Keterangan:

Dilakukan dua percobaan pengambilan data, percobaan pertama adalah mengangkat tangan kanan hingga sedikit lebih tinggi dari bahu dan berada di bagian kanan bahu. Percobaan kedua adalah memindahkan tangan kanan hingga ke bagian depan dada. Dengan melihat ke bagian kanan layar antarmuka seperti pada gambar 7, dapat dilihat posisi dari tangan kanan pada saat melakukan percobaan. Nilai dari X, Y dan Z dimasukkan ke tabel 3 untuk melihat perubahan nilai masing-masing percobaan.

Nilai X pada percobaan kedua lebih rendah dari pada nilai X pada percobaan pertama, yang berarti posisi tangan kanan pada percobaan kedua berada lebih ke kiri dibandingkan dengan percobaan pertama. Nilai Y pada percobaan kedua lebih rendah dari pada nilai Y pada percobaan pertama, yang berarti posisi tangan kanan pada percobaan kedua berada lebih ke bawah dibandingkan dengan posisi tangan kanan pada percobaan pertama.

Posisi dada pengguna berada lebih ke kiri dan ke bawah dibandingkan dengan posisi bahu. Hal ini membuktikan kalau data dari percobaan pada tabel 3 bernilai valid.

b) Pengujian pembacaan *gesture* dari telapak tangan

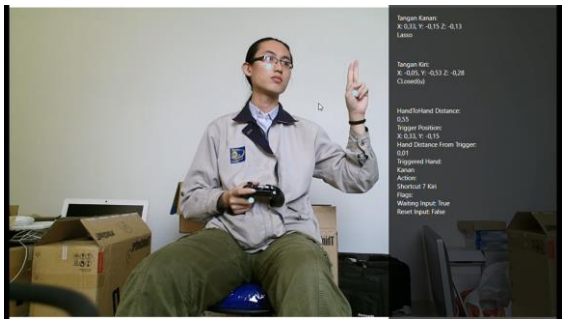
Dilakukan tiga percobaan untuk menguji pembacaan *gesture* tangan yang berupa *Lasso*, *Open*, *Close*.

Tabel 4 Pengujian pembacaan *gesture* tangan

No	Deskripsi Hasil Pengujian
1	Tangan kanan membentuk *gesture Lasso*
2	Tangan kanan membentuk *gesture Open*
3	Tangan kanan membentuk *gesture Close*

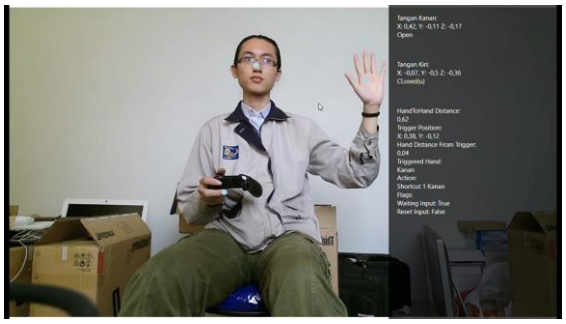
Keterangan:

Percobaan pertama, pengguna menampilkan telapak tangan kanan yang menghadap ke kamera kinect, membuat pose kelima jari tertutup, dan meluruskan jari telunjuk dan jari tengah seperti pada gambar 8.



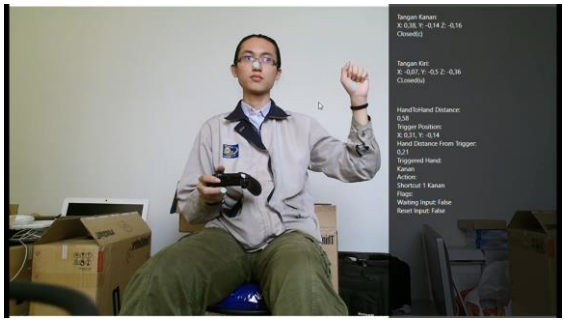
Gambar 8 Percobaan pertama

Percobaan kedua, pengguna menampilkan telapak tangan kanan menghadap ke kamera kinect, membuat pose kelima jari terbuka lurus seperti pada gambar 9.



Gambar 9 Percobaan kedua

Percobaan ketiga, pengguna menampilkan telapak tangan kanan menghadap ke kamera kinect, membuat pose kelima jari tertutup seperti pada gambar 10.



Gambar 10 Percobaan ketiga

Dapat dilihat bahwa hasil dari masing-masing percobaan dari tabel 4, sesuai dengan data yang telah ditampilkan pada tabel 1 yang membuktikan kalau *gesture* yang telah dibuat dalam sistem valid.

c) Pengujian pembacaan jarak antar kedua telapak tangan.

Tabel 5 Pengujian pembacaan jarak antar tangan

No	Jarak antar tangan	Deskripsi posisi tangan
1	0.41	Tangan kanan berada jauh dari tangan kiri
2	0.04	Tangan kanan berada di dekat tangan kiri

Keterangan:

Semakin tinggi nilai jarak antar tangan, semakin jauh posisi antar masing-masing tangan.

Pada kedua percobaan pada tabel 5, dapat dilihat bahwa nilai jarak antar tangan pada saat tangan kanan berada jauh dari tangan kiri pada percobaan 1, lebih besar dari pada nilai jarak antar tangan pada saat kedua tangan berdekatan. Hal ini membuktikan bahwa nilai jarak antar tangan valid.

d) Sistem dapat menyimpan variable posisi tangan dan state tangan pada saat memulai *input*.

Tabel 6 Pengujian penyimpanan posisi tangan pada saat *input*

No	Posisi Mulai *Input*	Posisi Tangan	Deskripsi Percobaan
1	X:0.38 Y:-0.12	X:0.38 Y:-0.1	Tangan kanan memulai *input* dengan melakukan *gesture open*
2	X:0.38	X:0.61	Posisi tangan kanan

	Y:-0.12	Y:-0.25	bergerak setelah *input* dimulai pada percobaan 1
3	X:-0.42 Y:-0.25	X:-0.42 Y:-0.21	Tangan kiri memulai *input* dengan melakukan *gesture open*

Keterangan:

Percobaan pertama, pada fase standby, pengguna menggunakan tangan kanan, melakukan *gesture Open* dan memasuki fase *input*. Posisi mulai *input* telah tercatat seperti yang ada pada tabel 6.

Percobaan kedua, pengguna menggerakkan tangan kanan yang telah melakukan *gesture Open* dan telah berada pada fase *input*. Posisi tangan kanan berubah, tetapi posisi mulai *input* tidak berubah.

Percobaan ketiga, pada fase standby, pengguna menggunakan tangan kiri, melakukan *gesture Open* dan memasuki fase *input*. Posisi mulai *input* telah tercatat seperti yang ada pada gambar 5.

Pada percobaan pertama dan ketiga pada tabel 6, posisi mulai *input* berubah sesuai dengan posisi tangan yang melakukan *gesture* pada saat *input* dimulai. Pada percobaan kedua, posisi tangan kanan yang melakukan *input* diubah, tetapi nilai pada Posisi Mulai *Input* yang berada pada tabel 6 tidak berubah.

Dapat dilihat dari hasil percobaan di tabel 6 bahwa penyimpanan posisi mulai *input* yang dilakukan oleh masing-masing tangan telah berhasil disimpan sesuai dengan posisi tangan yang melakukan *gesture*, dan tidak berubah hingga perubahan fase dari fase standby memasuki fase *input*.

e) Sistem dapat mendeteksi tangan yang melakukan *input*.

Tabel 7 Pengujian pembacaan tangan yang memulai *input*

No	Triggered *Hand*	Deskripsi
1	Kanan	Tangan kanan melakukan *gesture* untuk memulai *input*
2	Kiri	Tangan kiri melakukan *gesture* untuk memulai *input*

Keterangan:

Pada tabel 7, dapat dilihat pada saat tangan kanan melakukan *gesture* untuk memulai *input*, nilai dari Triggered *Hand* yang dapat dilihat pada antarmuka di gambar 7 berubah sesuai dengan tangan yang melakukan *input*.

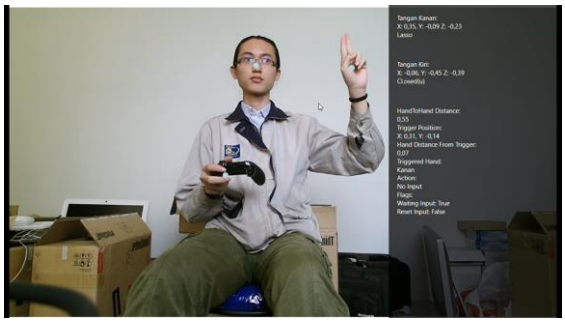
f) Sistem dapat memisahkan fase antara fase standby, fase *input*, dan fase *reset*.

Tabel 8 Pengujian perpindahan antar fase

No	Deskripsi Percobaan	Hasil
1	Merubah fase dari fase standby menjadi fase *input* dengan menjauhkan tangan dan melakukan sebuah *gesture* tangan.	Valid
2	Merubah fase dari fase *input* menjadi fase *reset* dengan menyelesaikan sebuah rangkaian *input*.	Valid
3	Merubah fase dari fase *input* menjadi fase *reset* dengan melakukan *gesture* tangan *Close* pada tangan yang melakukan *input*.	Valid
4	Merubah fase dari fase *reset* menjadi fase standby dengan mendekatkan posisi kedua tangan.	Valid

Keterangan:

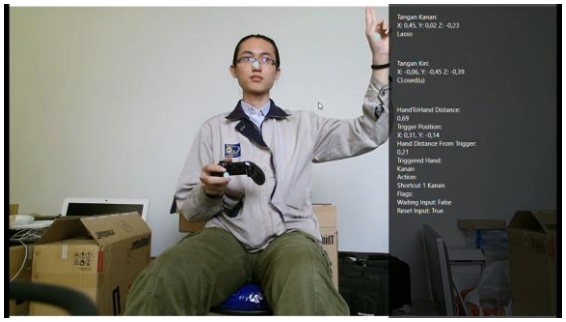
Percobaan pertama, pengguna menggerakkan tangan kanan menjauh dari tangan kiri, dan melakukan *gesture* tangan *lasso* Seperti pada gambar 11. Dapat terlihat pada bagian kanan gambar terdapat Flags, dimana nilai dari Waiting *Input* adalah true, dan nilai *Reset Input* adalah false, yang berarti program memasuki fase *input*.



Gambar 11 Percobaan pertama

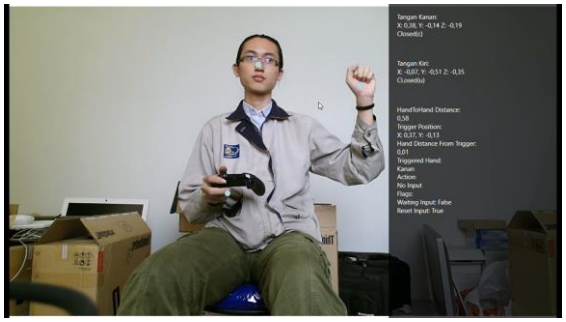
Percobaan kedua, pengguna menggerakkan tangan ke arah kanan atas melebihi jarak 0.2m untuk memenuhi syarat untuk melakukan sebuah *input* yang bernama Shortcut 1, yang dilakukan oleh tangan Kanan, seperti yang

tertulis pada bagian kanan layar pada gambar 12 di bagian Action. Pada saat pengguna melakukan *gesture* yang valid dan berhasil menghasilkan sebuah *input*, flag Waiting *Input* berubah menjadi false, dan flag *Reset Input* berubah menjadi true, yang berarti program memasuki fase *Reset*.



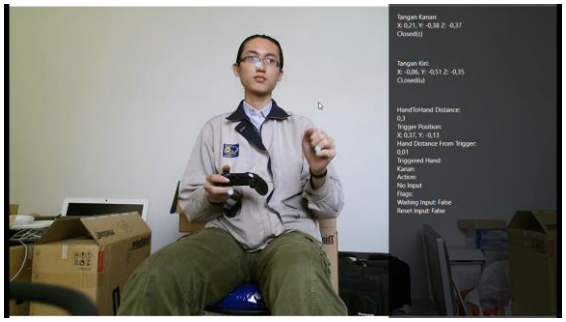
Gambar 12 Percobaan kedua

Percobaan ketiga, pengguna melakukan *gesture close* untuk membatalkan *input*. Dapat terlihat pada gambar 13, flag Waiting *Input* bernilai false, dan flag *Reset Input* bernilai true, yang berarti program berada pada fase *Reset*.



Gambar 13 Percobaan ketiga

Percobaan keempat, pengguna mendekatkan kedua tangan untuk memasuki fase standby. Dapat terlihat pada gambar 14, flag Waiting *Input* dan *Reset Input* bernilai false, yang berarti program berada pada fase Standby.



Gambar 14 Percobaan keempat

5. Kesimpulan

Jurnal ini menyajikan sebuah langkah untuk mengurangi permasalahan dari pembacaan data yang tidak efektif pada kinect dengan melakukan pembatasan *input* dengan menggunakan flag. Berdasarkan hasil pengujian *black box* pada BAB sebelumnya, telah didapat hasil bahwa metode penyelesaian ini berhasil mengurangi *input* yang tidak dimaksudkan pada pembacaan *gesture* kinect.

6. DAFTAR PUSTAKA

Adams, E. & Rollings, A. 2006. *Fundamentals of Game Design (Game Design and Development Series)*. Prentice-Hall, inc. Available at: http://ptgmedia.pearsoncmg.com/images/9780321929679/samplepages/0321929675.pdf [Acessed 11 July 2019]

AngryJoeShow. (2012, July 03). Steel Battalion: Heavy Armor Angry Review [Video file]. Retrieved from https://www.youtube.com/watch?v=_ArpWuV_zvo [Acessed 11 July 2019]

https://newatlas.com/. (2010). *New Atlas*. Retrieved July 5, 2019, from https://newatlas.com/

https://techterms.com. (2018). *Techterms*. Retrieved June 14, 2019, from https://techterms.com/definition/flag

Li, Y. (2012). *Hand Gesture* Recognition Using Kinect. 196.

Shah, N., & Patel, J. (2017). *Gesture* Recognition Technique:A Review. *International Journal of Recent Trends in Engineering & Research*, 550.