SINKRONISASI OBJEK *CLIENT* PADA *GAME MULTIPLAYER ONLINE* DENGAN METODE *DEAD RECKONING*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Dwi Hardyanto
NIM: 115060800111055

INFORMATIKA/ILMU KOMPUTER
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2015

PENGESAHAN

SINKRONISASI OBJEK *CLIENT* PADA *GAME MULTIPLAYER ONLINE* DENGAN METODE *DEAD RECKONING*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Dwi Hardyanto
NIM: 115060800111055

Skripsi ini telah diuji dan dinyatakan lulus pada
7 Januari 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Eriq Muhammad Adams J, S.T., M.Kom.	Issa Arwani, S.Kom., M.Sc.
NIP: 19850410 2012121001	NIP: 19830922 2012121003

Mengetahui
Ketua Program Studi Informatika/Ilmu Komputer

Drs. Marji, M.T.
NIP: 19670801 199203 1 001

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 pustaka.

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, 7 Januari 2016

Dwi Hardyanto

NIM: 115060800111055

KATA PENGANTAR

Alhamdulillahi rabbil 'alamin. Puji syukur penulis panjatkan kehadirat Allah SWT, karena atas segala rahmat dan limpahan hidayah-Nya, penulis dapat menyelesaikan tugas akhir yang berjudul "**SINKRONISASI OBJEK *CLIENT* PADA *GAME MULTIPLAYER ONLINE* DENGAN METODE *DEAD RECKONING***".

Dalam pelaksanaan dan penulisan tugas akhir ini penulis mendapatkan banyak bantuan dari berbagai pihak baik secara moril maupun materiil. Dalam kesempatan ini penulis ingin mengucapkan terimakasih yang sebesar–besarnya kepada:

1. Ibu Luh Suhartini, Bapak Edy Mugiharjo, Kakak Aang Samsudy, Adik Iga Andyni dan seluruh keluarga besar, atas segala nasehat, kasih sayang, perhatian dan kesabarannya di dalam membesarkan dan mendidik penulis, serta yang senantiasa tiada henti-hentinya memberikan doa dan semangat demi terselesaikannya skripsi ini.
2. Bapak Eriq Muhammad Adams J, S.T., M.Kom. dan Bapak Issa Arwani, S.Kom., M.Sc. selaku dosen pembimbing skripsi yang telah dengan sabar membimbing dan mengarahkan penulis sehingga dapat menyelesaikan skripsi ini.
3. Bapak Ir. Sutrisno, M.T, Bapak Ir. Heru Nurwasito, M.Kom, Bapak Himawat Aryadita, S.T, M.Sc, dan Bapak Eddy Santoso, S.Kom selaku Ketua, Wakil Ketua 1, Wakil Ketua 2 dan Wakil Ketua 3 Fakultas Ilmu Komputer Universitas Brawijaya.
4. Bapak Drs. Marji, M.T dan Bapak Issa Arwani, S.Kom, M.Sc selaku Ketua dan Sekretaris Program Studi Informatika/Ilmu Komputer Universitas Brawijaya.
5. Seluruh Dosen Informatika/Ilmu Komputer Universitas Brawijaya atas kesediaan membagi ilmunya kepada penulis.
6. Seluruh Civitas Akademika Informatika/Ilmu Komputer Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penulis menempuh studi di Teknik Informatika Universitas Brawijaya dan selama penyelesaian skripsi ini.
7. Ria Rizki Wardani yang selalu memberi motivasi, semangat, berbagi banyak pandangan-pandangan, pemikiran, dan berjalan bersama disisi penulis.
8. Afi Muftihul Situmorang, Claudio Fresta S, Arik Achmad, Grandis Mahendra, Weni Prameswari, Sheila Lasahido, Albilaga Linggra P, Anas Rachmadi P, Fendy Gusta P, Dwi Vendy Pratama, Roshikan Maulana Y, Andryanto, I Putu Yoga P, Nadia Previani, Alvin Hermawan yang sudah menjadi sahabat seperjuangan dari awal, selalu bertukar semangat, pendapat, dan ide-ide dengan penulis.
9. Keluarga besar LSO Raion Community yang telah berbagi ilmu dan pandangan dengan penulis.

10. Teman–teman angkatan 2011 Informatika/Ilmu Komputer, terima kasih atas segala bantuannya selama menempuh studi di Informatika/Ilmu Komputer Filkom Universitas Brawijaya.
11. Seluruh pihak yang telah membantu kelancaran penulisan tugas akhir yang tidak dapat kami sebutkan satu persatu.

Penulis menyadari bahwa dalam penyusunan tugas akhir ini masih banyak kekurangan baik format penulisan maupun isinya. Oleh karena itu, saran dan kritik membangun dari para pembaca senantiasa penulis harapkan guna perbaikan bagi tugas akhir selanjutnya. Semoga tugas akhir ini dapat memberikan manfaat bagi semua pihak, Amin.

Malang, 7 Januari 2016

Penulis

hardy.archer@gmail.com

ABSTRAK

Game multiplayer terus berkembang sesuai jaman, ada yang dapat dimainkan secara *offline* maupun *online*. *Game multiplayer online* ini membutuhkan koneksi *internet* dimana tingkat ketersediaan dan kestabilan koneksipun menjadi salah satu hal yang sangat penting untuk diperhatikan. Oleh karena itu, maka dibutuhkan sebuah metode yang dapat mengurangi beban *bandwidth* dan mengatasi masalah ketidak stabilan koneksi yang terjadi. Metode *Dead Reckoning* salah satunya. Metode ini diambil dari sistem navigasi kapal laut untuk melakukan prediksi letak sebuah objek pada jangka waktu tertentu sesuai dengan status kinematik awalnya. Pengujian untuk mengetahui tingkat efektifitas metode ini ada dua, yaitu dengan menguji kinerjanya berdasarkan jumlah pengiriman data yang dilakukan dan berdasarkan kesesuaian prediksi posisi objek. Ada banyak variasi *Dead Reckoning* yang telah dikembangkan, seperti variasi *Velocity Blending* salah satunya. Berdasarkan hasil penelitian yang dilakukan pada dokumen ini, dapat diketahui jika dari segi jumlah penghematan *bandwidth* metode *Dead Reckoning* asli memiliki tingkat yang paling optimal diantara metode *Dead Reckoning* variasi *Velocity Blending* maupun metode sinkronisasi konvensional, sedangkan dari segi kesesuaian posisinya menggunakan metode *Dead Reckoning* variasi *Velocity Blending* yang memiliki kesesuaian prediksi yang paling optimal.

Kata kunci: *Game Multiplayer Online*, Sinkronisasi Objek, *Dead Reckoning*, *Velocity Blending*, Jaringan, *Bandwidth*, *Latency*

ABSTRACT

Game multiplayer has growth by the improvement of technology, it can be play offline or online. A Game multiplayer online need an internet connection where a good and stable connection become something that really important to discuss. That is why, a method that can save bandwidth and handle the unstable connection is needed. Dead Reckoning is one of them. Dead Reckoning is actually a navigation system used by navy to predict a position of an object after several time by it's kinematic status. There were two ways to know the effectivity of this method, by the total data sent and how match the position of the original object and the remote object. So many variation of Dead Reckoning has been developed, Velocity Blending is one of them. Based on this research, the original Dead Reckoning is the most optimal to saving bandwidth, but the Velocity Blending variation of Dead Reckoning has the closest position between the original object and the remote object.

Keywords: Game Multiplayer Online, Object Syncronization, Dead Reckoning, Velocity Blending, Network, Bandwidth, Latency

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar belakang

Teknologi merupakan bagian yang tidak dapat dipisahkan dari kehidupan manusia, tidak hanya untuk membantu dan memudahkan kebutuhan fungsional saja, namun juga untuk memenuhi kebutuhan hiburan seperti *Video Games*. *Video games* merupakan sebuah permainan elektronik yang melibatkan interaksi antara manusia dan sebuah *user interface* dan menghasilkan gambaran visual pada sebuah alat *display* [TGA-08]. Beberapa *video games* juga tidak hanya bisa dinikmati oleh satu orang saja dalam satu kali permainan, namun ada juga yang dapat dimainkan oleh beberapa pemain sekaligus yang disebut dengan *Game Multiplayer* [OED-08].

Game Multiplayer terus berkembang sesuai jaman hingga dapat dimainkan secara *online*. *Game multiplayer online* ini membutuhkan koneksi *internet* untuk memainkannya, karena itulah tingkat ketersediaan dan kestabilan koneksipun menjadi salah satu hal yang sangat penting untuk diperhatikan agar setiap pemain mendapatkan *user experience* yang sama dan merata. Sebuah metode dibutuhkan guna memberikan solusi untuk melakukan sinkronisasi pada setiap objek dan komponen pada *game multiplayer online*. Metode tersebut harus dapat mengatasi tingkat kebutuhan pengiriman dan penerimaan data yang besar, sehingga bisa mengurangi beban *bandwidth* yang diperlukan dan *game* dapat berjalan secara lancar. Salah satu metode yang dapat digunakan untuk melakukan sinkronisasi pada *game multiplayer* ini adalah menggunakan metode *Dead Reckoning*. Metode *Dead Reckoning* sangat bermanfaat untuk menangani banyaknya paket data yang dikirimkan melalui jaringan nirkabel[BAH-08]. Dengan metode ini, data dan informasi objek *client* pada *game multiplayer online* tidak dikirim secara terus menerus seperti pada metode sinkronisasi konvensional, melainkan diestimasikan dan diprediksi pada jangka waktu tertentu dimana letak posisi objek tersebut berikutnya berdasarkan posisi lokasi sekarang dan laju kecepatan objek tersebut [HRC-10].

Metode *Dead Reckoning* dan variasinya telah beberapa kali dibahas baik dalam *paper*, *journal,* maupun buku, seperti pada *paper* hasil penelitian yang dilakukan oleh Alf Inge Wang, Martin Jarret, and Eivind Sorteberg pada tahun 2009 tentang implementasi sebuah *mobile multiplayer real-time game* pada jaringan nirkabel dengan *latency* yang tinggi, ada pula penelitian yang dilakukan *oleh* Séamus C. McLoone, Patrick J. Walsh dan Tomás E. Ward pada tahun 2012 mengenai penerapan *Dead Reckoning* pada *game* yang menggunakan fisika, dan ada juga variasi *Dead Reckoning Velocity Blending* yang ditulis oleh Curtiss Murphy dan dicantumkan dalam buku "Game Engine Gems 2" yang terbit pada tahun 2011. Dengan adanya bahasan yang telah dilakukan tersebut maka penulis ingin melakukan implementasi dan analisa tentang seberapa efektifnya metode asli *Dead Reckoning* (Interpolasi) maupun metode *Dead Reckoning* variasi *Velocity Blending* jika diterapkan pada sebuah *game multiplayer online* dibandingkan

dengan metode konvensional baik dari segi jumlah pengiriman datanya maupun kesesuaian posisinya.

1.2 Rumusan masalah

Berdasarkan pada permasalahan yang telah dijelaskan pada bagian latar belakang, maka rumusan yang akan dibahas pada penelitian ini meliputi :

1. Bagaimana menerapkan metode *Dead Reckoning* (Interpolasi) dan metode *Dead Reckoning* variasi *Velocity Blending* pada sebuah *game multiplayer online*?
2. Bagaimana tingkat efektifitas metode *Dead Reckoning* (Interpolasi) dan metode *Dead Reckoning* variasi *Velocity Blending* yang telah diterapkan pada *game multiplayer online* dibandingkan dengan metode sinkronisasi konvensional?

1.3 Tujuan

Penelitian ini dilakukan dengan tujuan untuk mengimplementasikan metode *Dead Reckoning* (Interpolasi) dan metode *Dead Reckoning* variasi *Velocity Blending* pada sebuah *game multiplayer online* untuk mengetahui tingkat efektifitas metode tersebut dalam mengatasi besarnya *bandwidth* yang diperlukan dengan tingkat kestabilan koneksi yang beragam.

1.4 Manfaat

Adapun manfaat yang diharapkan dalam penulisan penelitian ini adalah :

1. Dapat mengetahui tingkat efektifitas sinkronisasi dengan metode *Dead Reckoning* (Interpolasi) dan metode *Dead Reckoning* variasi *Velocity Blending* dibandingkan dengan sinkronisasi konvensional pada *game multiplayer online*
2. Sebagai acuan literatur dalam penelitian dan pengembangan *game multiplayer online* kedepannya

1.5 Batasan Masalah

Penelitian yang dilakukan memiliki batasan-batasan masalah sebagai berikut:

1. Studi kasus yang digunakan adalah game *multiplayer online* yang dikembangkan menggunakan Unity3D
2. Pengujian kinerja dilakukan dengan menggunakan simulator jaringan milik Unity3D

1.6 Sistematika pembahasan

Penelitian ini disusun berdasarkan sistematika penulisan yang terbagi menjadi enam bab, antara lain :

BAB I PENDAHULUAN

Bab ini memaparkan tentang latar belakang permasalahan, rumusan masalah, tujuan, dan manfaat dilakukannya penelitian Sinkronisasi Objek *Client* pada *Game Multiplayer Online* dengan Metode *Dead Reckoning*.

BAB II LANDASAN KEPUSTAKAAN

Bab ini memaparkan teori-teori dasar dan teori-teori penunjang yang digunakan dalam penelitian Sinkronisasi Objek *Client* pada *Game Multiplayer Online* dengan Metode *Dead Reckoning*. Teori yang dimaksud berkaitan dengan Jaringan Internet, *Game Multiplayer Online,* dan metode *Dead Reckoning*.

BAB III METODE PENELITIAN

Bab ini berisi tentang studi literatur untuk dasar teori, metode pengambilan data, metode yang digunakan dalam perancangan, pengujian dan analisis, serta metode-metode lain yang relevan dengan penelitian yang dilakukan.

BAB IV PERANCANGAN

Bab ini memaparkan tentang bagaimana analisa kebutuhan dan melakukan perancangan sesuai dengan target yang dicapai oleh penelitian.

BAB V IMPLEMENTASI

Bab ini memaparkan tentang batasan teknis implementasi penyelesaian masalah, algoritma penyelesaian masalah, dan *file-file* implementasi pendukung dalam pengembangan dan penelitian yang diimplementasikan.

BAB VI PENGUJIAN DAN ANALISIS

Bab ini memuat proses dan pengujian teknologi hasil implementasi penyelesaian masalah yang diterapkan dengan masalah sebenarnya dan dilakukan analisis hasil keluaran teknologi dengan ruang lingkup permasalahan yang ada.

BAB VII PENUTUP

Bab ini memaparkan kesimpulan serta saran yang diperoleh atas pengujian dan analisis yang telah dilakukan dalam proses penelitian.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini membahas dasar-dasar teori yang digunakan untuk menunjang penulisan skripsi mengenai implementasi dari metode *Dead Reckoning* (Interpolasi) dan metode *Dead Reckoning* variasi *Velocity Blending* pada sebuah game multiplayer online, yaitu melingkupi tentang jaringan dan *latency* pada sebuah game multiplayer online, dan juga konsep dan jenis dari Dead Reckoning itu sendiri.

2.1 Game Multiplayer Online

Game multiplayer online pertama terbentuk dan terinspirasi dari sebuah sistem PLATO *time-sharing* yang dibuat oleh Universitas Illinois pada tahun 1970 untuk tujuan belajar secara online. Sistem ini terus dikembangkan, dan pada tahun 1972 muncul PLATO IV terminal dengan graphical baru yang kemudian digunakan oleh para muridnya untuk membuat *game multiplayer*. Keluar dari tujuan sistem itu dibuat, muncul MUDs (*Multi-User* Dungeons). Pada tahun 1978, PLATO telah memiliki beberapa *game multiplayer* seperti *Dungeon Crawls, Air Combat (Airfight), Tank Combat,* dan *Space Battle* [CRC-01].

Game multiplayer online lebih mengarah ke sebuah model *single session* atau *persistent world*. Pada *single session game*, *player* terkoneksi ke sebuah *server* untuk bergabung dengan *player* lainnya, bisa berupa *server* yang dimiliki oleh sebuah *game publisher* atau berjalan langsung disebuah komputer milik seorang *player. Single session game* biasanya terdiri dari dua hingga sepuluh *player* yang bermain secara bersama-sama, dimana terdapat fitur *Lobby* untuk menunggu dan membuat tim terbentuk [CRC-01].

2.1.1 Jaringan pada *Game Multiplayer Online*

G*ame multiplayer online* menggunakan sistem *client-server*, dimana sebuah *client* mengirimkan paket data informasi pada sebuah *server* utama dan kemudian menerima paket kembali dari *server* tersebut sebagai kembalian. Server memiliki dan menyimpan informasi-informasi dari game tersebut, dan bertugas untuk mengatur setiap *client* yang terhubung [ZNL-04].

Pertukaran data antara *client* dan *server* terjadi melalui sebuah protokol *internet* yang disebut dengan TCP/IP dan dijembatani melalui sebuah *socket*. TCP/IP terbagi menjadi dua layer yang merupakan bagian dari *Open Systems Interconnection* (OSI) model. *Transmission Control Protocol* (TCP) merupakan bagian dari *transport layer* pada OSI model, sedangkan *Internet Protocol* (IP) merupakan bagian dari *network layer*. Terdapat pula protokol lain yang biasa digunakan untuk menggantikan TCP pada *transport layer* yaitu *User Datagram Protokol* (UDP) [MHW-04].

2.1.1.1 Transmission Control Protocol

Transmission Control Protocol merupakan protokol *reliable*, dimana setiap paket informasi yang dikirim akan dipastikan untuk dapat mencapai tujuan dan tanpa harus khawatir terjadinya duplikasi data. Protokol ini akan membuat koneksi terlebih dahulu pada target yang dituju sebelum melakukan komunikasi. Jika terjadi kegagalan atau paket tidak mencapai tujuan dalam waktu tertentu, maka *host* akan menganggap paket tersebut telah hilang, kemudian mengirim paket informasi yang baru sebagai penggantinya [MHW-04].

2.1.1.2 User Datagram Protocol

User Datagram Protocol merupakan protokol yang tidak reliable, dimana setiap paket yang dikirim tidak dapat dipastikan untuk mencapai tujuan dan dapat terjadi duplikasi data. Protokol ini berkomunikasi dengan cara mengirimkan paket pada target *host*, kemudian menunggu kembalian dari *host* tersebut. Ketika sebuah *host* menerima sebuah paket maka *host* ini akan tahu darimana paket itu berasal, kemudian dapat mengirim paket kembalian ke alamat *host* pengirim tersebut. Banyak game yang menggunakan protokol ini untuk melakukan komunikasi antar *client-server* [MHW-04].

2.1.2 Efek *Latency* Jaringan pada *Game Multiplayer Online*

Pesan informasi data yang dikirim antara *client* dan *server* disebut dengan *protocol data units* (PDUs). *Latency* jaringan merupakan waktu yang dibutuhkan untuk sebuah paket PDUs untuk berjalan dari pengirim ke penerima [AJS-12]. Sebuah *game* yang dimainkan melalui internet akan mengalami *round trip delay* berkisar 50 hingga 250 *milliseconds* [BHN-05].

Salah satu tantangan yang dihadapi dalam membuat sebuah game multiplayer online adalah cara mengatasi adanya latency jaringan. Latency yang terjadi pada data paket yang dikirim antara client dan server menyebabkan perbedaan kondisi dan tampilan game pada player satu dengan player yang lainnya, dan dapat memberikan keuntungan atau kerugian tertentu pada pemain tertentu yang dapat mempengaruhi hasil permainan [WJS-09].

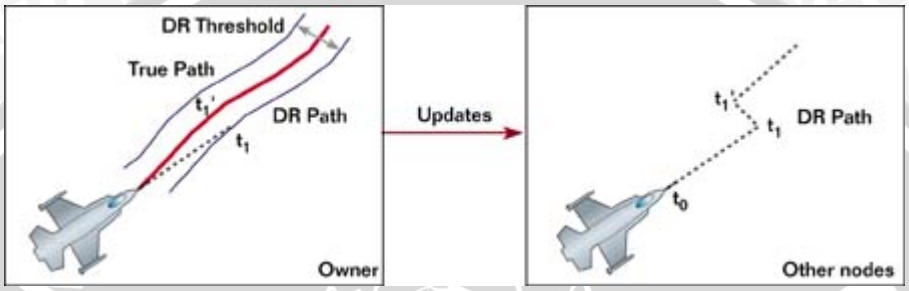
2.2 Dead Reckoning

Distributed Interactive Simulation (DIS) berisi tentang sebuah teknik untuk menyembunyikan adanya *latency* dan mengurangi penggunaan *bandwidth* yang diperlukan, yaitu *Dead Reckoning*. Konsep dasar *dead reckoning* berasal dari sistem navigasi yang digunakan oleh pelaut pada akhir abad ke-15 untuk memperkirakan posisi mereka di lautan [BHN-05]. *Dead Reckoning* juga digunakan oleh *Department of Defends United States of America* untuk simulasi perang [BAH-08].

2.2.1 Konsep *Dead Reckoning*

Dead Reckoning merupakan proses untuk memprediksi letak dimana aktor sekarang berada berdasarkan posisi, kecepatan, dan akselerasi sebelumnya.

Setiap objek aktor yang dikendalikan oleh seorang *player* akan mengirimkan update datanya kepada *server* berupa status *kinematic* dari objek tersebut kemudian menyalurkannya kepada setiap *client* yang terhubung. Status *kinematic* yang diterima meliputi posisi, kecepatan, akselerasi, orientasi, dan kecepatan angular [LYE-11]. Setelah menerima data objek dari *server* setiap *client* akan mulai menggerakkan objek tersebut secara lokal menggunakan algoritma *dead reckoning* sebagai acuan, sambil menunggu *update* data yang masuk berikutnya. Selama objek tersebut tidak melebihi batas prediksi yang sudah ditentukan maka tidak diperlukan adanya update data, seperti yang ditunjukkan pada Gambar 2.1 [BHN-05].

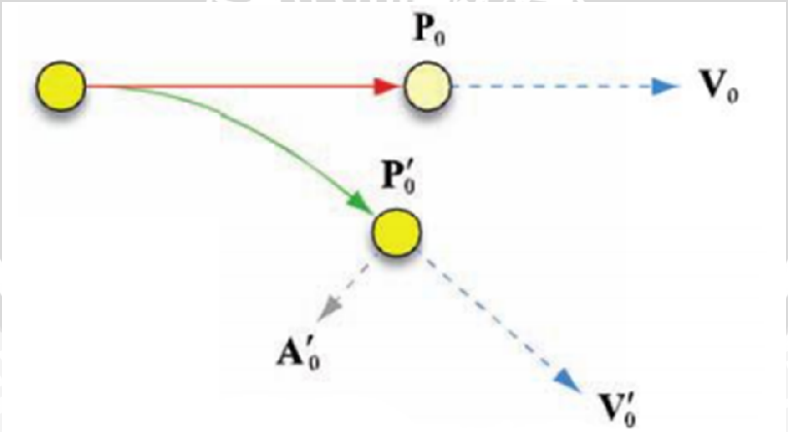


Gambar 2.1 Simulasi *Dead Reckoning*
Sumber : [BHN-05]

Dari data yang diterima maka dapat direpresentasikan objek dengan posisi $\mathbf{P'_0}$ mulai bergerak dengan kecepatan $\mathbf{V'_0}$ dan akselerasi $\mathbf{A'_0}$. Posisi hasil *dead reckoning* $\mathbf{Q_t}$ pada waktu **T** dihitung dengan rumus persamaan 2.1.

$$\boldsymbol{Q_t = P'_0 + V'_0 T + \frac{1}{2} A'_0 T^2} \quad (2.1)$$

Pada data update berikutnya letak posisi objek dapat jauh dari posisi hasil prediksi yang dilakukan. Seperti pada gambar 2.1, hasil prediksi yang ditandai garis merah berbeda arah dengan arah sesungguhnya yang ditandai dengan garis hijau.



Gambar 2.2 Perbedaan *Path* Sesungguhnya dan Hasil Prediksi
Sumber : [LYE-11]

Karena kondisi yang berbeda tersebut dapat terjadi, maka dibutuhkan sebuah algoritma yang digunakan untuk menyelaraskan posisi objek tersebut.

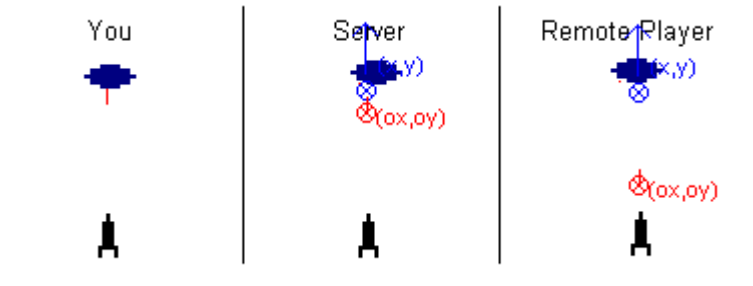
2.2.2 Targeting

Targeting merupakan variasi *dead reckoning* dimana sebuah objek milik *player* lain yang berada dilayar akan bergerak sedikit demi sedikit mendekat ke tempat dimana *player* tersebut terletak seharusnya. Variasi ini sangat cocok digunakan pada jenis objek yang diketahui dimana letak objek tersebut dalam jangka waktu tertentu. Pseudocode dari variasi ini direpresentasikan dalam tabel 2.2.

Tabel 2.1 Pseudocode *Dead reckoning* variasi *Targeting*

```
for each frame {
      target.velocity = target.velocity + target.acceleration;
      target.position = target.position + target.velocity;
      laser.position = (laser.position + target.position) / 2;
}
```

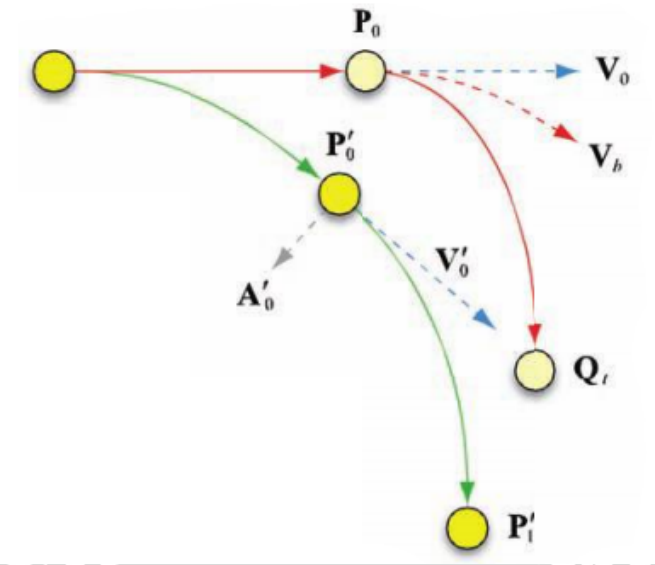
Target objek akan bergerak lebih cepat pada server, dan akan lebih cepat lagi pada *remote client* lainnya. Sehingga dapat mencapai target yang sama pada waktu yang sama, seperti yang ditunjukkan pada Gambar 2.3.



Gambar 2.3 Objek Laser Menuju Target yang Sama
Sumber : [HAC-01]

2.2.3 Projective Velocity Blending

Dengan metode ini lintasan objek hasil prediksi akan disatukan dengan lintasan objek sebenarnya, sehingga objek dapat berjalan kembali sesuai dengan lintasan seharusnya, seperti yang ditunjukkan pada Gambar 2.4.



Gambar 2.4 *Dead Reckoning* dengan *Velocity Blending* Direpresentasikan dengan Lintasan Warna Merah
Sumber : [LYE-11]

Teknik ini digunakan dengan bantuan dari standar *linear interpolation* (lerp), hanya saja dilakukan perbaikan dibagian penyatuan kecepatannya. Metode ini dapat direpresentasikan dengan rumus persamaan 2.2 hingga 2.5.

$$\boldsymbol{V_b = V_0 + (V'_0 - V_0)T'} \quad (2.2)$$

$$\boldsymbol{P_t = P_0 + V_b T_t + \frac{1}{2} A'_0 T^2} \quad (2.3)$$

$$\boldsymbol{P'_t = P'_0 + V'_0 T_t + \frac{1}{2} A'_0 T^2} \quad (2.4)$$

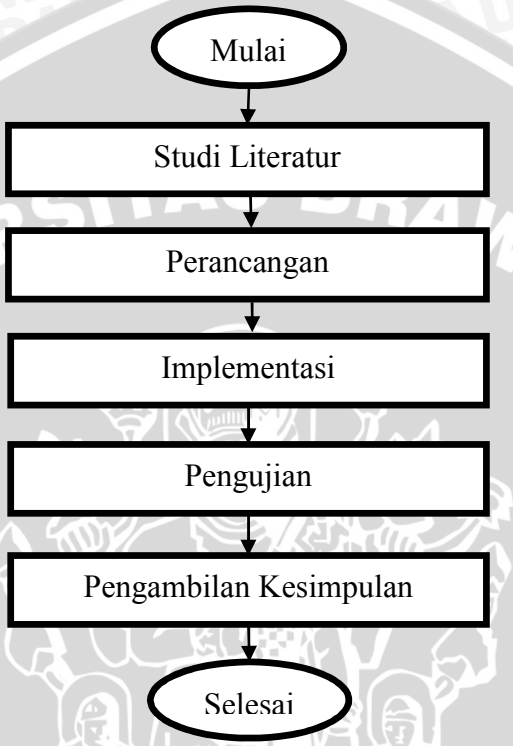
$$\boldsymbol{Q_t = P_t + (P'_t - P_t)T'} \quad (2.5)$$

Keterangan :

V_b : Kecepatan hasil kombinasi antara kecepatan awal dan kecepatan sekarang
V_0 : Kecepatan awal
V'_0 : Kecepatan sesungguhnya
A'_0 : Akselerasi sesungguhnya
T' : Perbandingan total waktu sekarang dan total waktu yang akan datang
T_t : Total waktu antara waktu prediksi sebelumnya dengan waktu sekarang
P_0 : Posisi objek *remote* sekarang
P'_0 : Posisi objek sesungguhnya
P_t : Proyeksi dari letak objek *remote* sekarang berada
P'_t : Proyeksi dari letak posisi sesungguhnya
Q_t : Hasil kalkulasi *Dead Reckoning*

BAB 3 METODE PENELITIAN

Bab ini menjelaskan langkah-langkah yang ditempuh dalam penelitian, meliputi studi literatur, perancangan, implementasi, pengujian, dan pengambilan kesimpulan. Kesimpulan dan saran disertakan sebagai catatan atas metode yang diterapkan dan kemungkinan arah pengembangan selanjutnya. Diagram alir dalam pengerjaan penelitian ini ditunjukkan pada Gambar 3.1 sebagai berikut :



Gambar 3.1 Diagram Alir Runtutan Metode Penelitian

3.1 Studi Literatur

Tahap ini bertujuan untuk mendapatkan teori-teori yang akan menjadi acuan dan penunjang dalam melakukan penelitian Sinkronisasi Objek *Client* pada *Game Multiplayer Online* dengan Menggunakan Metode *Dead Reckoning*. Teori-teori tersebut meliputi :

1. *Game Multiplayer Online*
   a. Jaringan pada *game multiplayer online*
   b. Efek *latency* jaringan pada *game multiplayer online*
2. *Dead Reckoning*
   a. Konsep *Dead Reckoning*
   b. *Targeting*
   c. *Projective Velocity Blending*

3.2 Perancangan

Tahap perancangan merupakan langkah yang dilakukan bertujuan untuk memberikan rencana yang baik dalam melakukan implementasi metode *Dead Reckoning* pada sebuah *game multiplayer online.* Perancangan dilakukan dengan memadukan berbagai ilmu yang telah diperoleh dari dasar-dasar teori yang dijadikan referensi.

3.3 Implementasi

Tahap implementasi dilakukan dengan mengacu pada perancangan yang telah dibuat, meliputi pembuatan contoh *game multiplayer* yang akan digunakan dan pengintegrasian metode *Dead Reckoning* didalamnya.

3.4 Pengujian

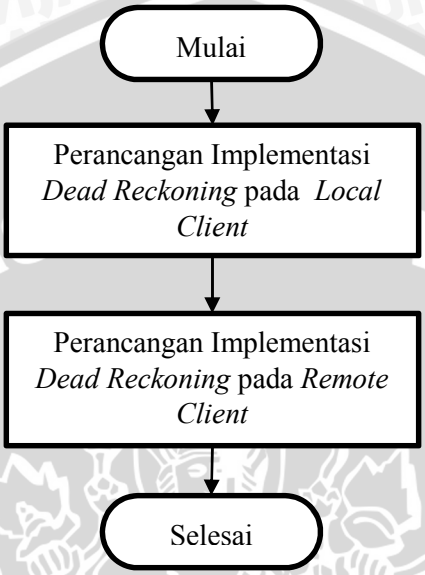
Tahap pengujian dilakukan berdasarkan implementasi yang telah dibuat. Pengujian yang dilakukan meliputi pengujian kinerja, untuk melihat keberhasilan pengimplementasian metode *Dead Reckoning* dalam mengatasi latency dan tingginya *bandwidth* yang digunakan pada sebuah *game multiplayer online*.

3.5 Pengambilan Kesimpulan

Tahap pengambilan kesimpulan merupakan tahap akhir yang dilakukan setelah semua tahapan studi literatur, perancangan, implementasi, dan pengujian selesai dilakukan. Kesimpulan dibuat untuk memberikan jawaban terhadap rumusan masalah yang telah ditetapkan sebelumnya. Disertakan pula pembuatan saran yang ditujukan untuk memperbaiki berbagai kesalahan yang terjadi selama penulisan dan memberikan pertimbangan atau arahan untuk pembangan berikutnya.

BAB 4 PERANCANGAN

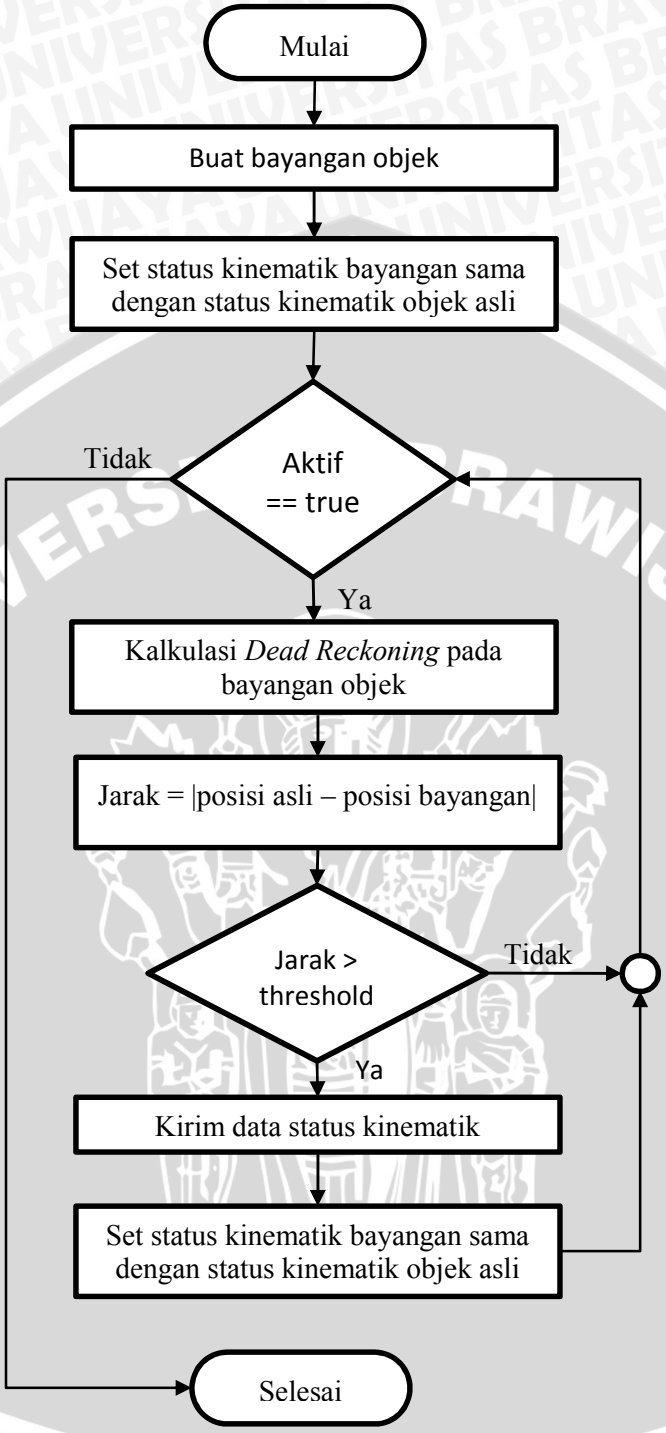
Bab perancangan ini terdiri dari dua tahap, yaitu perancangan implementasi metode *Dead Reckoning* pada *Local Client* dan perancangan implementasi metode *Dead Reckoning* pada *Remote Client*. Tahap – tahap perancangan dijabarkan dalam diagram alir dalam Gambar 4.1.



Gambar 4.1 Diagram Alir Perancangan Implementasi

4.1 Perancangan Implementasi *Dead Reckoning* pada *Local Client*

Pada *local client* implementasi *Dead Reckoning* digunakan sebagai acuan untuk menentukan waktu pengiriman data. Setiap *local client* akan membuat bayangannya sendiri dan melakukan prediksi posisi menggunakan metode *Dead Reckoning* yang sama pada *remote client*. Ketika jarak posisi antara objek asli dan posisi hasil prediksi pada bayangannya telah mencapai *threshold* tertentu maka data status kinematik objek tersebut akan dikirim ke *remote client* dan digunakan sebagai acuan baru juga pada objek bayangan. Tahap-tahap implementasinya dijabarkan pada diagram alir dalam Gambar 4.2.

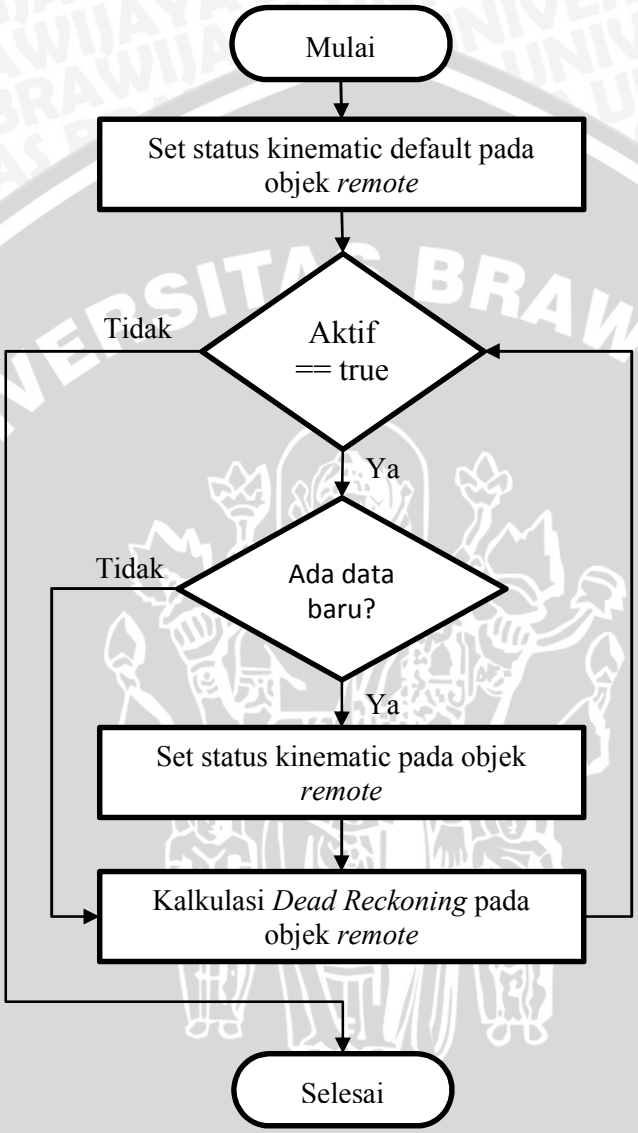


Gambar 4.2 Diagram Alir Implementasi *Dead Reckoning* pada *Local Client*

4.2 Perancangan Implementasi *Dead Reckoning* pada *Remote Client*

Pada *remote client* implementasi *Dead Reckoning* digunakan untuk memprediksi posisi pergerakan objek asli hingga data status kinematik yang baru

dari *local client* sampai pada *remote client*. Objek *remote* akan mengecek terus apakah ada data status kinematic baru yang diterima, jika iya maka status kinematic pada objek *remote* akan diperbarui dengan data tersebut, lalu dilakukan kembali kalkulasi *Dead Reckoning* hingga data yang baru telah diterima. Tahap-tahap implementasinya dijabarkan pada diagram alir dalam Gambar 4.3.



Gambar 4.3 Diagram Alir Implementasi *Dead Reckoning* pada *Remote Client*

4.3 Perhitungan Kalkulasi *Dead Reckoning*

Pada penelitian ini metode *Dead Reckoning* yang digunakan ada dua macam, yaitu metode *Dead Reckoning* asli yang menggunakan interpolasi, dan metode *Dead Reckoning* variasi *Velocity Blending*. Berikut contoh perhitungan dari kedua metode tersebut :

4.3.1 Metode *Dead Reckoning* (Interpolasi)

Contoh kasus, sebuah objek memiliki nilai posisi Vector3(0.1, 0.0, 0.0) bergerak dengan kecepatan *velocity* Vector3(5.0, 0.0, 0.0) dan total waktu yang telah terlampaui sejumlah 0.02s, sedangkan nilai akselerasinya Vector3(0.0, 0.0, 0.0) maka dapat dihitung seperti pada tabel 4.1.

Tabel 4.1 Contoh Perhitungan *Dead Reckoning* (Interpolasi)

Diketahui :

$P_0' = Vector3(0.1, 0.0, 0.0)$
$V_0' = Vector3(5.0, 0.0, 0.0)$
$A_0' = Vector3(0.0, 0.0, 0.0)$
$T = 0.02s$

Jawab :

$$Q_t = P_0' + V_0'T + \frac{1}{2}A_0'T^2$$

$$Q_t = Vector3(0.1, 0.0, 0.0) + (Vector3(5.0, 0.0, 0.0) * 0.02) + (\frac{1}{2} * Vector3(0.0, 0.0, 0.0) * 0.2^2$$

$$Q_t = Vector3(0.1, 0.0, 0.0) + Vector3(0.1, 0.0, 0.0) + Vector3(0.0, 0.0, 0.0)$$

$$Q_t = Vector3(0.2, 0.0, 0.0)$$

4.3.2 Metode *Dead Reckoning* Variasi *Velocity Blending*

Contoh kasus, sebuah objek memiliki nilai posisi awal Vector3(0.0, 0.0, 0.0) dan posisi seharusnya Vector3(0.1, 0.0, 0.0) bergerak dengan kecepatan *velocity* Vector3(5.0, 0.0, 0.0), sedangkan *velocity* awalnya Vector3(0.0, 0.0, 0.0) dan total waktu yang telah terlampaui sejumlah 0.02s, sedangkan nilai akselerasinya Vector3(0.0, 0.0, 0.0) dan *latency* yang terjadi rata-rata 50ms, maka dapat dihitung seperti pada tabel 4.2.

Tabel 4.2 Contoh Perhitungan *Dead Reckoning* Variasi *Velocity Blending*

Diketahui :

$V_0 = Vector3(0.0, 0.0, 0.0)$
$V_0' = Vector3(5.0, 0.0, 0.0)$
$P_0 = Vector3(0.0, 0.0, 0.0)$
$P_0' = Vector3(0.1, 0.0, 0.0)$
$A_0' = Vector3(0.0, 0.0, 0.0)$
$latency = 0.05s$
$T_t = 0.02s$

Jawab :

$$T' = \frac{T_t}{T_t + latency}$$

$$T' = \frac{0.02s}{0.02s + 0.05s} = 0.2857s$$

$$\boxed{V_b = V_0 + (V_0' - V_0)T'}$$

$$V_b = Vector3(0.0, 0.0, 0.0) + \big(Vector3(5.0, 0.0, 0.0) - Vector3(0.0, 0.0, 0.0)\big) * 0.2857 = Vector3(1.4285, 0.0, 0.0)$$

$$\boxed{P_t = P_0 + V_b T_t + \frac{1}{2} A_0' T^2}$$

$$P_t = Vector3(0.0, 0.0, 0.0) + (Vector3(1.4285, 0.0, 0.0) * 0.02) + (\frac{1}{2} * Vector3(0.0, 0.0, 0.0) * 0.02^2 = Vector3(0.02857, 0.0, 0.0)$$

$$\boxed{P_t' = P_0' + V_0' T_t + \frac{1}{2} A_0' T^2}$$

$$P_t' = Vector3(0.1, 0.0, 0.0) + (Vector3(5.0, 0.0, 0.0) * 0.02) + (\frac{1}{2} * Vector3(0.0, 0.0, 0.0) * 0.02^2 = Vector3(0.2, 0.0, 0.0)$$

$$\boxed{Q_t = P_t + (P_t' - P_t)T'}$$

$$Q_t = Vector3(0.02857, 0.0, 0.0) + ((Vector3(0.2, 0.0, 0.0) - Vector3(0.02857, 0.0, 0.0)) * 0.2857)$$

$$Q_t = \text{Vector3}(0.07755, 0.0, 0.0)$$

BAB 5 IMPLEMENTASI

Bab ini membahas mengenai implementasi metode *Dead Reckoning* berdasarkan hasil perancangan yang telah dibuat. Pembahasan terdiri dari penjelasan tentang spesifikasi sistem, batasan-batasan dalam implementasi, dan implementasi rancangan algoritma.

5.1 Penentuan Spesifikasi Sistem

Hasil perancangan pada bab 4 menjadi acuan untuk melakukan implementasi algoritma *Dead Reckoning* baik pada *local client* maupun *remote client* yang dapat berfungsi sesuai dengan kebutuhan. Spesifikasi sistem untuk implementasi algoritma dijelaskan pada spesifikasi perangkat keras dan perangkat lunak.

5.1.1 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras yang dipakai dalam proses pengembangan dijelaskan dalam Tabel 5.1.

Tabel 5.1 Spesifikasi Perangkat Keras Komputer

Nama Komponen	Spesifikasi
Prosesor	2.10 GHz *Intel Core 2 Duo*
Memori	4 GB DDR2
Kartu Grafis	Mobile Intel(R) 4 Series Express Chipset Family

5.1.2 Spesifikasi Perangkat Lunak

Spesifikasi perangkat lunak yang dipakai dalam proses pengembangan dijelaskan dalam Tabel 5.2.

Tabel 5.2 Spesifikasi Lingkungan Perangkat Lunak Komputer

Perangkat Lunak	Spesifikasi
Operating System	*Microsoft Windows 8.1 Pro 64-bit*
Integrated Development Environment	*Visual Studio Pro 2013*
Programming Language	*C#*
Software Development Kit	*Unity 5.0.1f*
Graphic Editor	*Adobe Photoshop CS 5*

5.2 Implementasi Metode *Dead Reckoning*

Metode *Dead Reckoning* digunakan untuk melakukan prediksi pergerakan objek asli *local client* pada *remote client* dan juga pada objek bayangan milik objek

asli itu sendiri. Variasi *Dead Reckoning* yang diimplementasikan ada dua macam variasi, yaitu *Dead Reckoning* (Interpolasi) dan *Dead Reckoning* variasi *Velocity Blending*. Data yang diperlukan untuk melakukan kalkulasi adalah status kinematik dari objek asli, yaitu posisi, kecepatan (*velocity*), sudut euler, kecepatan angular. Kalkulasi *Dead Reckoning* dilakukan dua kali, yaitu pada posisi dan rotasinya. Setiap kalkulasi menghasilkan data berupa *Vector3* untuk digunakan sebagai posisi baru dan rotasi baru. Implementasi dari metode *Dead Reckoning* (Interpolasi) ditunjukkan dalam Tabel 5.3, sedangkan metode *Dead Reckoning* variasi *Velocity Blending* ditunjukkan dalam Tabel 5.4.

Tabel 5.3 Implementasi Metode *Dead Reckoning* (Interpolasi)

```
1  float Tt = (System.DateTime.Now.Ticks - t0) / 10000000f;
2  Vector3 Qt = p1 + targetobject.transform.right * v1 * Tt;
3  targetobject.transform.position = Qt;
4  Vector3 aqt = ae1 + vector1rotation * av1 * Tt;
5  targetobject.transform.eulerAngles = aqt;
```

Penjelasan :

Baris 1 : Mengambil total waktu antara *update* data terakhir dan waktu sekarang
Baris 2 : Melakukan kalkulasi *Dead Reckoning*
Baris 3 : Mengeset posisi objek sesuai dengan hasil kalkulasi *Dead Reckoning*
Baris 4 : Melakukan kalkulasi *Dead Reckoning* untuk rotasi
Baris 5 : Mengeset rotasi objek sesuai dengan hasil kalkulasi *Dead Reckoning*

Tabel 5.4 Implementasi Metode *Dead Reckoning* variasi *Velocity Blending*

```
1  float Tt = (System.DateTime.Now.Ticks - t0) / 10000000f;
2  float TAksen = Tt / (Tt + (latency/1000f));
3  float vb = v0 + ((v1 - v0) * TAksen);
4  Vector3    pt0    =    targetobject.transform.position    +
5  targetobject.transform.right * vb * Tt;
6  Vector3 pt1 = p1 + targetobject.transform.right * v1 * Tt;
7  Vector3 qt = pt0 + ((pt1 - pt0) * TAksen);
8  targetobject.transform.position = qt;
9  v0 = v1;
10 float avb = av0 + ((av1 - av0) * TAksen);
11 Vector3 apt0 = drrotation + vector1rotation * avb * Tt;
12 Vector3 apt1 = ae1 + vector1rotation * av1 * Tt;
13 Vector3 aqt = apt0 + ((apt1 - apt0) * TAksen);
14 targetobject.transform.eulerAngles = aqt;
15 av0 = av1;
16 drrotation = aqt;
```

Penjelasan :

Baris 1 : Mengambil total waktu antara *update* data terakhir dan waktu sekarang
Baris 2 : Menghitung perbandingan total waktu sekarang dengan total waktu seharusnya
Baris 3 : Menghitung *velocity blending*
Baris 4-5 : Melakukan kalkulasi *Dead Reckoning* berdasarkan posisi objek sekarang

Baris 6 : Melakukan kalkukasi *Dead Reckoning* berdasarkan posisi objek seharusnya
Baris 7 : Melakukan *position blending* dari hasil kalkulasi *Dead Reckoning* yang telah dilakukan
Baris 8 : Mengeset posisi objek sesuai dengan hasil kalkulasi
Baris 9 : Mengeset nilai v0 (*velocity* sebelumnya) sama dengan nilai v1 (*velocity* sekarang)
Baris 10 : Menghitung *velocity blending* untuk rotasi
Baris 11 : Melakukan kalkulasi *Dead Reckoning* berdasarkan rotasi objek sekarang
Baris 12 : Melakukan kalkulasi *Dead Reckoning* berdasarkan rotasi objek seharusnya
Baris 13 : Melakukan *rotation blending* dari hasil kalkulasi *Dead Reckoning* yang telah dilakukan
Baris 14 : Mengeset rotasi objek sesuai dengan hasil kalkulasi
Baris 15 : Mengeset nilai av0 (*angular velocity* sebelumnya) sama dengan nilai av1 (*angular velocity* sekarang)
Baris 16 : Menyimpan nilai rotasi hasil kalkulasi sebagai referensi untuk kalkulasi berikutnya

5.3 Implementasi *Dead Reckoning* pada *Local Client*

Objek asli pada *local client* akan membuat bayangannya sendiri yang digunakan sebagai acuan untuk menentukan waktu pengiriman data ke server. Objek bayangan ini akan berada pada posisi hasil prediksi *Dead Reckoning*, sedangkan objek asli akan tetap berjalan sesuai input dari *player.* Jika jarak antara objek bayangan dan objek asli telah mencapai batas (*threshold*) tertentu maka objek asli akan mengirimkan data posisi baru ke server, kemudian objek bayangan akan menyesuaikan diri sesuai dengan data yang dikirimkan. Implementasi pada *Local Client* ditunjukkan pada Tabel 5.5.

Tabel 5.5 Implementasi Metode *Dead Reckoning* pada *Local Client*

```
1   public override void OnStartLocalPlayer()
2   {
3         player.enabled = true;
4         drobject = Instantiate(prefdrobject) as Transform;
5         drobject.transform.position = transform.position;
6         drobject.rotation = transform.rotation;
7         TransmitData();
8   }
9
10  [Client]
11  public void TransmitData()
12  {
13        t0 = System.DateTime.Now.Ticks;
14        this.latency = networkmanager.client.GetRTT();
15        p1 = transform.position;
16        v1 = player.movespeed;
17        ae1 = transform.eulerAngles;
18        av1 = player.currentrotatespeed;
```

```
19          CmdProvidePositionToServer(p1, v1, ae1, av1);
20      }
21
22      void Update()
23      {
24          SetDRPosition(drobject);
25          if                  (Vector3.Distance(transform.position,
26      drobject.transform.position) > threshold)
27          {
28              TransmitData();
29          }
        }
```

Penjelasan :

Baris 3 : Aktifkan script pengontrol objek asli
Baris 4-6 : Membuat objek bayangan dan menyesuaikan posisi dan rotasi awalnya
Baris 7 : Mengirim data awal ke server
Baris 13-18 : Mengirim data posisi, kecepatan, sudut euler, dan kecepatan rotasi ke server
Baris 23 : Mengeset posisi *Dead Reckoning* pada objek bayangan
Baris 24-28 : Jika jarak antara objek asli dan objek bayangan telah mencapai *threshold* maka kirimkan data

5.4 Implementasi *Dead Reckoning* pada *Remote Client*

Pada *Remote Client* objek hanya akan menerima data status kinematic dari objek asli pada *client* yang lain, kemudian melakukan kalkulasi *Dead Reckoning* lagi berdasarkan data status yang baru tersebut. Implementasi pada *Remote Client* ditunjukkan dalam Tabel 5.6.

Tabel 5.6 Implementasi Metode *Dead Reckoning* pada *Remote Client*

```
1       [Command]
2       void  CmdProvidePositionToServer(Vector3  position,  float
3       velocity, Vector3 eulerangles, float angularvelocity)
4       {
5             this.latency = networkmanager.client.GetRTT();
6             p1 = position;
7             v1 = velocity;
8             ae1 = eulerangles;
9             av1 = angularvelocity;
10      }
11
12      void Update()
13      {
14            SetDRPosition(transform);
15      }
16
17      void OnP1Changed(Vector3 pos)
18      {
19          t0 = System.DateTime.Now.Ticks;
20          p1 = pos;
21      }
```

Penjelasan :

Baris 1-10 : Menerima data baru yang dikirimkan dari server
Baris 14 : Mengeset posisi *Dead Reckoning* pada objek *remote*
Baris 19 : Mengambil waktu *update* sekarang
Baris 20 : Mengeset nilai p1 sesuai dengan posisi yang dikirimkan

BAB 6 PENGUJIAN DAN ANALISIS

Bab ini dilakukan proses pengujian dan analisis terhadap implementasi metode *Dead Reckoning* pada sebuah *game multiplayer online*. Pengujian dilakukan dengan menguji tiap kasus uji dan pencatatan hasil berdasarkan parameter pengujian. Parameter pengujian kinerja berupa jumlah pengiriman data dan kesesuaian posisi prediksi dengan posisi objek asli pada *remote client*.

6.1 Uji Coba dan Hasil Kinerja

Pengujian dilakukan untuk mengetahui kinerja dari pengimplementasian metode *Dead Reckoning* dalam mengatasi *latency* dan tingginya *bandwidth* yang digunakan pada sebuah *game multiplayer online*. Kinerja dari pengimplementasian metode ini dapat ditinjau dari jumlah pengiriman data yang terjadi dalam sejumlah waktu tertentu dan kesesuaian posisi prediksi dengan posisi objek asli *remote client* pada setiap waktu tersebut. Pengujian dilakukan dengan menggunakan simulator jaringan milik Unity3D. Pengujian dari program dilakukan pada perangkat dengan spesifikasi yang dijelaskan pada Table 6.1.

Tabel 6.1 Spesifikasi Komponen Perangkat Pengujian

Nama Komponen	Spesifikasi
Prosesor	2.10 GHz *Intel Core 2 Duo*
Memori	4 GB DDR2
Kartu Grafis	Mobile Intel(R) 4 Series Express Chipset Family
Tools	*Unity 5.0.1f*

6.1.1 Pengujian Kinerja Berdasarkan Jumlah Pengiriman Data

Pengujian kinerja berdasarkan jumlah pengiriman data merupakan pengujian yang dilakukan untuk mengetahui berapa kali jumlah pengiriman data yang dilakukan oleh *client* pada *server* dalam kurun waktu tertentu.

6.1.1.1 Tujuan

Tujuan pengujian kinerja berdasarkan jumlah pengiriman data adalah untuk mengetahui apakah program dapat berjalan dan memberikan hasil optimal dalam mengatasi jumlah *bandwidth* yang tinggi pada sebuah *game multiplayer online*. Kesuksesan pengujian ditunjukkan dengan diperolehnya nilai jumlah pengiriman data yang optimal.

6.1.1.2 Prosedur

Prosedur pengujian kinerja berdasarkan jumlah pengiriman data dilakukan dengan menjalankan program pada dua kondisi gerak utama yaitu lurus dan melingkar. Pada setiap kondisi gerak dilakukan pengambilan data dengan

melakukan percobaan pada *threshold* jarak *Dead Reckoning* 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09 dan 0.1 yang disimulasikan pada *latency* sebesar 50, 100, 150, 200, dan 250 *milliseconds* dan berjalan selama 1000 *frame*. Kasus uji percobaan kinerja berdasarkan jumlah pengiriman data adalah sebagai berikut :

1. Kasus Uji pada Gerak Lurus

Penerapan metode konvensional pada pengujian berdasarkan jumlah pengiriman data pada kasus uji gerak lurus tidak menggunakan parameter *threshold*, seperti yang ditunjukkan pada Tabel 6.2, pada rata-rata *latency* 50ms rata-rata jumlah pengiriman data yang dilakukan adalah 1000 kali, pada rata-rata *latency* 100ms-pun sama, dan begitu seterusnya. Sedangkan untuk penerapan metode *Dead Reckoning* terdapat *threshold* seperti yang ditunjukkan pada Tabel 6.3 dan Tabel 6.4. Pada Tabel 6.3, penerapan metode *Dead Reckoning* (Interpolasi) dengan *threshold* sebesar 0.01 dan rata-rata *latency* 50ms, rata-rata jumlah pengiriman data yang terjadi adalah 138.5 kali, sedangkan pada *threshold* sebesar 0.02 dan rata-rata *latency* 50ms, rata-rata jumlah pengiriman data yang terjadi adalah 141.7 kali, dan begitu seterusnya. Pada Tabel 6.4, penerapan metode *Dead Reckoning* variasi *Velocity Blending* dengan *threshold* sebesar 0.01 dan rata-rata *latency* 50ms, rata-rata jumlah pengiriman data yang terjadi adalah 939.4 kali, sedangkan pada *threshold* sebesar 0.02 dan rata-rata *latency* 50ms, rata-rata jumlah pengiriman data yang terjadi adalah 899.9 kali, dan begitu seterusnya.

Tabel 6.2 Pengujian Kinerja Berdasarkan Jumlah Pengiriman Data pada Gerak Lurus Menggunakan Metode Konvensional

Threshold	Rata-rata Latency (ms)				
	50	100	150	200	250
Konvensional (Tanpa Prediksi)	1000	1000	1000	1000	1000

Tabel 6.3 Pengujian Kinerja Berdasarkan Jumlah Pengiriman Data pada Gerak Lurus Menggunakan Metode *Dead Reckoning* (Interpolasi)

Threshold	Rata-rata Latency (ms)				
	50	100	150	200	250
0.01	138.5	94.8	75.1	55.1	64.2
0.02	141.7	94.4	67.2	59.3	62.9
0.03	138.3	95	69.4	51.6	64.1
0.04	140.9	94.7	67.3	51.2	60.9
0.05	138.3	92.4	66.9	51	62
0.06	139.4	93.1	66.8	51.9	62.2

0.07	139.4	93.3	67.4	67.6	58.4
0.08	139.4	93.9	64.6	72.1	61
0.09	137.6	93.5	67.6	73.5	60.1
0.1	139.3	95.4	67.3	75.2	61.4

Tabel 6.4 Pengujian Kinerja Berdasarkan Jumlah Pengiriman Data pada Gerak Lurus Menggunakan Metode *Dead Reckoning* Variasi *Velocity Blending*

Threshold	**Rata-rata Latency (ms)**				
	50	**100**	**150**	**200**	**250**
0.01	939.4	997.4	999.9	999.9	1000
0.02	899.9	994	997.9	998	999.8
0.03	828.9	986.3	992.2	994.3	997.2
0.04	733.8	964.8	980.5	994.7	996.7
0.05	660.5	895.5	962.8	978.6	987.3
0.06	627.8	882.5	944.7	966.1	971.3
0.07	440.1	833.4	919	953.2	957.4
0.08	518.5	768.8	881.9	922.3	947.6
0.09	57.6	734.6	853.4	926.7	942.7
0.1	70.8	1	799.5	895.8	925.9

2. Kasus Uji pada Gerak Melingkar

Penerapan metode konvensional pada pengujian berdasarkan jumlah pengiriman data pada kasus uji gerak melingkar tidak menggunakan parameter *threshold*, seperti yang ditunjukkan pada Tabel 6.5, pada rata-rata *latency* 50ms rata-rata jumlah pengiriman data yang dilakukan adalah 1000 kali, pada rata-rata *latency* 100ms-pun sama, dan begitu seterusnya. Sedangkan untuk penerapan metode *Dead Reckoning* terdapat *threshold* seperti yang ditunjukkan pada Tabel 6.6 dan Tabel 6.7. Pada Tabel 6.6, penerapan metode *Dead Reckoning* (Interpolasi) dengan *threshold* sebesar 0.01 dan rata-rata *latency* 50ms, rata-rata jumlah pengiriman data yang terjadi adalah 312.6 kali, sedangkan pada *threshold* sebesar 0.02 dan rata-rata *latency* 50ms, rata-rata jumlah pengiriman data yang terjadi adalah 195.6 kali, dan begitu seterusnya. Pada Tabel 6.7, penerapan metode *Dead Reckoning* variasi *Velocity Blending* dengan *threshold* sebesar 0.01 dan rata-rata *latency* 50ms, rata-rata jumlah pengiriman data yang terjadi adalah 952.5 kali, sedangkan pada *threshold* sebesar 0.02 dan rata-rata *latency* 50ms,

rata-rata jumlah pengiriman data yang terjadi adalah 892.6 kali, dan begitu seterusnya.

Tabel 6.5 Pengujian Kinerja Berdasarkan Jumlah Pengiriman Data pada Gerak Melingkar Menggunakan Metode Konvensional

Threshold	Rata-rata Latency (ms)				
	50	100	150	200	250
Konvensional (Tanpa Prediksi)	1000	1000	1000	1000	1000

Tabel 6.6 Pengujian Kinerja Berdasarkan Jumlah Pengiriman Data pada Gerak Melingkar Menggunakan Metode *Dead Reckoning* (Interpolasi)

Threshold	Rata-rata Latency (ms)				
	50	100	150	200	250
0.01	312.6	310.1	326.5	339.8	324.2
0.02	195.6	202.1	198.4	204.6	210.3
0.03	173.5	180	178.8	177.9	171.1
0.04	157.9	160.4	159.3	158.6	152.2
0.05	157.7	147.2	150.6	146.7	143.1
0.06	156.7	138.8	137.8	137	131.6
0.07	179.6	134.6	129.2	126.6	125.4
0.08	157	128.1	123.7	118.4	115.5
0.09	156.7	126.7	117.6	118.1	109.7
0.1	156	123	120	116	109.4

Tabel 6.7 Pengujian Kinerja Berdasarkan Jumlah Pengiriman Data pada Gerak Melingkar Menggunakan Metode *Dead Reckoning* Variasi *Velocity Blending*

Threshold	Rata-rata Latency (ms)				
	50	100	150	200	250
0.01	952.5	995.9	999.2	999.9	999.9
0.02	892.6	990.4	997.7	998.8	999.3
0.03	831.5	980.8	992.4	996.2	997.4
0.04	748.4	963.8	979	994.4	993.2

0.05	673.4	926.3	963.7	983.9	987.9
0.06	650.3	868.9	961.3	960.1	970.5
0.07	598.5	803.5	895	945.2	960.2
0.08	521.9	781.8	876.2	927.6	952.5
0.09	464.8	741.1	851	896.7	936.3
0.1	429.1	716.2	813.3	881.4	923.7

6.1.2 Pengujian Kinerja Berdasarkan Kesesuaian Prediksi

Pengujian kinerja berdasarkan kesesuaian prediksi merupakan pengujian yang dilakukan untuk mengetahui kesesuaian posisi antara objek asli dan objek *remote* yang ada pada *client* yang lain.

6.1.2.1 Tujuan

Tujuan pengujian kinerja berdasarkan kesesuaian prediksi adalah untuk mengetahui seberapa tepat hasil prediksi yang didapat dengan menggunakan metode *Dead Reckoning*, sehingga objek pada *local client* akan selaras dengan objek pada *remote client*. Kesuksesan pengujian ditunjukkan dengan diperolehnya nilai jarak yang optimal antar objek.

6.1.2.2 Prosedur

Prosedur pengujian kinerja berdasarkan kesesuaian prediksi dilakukan dengan menjalankan program pada dua kondisi gerak utama yaitu lurus dan melingkar. Pada setiap kondisi gerak dilakukan pengambilan data dengan melakukan percobaan pada *threshold* jarak *Dead Reckoning* 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, dan 0.1 yang disimulasikan pada *latency* sebesar 50, 100, 150, 200, dan 250 *milliseconds* dan berjalan selama 60 detik. Data yang diambil berupa selisih rata-rata posisi objek pada kurun waktu tersebut. Kasus uji percobaan kinerja berdasarkan kesesuaian prediksi adalah sebagai berikut :

1. Kasus Uji pada Gerak Lurus

Penerapan metode konvensional pada pengujian berdasarkan kesesuaian prediksi pada kasus uji gerak lurus tidak menggunakan parameter *threshold*, seperti yang ditunjukkan pada Tabel 6.8, pada rata-rata *latency* 50ms rata-rata jarak perbedaan posisi objek asli dan objek *remote* adalah 0.35414, sedangkan pada rata-rata *latency* 100ms adalah 0.74144, dan begitu seterusnya. Sedangkan untuk penerapan metode *Dead Reckoning* terdapat *threshold* seperti yang ditunjukkan pada Tabel 6.9 dan Tabel 6.10. Pada Tabel 6.9, penerapan metode *Dead Reckoning* (Interpolasi) dengan *threshold* sebesar 0.01 dan rata-rata *latency* 50ms rata-rata jarak perbedaan posisi objek asli dan objek *remote* adalah 0.20584, sedangkan pada *threshold* sebesar 0.02 dan rata-rata *latency* 50ms adalah 0.21212, dan begitu seterusnya. Pada Tabel 6.10, penerapan metode *Dead*

Reckoning variasi *Velocity Blending* dengan *threshold* sebesar 0.01 dan rata-rata *latency* 50ms rata-rata jarak perbedaan posisi objek asli dan objek *remote* adalah 0.11575, sedangkan pada *threshold* sebesar 0.02 dan rata-rata *latency* 50ms adalah 0.13501, dan begitu seterusnya.

Tabel 6.8 Pengujian Kinerja Berdasarkan Kesesuain Prediksi pada Gerak Lurus Menggunakan Metode Konvensional

Threshold	Rata-rata Latency (ms)				
	50	100	150	200	250
Konvensional (Tanpa Prediksi)	0.35414	0.74144	0.87299	1.09190	1.28189

Tabel 6.9 Pengujian Kinerja Berdasarkan Kesesuain Prediksi pada Gerak Lurus Menggunakan Metode *Dead Reckoning* (Interpolasi)

Threshold	Rata-rata Latency (ms)				
	50	100	150	200	250
0.01	0.20584	0.58478	0.91161	0.81077	1.27688
0.02	0.21212	0.54958	0.73422	1.08811	1.28189
0.03	0.20779	0.56922	0.67540	0.89835	0.99489
0.04	0.18691	0.48775	0.71897	0.85428	0.87232
0.05	0.19394	0.55829	0.70342	0.87314	1.14727
0.06	0.19181	0.58044	0.70867	0.82889	1.09938
0.07	0.22242	0.52860	0.74372	0.89527	1.07810
0.08	0.20771	0.50067	0.72896	0.95509	0.96371
0.09	0.22148	0.53550	0.67419	0.82895	0.92245
0.1	0.22439	0.49441	0.62651	0.98052	1.08202

Tabel 6.10 Pengujian Kinerja Berdasarkan Kesesuain Prediksi pada Gerak Lurus Menggunakan Metode *Dead Reckoning* Variasi *Velocity Blending*

Threshold	Rata-rata Latency (ms)				
	50	100	150	200	250
0.01	0.11575	0.15189	0.16577	0.27515	0.37618
0.02	0.13501	0.14025	0.14336	0.31271	0.38227
0.03	0.13577	0.12587	0.16073	0.24595	0.40955

0.04	0.10094	0.14222	0.20998	0.26401	0.48975
0.05	0.11551	0.11258	0.17147	0.36158	0.38937
0.06	0.08681	0.14701	0.16018	0.28348	0.49166
0.07	0.12696	0.14125	0.20142	0.24376	0.31402
0.08	0.05843	0.14544	0.15667	0.26229	0.31325
0.09	0.06981	0.12690	0.15747	0.29948	0.36571
0.1	0.11349	0.15654	0.18078	0.25567	0.38644

2. Kasus Uji pada Gerak Melingkar

Penerapan metode konvensional pada pengujian berdasarkan kesesuaian prediksi pada kasus uji gerak melingkar tidak menggunakan parameter *threshold*, seperti yang ditunjukkan pada Tabel 6.11, pada rata-rata *latency* 50ms rata-rata jarak perbedaan posisi objek asli dan objek *remote* adalah 0.29488, sedangkan pada rata-rata *latency* 100ms adalah 0.68041, dan begitu seterusnya. Sedangkan untuk penerapan metode *Dead Reckoning* terdapat *threshold* seperti yang ditunjukkan pada Tabel 6.11 dan Tabel 6.12. Pada Tabel 6.11, penerapan metode *Dead Reckoning* (Interpolasi) dengan *threshold* sebesar 0.01 dan rata-rata *latency* 50ms rata-rata jarak perbedaan posisi objek asli dan objek *remote* adalah 0.16043, sedangkan pada *threshold* sebesar 0.02 dan rata-rata *latency* 50ms adalah 0.35480, dan begitu seterusnya. Pada Tabel 6.12, penerapan metode *Dead Reckoning* variasi *Velocity Blending* dengan *threshold* sebesar 0.01 dan rata-rata *latency* 50ms rata-rata jarak perbedaan posisi objek asli dan objek *remote* adalah 0.09828, sedangkan pada *threshold* sebesar 0.02 dan rata-rata *latency* 50ms adalah 0.06655, dan begitu seterusnya.

Tabel 6.11 Pengujian Kinerja Berdasarkan Kesesuain Prediksi pada Gerak Melingkar Menggunakan Metode Konvensional

Threshold	**Rata-rata Latency (ms)**				
	50	**100**	**150**	**200**	**250**
Konvensional (Tanpa Prediksi)	0.29488	0.68041	1.13417	1.42146	1.75303

Tabel 6.12 Pengujian Kinerja Berdasarkan Kesesuain Prediksi pada Gerak Melingkar Menggunakan Metode *Dead Reckoning* (Interpolasi)

Threshold	**Rata-rata Latency (ms)**				
	50	**100**	**150**	**200**	**250**
0.01	0.16043	0.36712	0.45474	0.56728	0.90572

0.02	0.35480	0.37358	1.01696	0.84252	0.97719
0.03	0.16160	0.37770	0.55145	0.65322	0.98973
0.04	0.17697	0.73324	0.55857	1.08963	1.06111
0.05	0.40678	0.47209	0.61347	0.84384	1.03349
0.06	0.18221	0.74399	0.51231	0.80814	1.64358
0.07	0.58614	0.43094	1.28015	0.80048	1.02236
0.08	0.37343	0.74931	1.30518	0.96049	1.04720
0.09	0.61883	0.79140	1.57234	0.80761	1.13503
0.1	0.25858	0.49313	0.97050	0.87337	0.90470

Tabel 6.13 Pengujian Kinerja Berdasarkan Kesesuain Prediksi pada Gerak Melingkar Menggunakan Metode *Dead Reckoning* Variasi *Velocity Blending*

Threshold	**Rata-rata Latency (ms)**				
	50	**100**	**150**	**200**	**250**
0.01	0.09828	0.10378	0.15451	0.24725	0.32057
0.02	0.06655	0.10274	0.13677	0.22905	0.43192
0.03	0.10197	0.09626	0.14620	0.21696	0.42451
0.04	0.09056	0.10541	0.13084	0.20182	0.47283
0.05	0.08829	0.11182	0.11815	0.23243	0.31928
0.06	0.07475	0.10701	0.14754	0.22528	0.31535
0.07	0.09088	0.08701	0.14131	0.26668	0.40238
0.08	0.10714	0.10627	0.11628	0.21056	0.52572
0.09	0.08645	0.09963	0.17697	0.27744	0.45829
0.1	0.09555	0.10208	0.13746	0.19258	0.33369

6.2 Analisis Uji Coba

Analisis bertujuan untuk mendapatkan kesimpulan dari hasil pengujian implementasi metode *Dead Reckoning*. Berdasarkan pengujian kinerja yang dilakukan maka analisis dapat dibagi menjadi dua bagian, yaitu analisis kinerja berdasarkan *threshold* dan analisis kinerja berdasarkan *latency*.

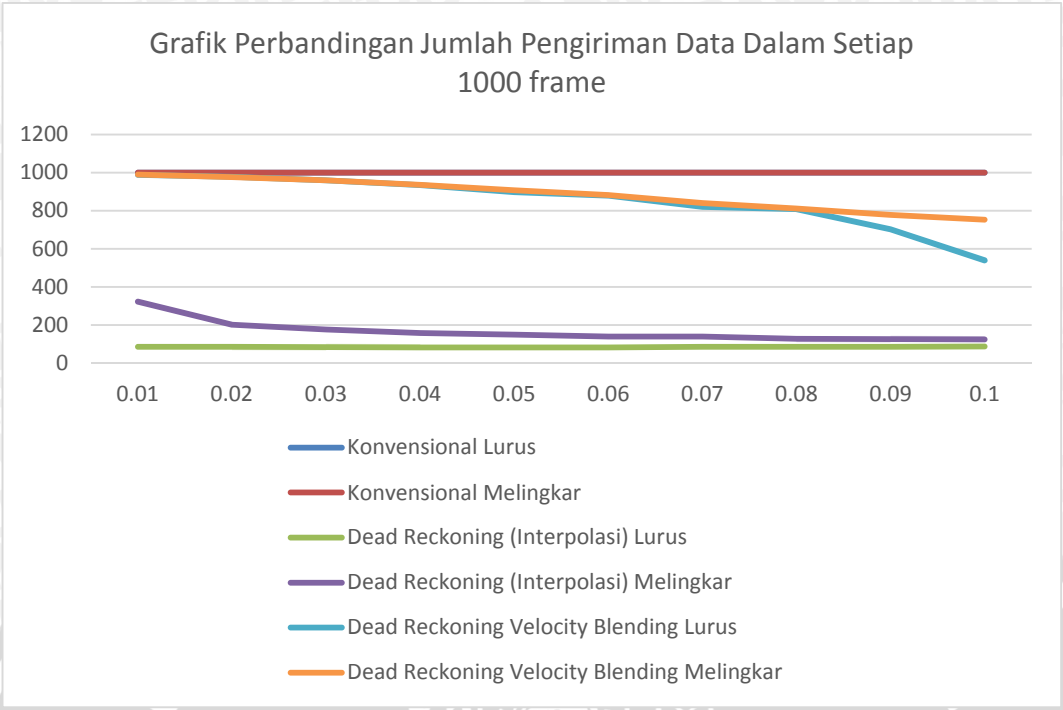
6.2.1 Analisis Kinerja Berdasarkan *Threshold*

Analisis kinerja berdasarkan *threshold* pada pengujian jumlah pengiriman data, rata-rata jumlah pengiriman data yang terjadi ditunjukkan pada Tabel 6.14.

Tabel 6.14 Rata-rata Jumlah Pengiriman Data Berdasarkan *Threshold*

Threshold	Konvensional		*Dead Reckoning* (Interpolasi)		*Dead Reckoning* Variasi *Velocity Blending*	
	Lurus	Melingkar	Lurus	Melingkar	Lurus	Melingkar
0.01	1000	1000	85.54	322.64	987.32	989.48
0.02	1000	1000	85.1	202.2	977.92	975.76
0.03	1000	1000	83.68	176.26	959.78	959.66
0.04	1000	1000	83	157.68	934.1	935.76
0.05	1000	1000	82.12	149.06	896.94	907.04
0.06	1000	1000	82.68	140.38	878.48	882.22
0.07	1000	1000	85.22	139.08	820.62	840.48
0.08	1000	1000	86.2	128.54	807.82	812
0.09	1000	1000	86.46	125.76	703	777.98
0.1	1000	1000	87.72	124.88	538.6	752.74
Rata-rata	**1000**	**1000**	**84.77**	**166.65**	**850.46**	**883.31**

Hasil pengujian kinerja berdasarkan jumlah pengiriman data dapat diketahui bahwa dalam kurun 1000 frame rata-rata pengiriman data yang terjadi dengan menggunakan metode konvensional adalah sejumlah 1000 kali baik pada gerak lurus maupun gerak melingkar, sedangkan dengan menggunakan metode *Dead Reckoning* (Interpolasi) adalah sejumlah 84.77 kali pada gerak lurus dan 166.65 kali pada gerak melingkar, lalu dengan menggunakan metode *Dead Reckoning* variasi *Velocity Blending* adalah sejumlah 850.46 kali pada gerak lurus dan 883.31 kali pada gerak melingkar. Grafik perbandingan jumlah pengiriman data ditunjukkan dalam Gambar 6.1.



Gambar 6.1 Grafik Perbandingan Jumlah Pengiriman Data Berdasarkan *Threshold* Dalam Setiap 1000 *Frame*

Gambar Grafik 6.1 menunjukkan bahwa pada kurun setiap 1000 frame tingkat jumlah pengiriman data pada metode *Dead Reckoning* baik interpolasi maupun variasi *Velocity Blending* dipengaruhi oleh *threshold* dan tipe pergerakannya, dimana semakin besar *threshold* maka semakin kecil tingkat jumlah pengiriman datanya dan berdasarkan tipe pergerakannya jumlah pengiriman data pada tipe pergerakan lurus lebih kecil dari pada pergerakan melingkar. Dari grafik tersebut juga dapat diketahui bahwa metode *Dead Reckoning* (interpolasi) memiliki jumlah pengiriman data yang paling optimal.

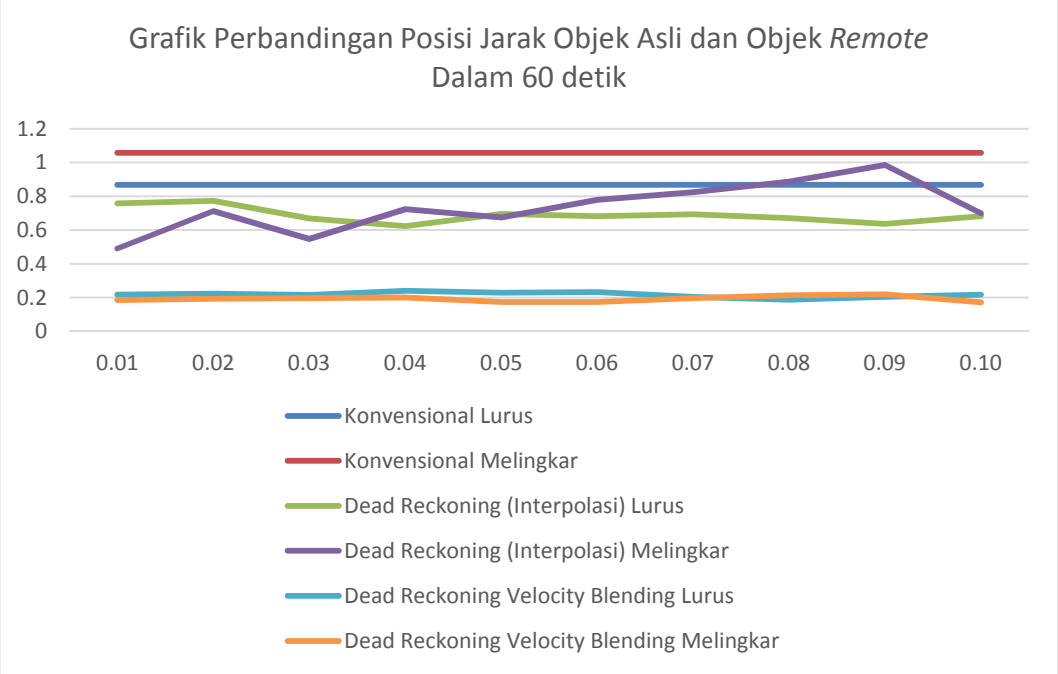
Pada hasil pengujian kinerja kesesuaian prediksi, rata-rata jarak yang terjadi berdasarkan *threshold* antara posisi objek asli dan posisi objek *remote* ditunjukkan pada Tabel 6.15.

Tabel 6.15 Rata-rata Jarak Objek Asli dan Objek *Remote* Berdasarkan *Threshold*

Threshold	Konvensional		*Dead Reckoning* (Interpolasi)		*Dead Reckoning* Variasi *Velocity Blending*	
	Lurus	Melingkar	Lurus	Melingkar	Lurus	Melingkar
0.01	0.86847	1.05679	0.75798	0.49106	0.21695	0.18488
0.02	0.86847	1.05679	0.77318	0.71301	0.22272	0.19341
0.03	0.86847	1.05679	0.66913	0.54674	0.21557	0.19718
0.04	0.86847	1.05679	0.62405	0.72391	0.24138	0.20029

0.05	0.86847	1.05679	0.69521	0.67394	0.23010	0.17399
0.06	0.86847	1.05679	0.68184	0.77804	0.23383	0.17399
0.07	0.86847	1.05679	0.69362	0.82401	0.20548	0.19765
0.08	0.86847	1.05679	0.67123	0.88712	0.18722	0.21319
0.09	0.86847	1.05679	0.63651	0.98504	0.20388	0.21976
0.1	0.86847	1.05679	0.68157	0.70006	0.21858	0.17227
Rata-rata	0.86847	1.05679	0.68843	0.73229	0.21757	0.19266

Hasil pengujian kinerja berdasarkan kesesuaian prediksi dapat diketahui bahwa dalam kurun waktu 60 detik rata-rata jarak posisi antara objek asli dan objek *remote* menggunakan metode konvensional adalah 0.86847 pada gerak lurus dan 1.05679 pada gerak melingkar, sedangkan menggunakan metode *Dead Reckoning* (interpolasi) adalah 0.68843 pada gerak lurus dan 0.73229 pada gerak melingkar, lalu menggunakan metode *Dead Reckoning* variasi *Velocity Blending* adalah 0.21757 pada gerak lurus dan 0.19266 pada gerak melingkar. Grafik perbandingan rata-rata jarak yang terbentuk ditunjukkan dalam Gambar 6.2.



Gambar 6.2 Grafik Perbandingan Jarak Rata-rata Berdasarkan *Threshold* Dalam 60 detik

Gambar Grafik 6.2 menunjukkan bahwa pada kurun waktu 60 detik rata-rata jarak yang terjadi dipengaruhi oleh tipe pergerakannya pada metode konvensional sedangkan cenderung tidak menentu pada metode *Dead Reckoning*. Sedangkan dari sisi rata-rata jarak yang terjadi menggunakan metode *Dead Reckoning* baik interpolasi maupun variasi *Velocity Blending* terlihat tidak menentu, jadi dapat disimpulkan bahwa rata-rata jarak pada metode *Dead Reckoning* baik interpolasi

maupun variasi *Velocity Blending* tidak dipengaruhi oleh *threshold*. Dari gambar grafik tersebut pula maka dapat disimpulkan dari segi jarak kesesuaian prediksi yang terjadi, menggunakan metode *Dead Reckoning* variasi *Velocity Blending* menunjukkan hasil yang paling optimal.

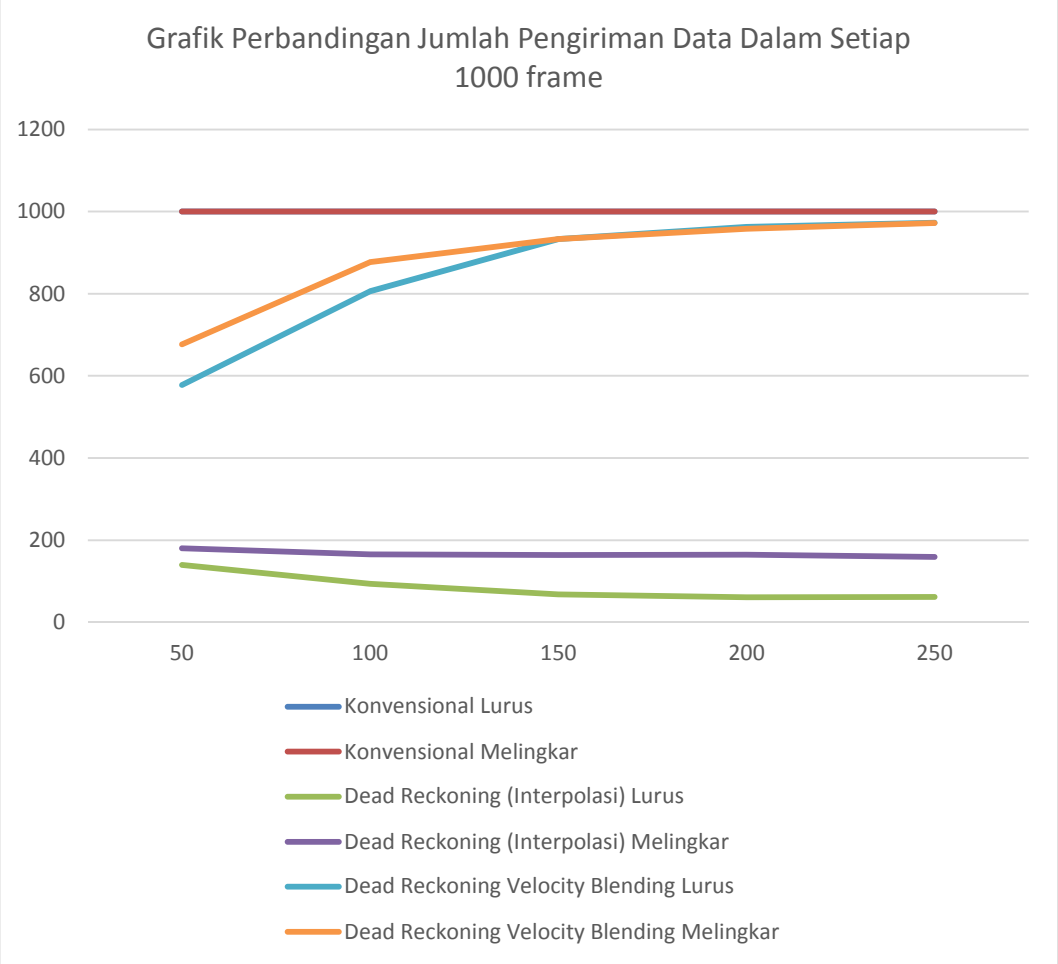
6.2.2 Analisis Kinerja Berdasarkan *Latency*

Analisis kinerja berdasarkan *latency* pada pengujian jumlah pengiriman data, rata-rata jumlah pengiriman data yang terjadi ditunjukkan pada Tabel 6.16.

Tabel 6.16 Rata-rata Jumlah Pengiriman Data Berdasarkan *Latency*

Rata-rata Latency (ms)	Konvensional		*Dead Reckoning* (Interpolasi)		*Dead Reckoning* Variasi *Velocity Blending*	
	Lurus	Melingkar	Lurus	Melingkar	Lurus	Melingkar
50	1000	1000	139.28	180.33	577.73	676.3
100	1000	1000	94.05	165.1	805.83	876.87
150	1000	1000	67.96	164.19	933.18	932.88
200	1000	1000	60.85	164.37	962.96	958.42
250	1000	1000	61.72	159.25	972.59	972.09
Rata-rata	**1000**	**1000**	**84.77**	**166.65**	**850.46**	**883.31**

Hasil pengujian kinerja berdasarkan jumlah pengiriman data dapat diketahui bahwa dalam kurun 1000 frame rata-rata pengiriman data yang terjadi dengan menggunakan metode konvensional adalah sejumlah 1000 kali baik pada gerak lurus maupun gerak melingkar, sedangkan dengan menggunakan metode *Dead Reckoning* (Interpolasi) adalah sejumlah 84.77 kali pada gerak lurus dan 166.65 kali pada gerak melingkar, lalu dengan menggunakan metode *Dead Reckoning* variasi *Velocity Blending* adalah sejumlah 850.46 kali pada gerak lurus dan 883.31 kali pada gerak melingkar. Grafik perbandingan jumlah pengiriman data yang terjadi ditunjukkan dalam Gambar 6.3.



Gambar 6.3 Grafik Perbandingan Jumlah Pengiriman Data Berdasarkan *Latency* dalam Setiap 1000 Frame

Gambar Grafik 6.3 menunjukkan bahwa pada kurun setiap 1000 frame tingkat jumlah pengiriman data pada metode *Dead Reckoning* baik interpolasi maupun variasi *Velocity Blending* dipengaruhi oleh *latency*, dimana pada metode *Dead Reckoning* (interpolasi) semakin besar *latency* maka semakin kecil jumlah pengiriman datanya, sedangkan metode *Dead Reckoning* variasi *Velocity Blending* cenderung meningkat. Pada gambar tersebut juga dapat diketahui bahwa metode *Dead Reckoning* (interpolasi) memberikan hasil yang paling optimal.

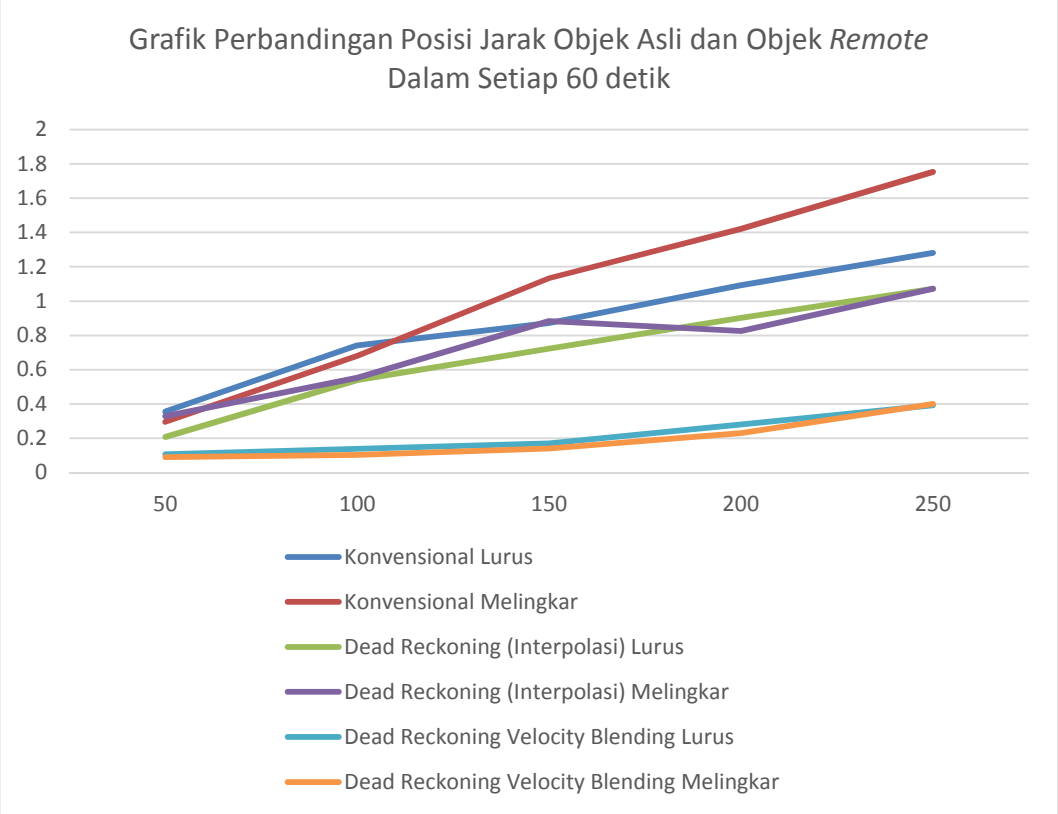
Pada hasil pengujian kinerja kesesuaian prediksi, rata-rata jarak yang terjadi berdasarkan *latency* antara posisi objek asli dan posisi objek *remote* ditunjukkan pada Tabel 6.17.

Tabel 6.17 Rata-rata Jarak Objek Asli dan Objek *Remote* Berdasarkan *Latency*

Rata-rata Latency (ms)	Konvensional		*Dead Reckoning* (Interpolasi)		*Dead Reckoning* Variasi *Velocity Blending*	
	Lurus	Melingkar	Lurus	Melingkar	Lurus	Melingkar

50	0.35414	0.29488	0.20744	0.32798	0.10585	0.09004
100	0.74144	0.68041	0.53892	0.55325	0.13900	0.10220
150	0.87299	1.13417	0.72257	0.88357	0.17078	0.14060
200	1.09190	1.42146	0.90134	0.82466	0.28041	0.23001
250	1.28189	1.75303	1.07189	1.07201	0.39182	0.40045
Rata-rata	**0.86847**	**1.05679**	**0.68843**	**0.73229**	**0.21757**	**0.19266**

Hasil pengujian kinerja berdasarkan kesesuaian prediksi dapat diketahui bahwa dalam kurun waktu 60 detik rata-rata jarak posisi antara objek asli dan objek *remote* menggunakan metode konvensional adalah 0.86847 pada gerak lurus dan 1.05679 pada gerak melingkar, sedangkan menggunakan metode *Dead Reckoning* (interpolasi) adalah 0.68843 pada gerak lurus dan 0.73229 pada gerak melingkar, lalu menggunakan metode *Dead Reckoning* variasi *Velocity Blending* adalah 0.21757 pada gerak lurus dan 0.19266 pada gerak melingkar. Grafik perbandingan rata-rata jarak yang terbentuk ditunjukkan dalam Gambar 6.4.



Gambar 6.4 Grafik Perbandingan Jarak Rata-rata Berdasarkan *Latency* Dalam 60 detik

Gambar Grafik 6.4 menunjukkan bahwa pada kurun waktu 60 detik rata-rata jarak yang terjadi dipengaruhi oleh *latency* dan tipe pergerakannya, dimana

semakin tinggi *latency* maka semakin besar pula jarak yang terbentuk, baik menggunakan metode konvensional maupun *Dead Reckoning* (interpolasi) dan *Dead Reckoning* variasi *Velocity Blending*. Dari grafik tersebut juga terlihat bahwa dengan menggunakan metode *Dead Reckoning* variasi *Velocity Blending* memberikan hasil yang paling optimal.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan perancangan, implementasi dan hasil pengujian dari penerapan metode *Dead Reckoning* pada *game multiplayer online*, maka dapat disimpulkan bahwa untuk menerapkan metode *Dead Reckoning* pada sebuah *game multiplayer online* dapat dilakukan dengan menerapkan algoritma baik *Dead Reckoning* (interpolasi) maupun variasinya sesuai kebutuhan pada objek yang ingin dilakukan sinkronisasi.

Penerapan metode *Dead Reckoning* terbukti lebih efektif daripada metode konvensional baik dari segi jumlah pengiriman data maupun kesesuaian posisinya. Penggunaan metode *Dead Reckoning* (Interpolasi) memiliki nilai penghematan bandwidth yang paling optimal, sedangkan dari segi kesesuaian posisinya menggunakan metode *Dead Reckoning* variasi *Velocity Blending* memiliki kesesuaian yang paling optimal. Nilai *threshold* dalam penerapan metode ini harus ditetapkan secara bijak, karena dapat mempengaruhi jumlah pengiriman data maupun kesesuaian jaraknya pada setiap *latency* yang terjadi.

7.2 Saran

Untuk meningkatkan kasil yang telah dicapai dari penelitian ini dapat dilakukan beberapa perbaikan sebagai berikut :

1. Diharapkan pada penelitian selanjutnya, penerapan metode *Dead Reckoning* dilakukan pada game dengan jenis yang berbeda dan dengan variasi *Dead Reckoning* yang berbeda
2. Melakukan variasi pada metode *Dead Reckoning* untuk menciptakan suatu algoritma baru yang lebih efisien dari segi jumlah pengiriman data dan kesuaian jarak prediksinya

DAFTAR PUSTAKA

[TGA-08] “Television Game Apparatus and Method”. United States Patents. Diambil pada 25/06/2008

[OED-08] Oxford English Dictionary. Oxford University Press. 2008. “Designed for or involving more than two (esp. Many) players or participants”

[BAH-08] Basori, Ahmad Hoirul., Sarno, Riyanarto., dan Surya, Widyanto. 2008. “The Development of 3D Multiplayer Mobile Racing Games Based on 3D Photo Satellite Map”

[HRC-10] Harvey, R. Cameron., Hamza, Ahmed., Ly, Cong., dan Hefeeda, Mohamed. 2010. “Energy-Efficient Gaming on Mobile Devices using Dead Reckoning-based Power Management”

[CRC-01] CRC Press LCC. 2001. "Internet-Based Games"

[ZNL-04] Zang, Li. Concordia University. 2004. "Multiplayer Network Game Programming in MFC"

[MHW-04] Mulholland, Andrew., dan Hakala, Teijo. Wordware Publishing, Inc. 2004. "Programming Multiplayer Games"

[WJS-09] Wang, Alf Inge., Jarret, Martin., dan Sorteberg, Eivind. 2009. "Experiences from Implementing a Mobile Multiplayer Real-time Game for Wireless Networks with High Latency"

[AJS-12] Agar, Jacob., Corriveau, Jean-Pierre., dan Shi, Wei. 2012. "Play Patterns for Path Prediction in Multiplayer Online Games"

[BHN-05] Banavar, Hemant N. Florida State University. 2005. "Accuracy and Fairness in Dead Reckoning Based Distributed Multiplayer Games"

[LYE-11] Lengyel, Eric. A K Peters, Ltd. 2011. "Game Engine Gems 2"

[HAC-01] Haag, Chris (2001). "Targeting - A variation of dead reckoning ". gamedev.net. Retrieved October 14, 2015.

LAMPIRAN A MANUAL PROGRAM

A.1 Minimum Requirement

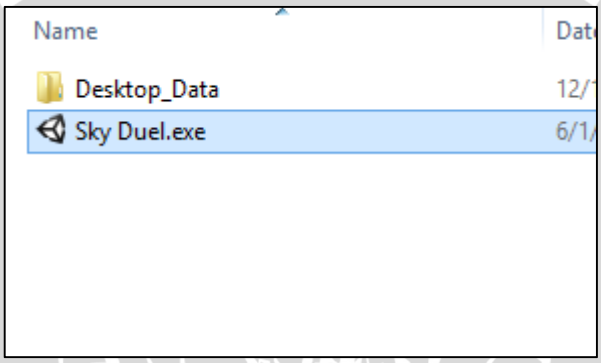
Minimum Requirement untuk menjalankan aplikasi adalah sebagai berikut :

- Intel P4
- 512MB RAM
- Windows 7 Personal Edition

A.2 Petunjuk Menjalankan Aplikasi

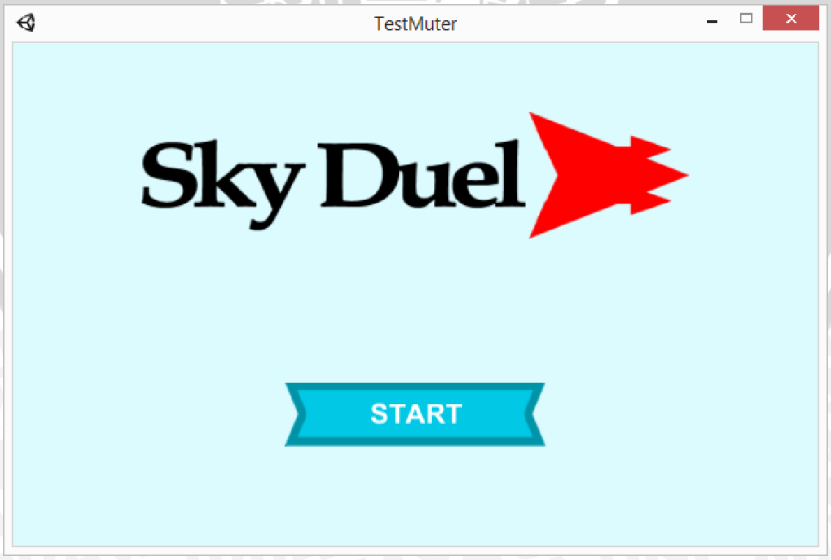
Langkah-langkah untuk menjalankan aplikasi adalah sebagai berikut :

a. Buka direktori tempat aplikasi berada seperti ditunjukkan pada Gambar L-A.1, lalu jalankan aplikasi.



Gambar L-A.1 Direktori Aplikasi

b. Pada awal *game* akan muncul tampilan menu seperti ditunjukkan pada Gambar L-A.2, lalu klik tombol *start* untuk memulai permainan.

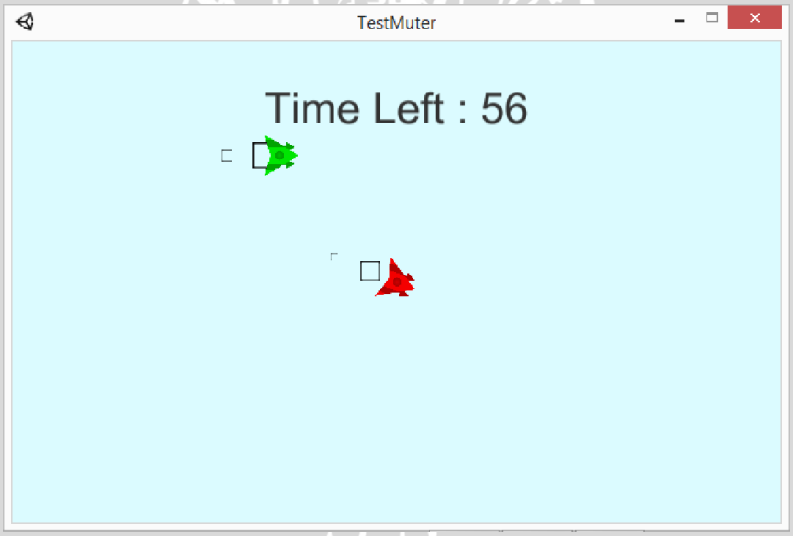


Gambar L-A.2 Tampilan Main Menu

c. Pemain menunggu terlebih dahulu hingga ada pemain lain yang ikut bermain, seperti yang ditunjukkan pada Gambar L-A.3.

Gambar L-A.3 Tampilan Menunggu Pemain Lain

d. Ketika pemain lain telah ikut bermain maka permainan dimulai seperti ditunjukkan pada Gambar L-A.4.



Gambar L-A.4 Tampilan Saat Permainan Berlangsung

e. Pemain dapat menggunakan tombol spasi untuk menembak, jika pemain berhasil menembak pemain lain maka pemain tersebut menang seperti yang ditunjukkan pada Gambar L-A.5, namun jika waktu habis maka permainan akan seri. Klik tombol *Main Menu* untuk kembali ke menu utama.

Gambar L-A.5 Pemain Menang

LAMPIRAN B PENGUJIAN KINERJA

B.1 Pengujian Kinerja Berdasarkan Jumlah Pengiriman Data

Pengujian kinerja berdasarkan jumlah pengiriman data pada metode *Dead Reckoning* dilakukan dengan menghitung rata-rata jumlah setiap pengiriman data yang terjadi dalam kurun 1000 *frame* dalam 10 kali pengujian pada *threshold* tertentu dengan rata-rata *latency* tertentu, sedangkan pada metode konvensional dilakukan sekali saja pada setiap rata-rata *latency*. Setiap parameter tersebut juga dilakukan pengujian dengan dua tipe pergerakan, yaitu gerak lurus dan gerak melingkar. Setiap kali pengiriman data terjadi, maka total pengiriman akan ditambahkan. Saat permainan sudah mencapai 1000 *frame* maka data tersebut akan dimunculkan pada *console* untuk direkap keseluruhan nantinya. Kode untuk mendapatkan nilai total tersebut ditunjukkan pada Tabel L-B.1.

Tabel L-B.1 Kode Pengujian Kinerja Berdasarkan Jumlah Pengiriman Data

```
int totalsent = 0;
int frame;

void TransmitData()
{
	totalsent++;
	... [kode lainnya] ...
}

void Update()
{
	... [kode lainnya] ...
	frame++;
	if(frame == 1000)
	{
		Debug.Log(totalsent);
	}
}
```

B.2 Pengujian Kinerja Berdasarkan Jarak Kesesuaian Prediksi

Pengujian kinerja berdasarkan jarak kesesuaian prediksi pada metode *Dead Reckoning* dilakukan dengan menghitung jarak antara posisi objek asli dan objek *remote* dalam kurun waktu 60 detik dengan *threshold* dan *latency* tertentu, sedangkan pada metode konvensional dilakukan sekali saja pada setiap rata-rata *latency.* Setiap *client* akan menyimpan posisi objek dan waktu penyimpanannya, hanya data-data yang memiliki waktu penyimpanan yang sama yang dibandingkan. Jarak antara objek asli dan objek *remote* dihitung menggunakan metode *Euclidean Distance* yang ditunjukkan pada Persamaan L-B.1.

$$d(p,q) = d(q,p) = \sqrt{(q_1 + p_1)^2 + (q_2 + p_2)^2 + \cdots + (q_n + p_n)^2} \quad \text{(L-B.1)}$$

Pada setiap *frame* yang terjadi dalam kurun waktu 60 detik, data posisi baik objek asli maupun objek *remote* disimpan kedalam sebuah *file* .csv sehingga nantinya

dapat dibaca dan dilakukan kalkulasi diantara keduanya. Kode untuk proses ini ditunjukkan pada Tabel L-B.2.

Tabel L-B.2 Kode Pengujian Kinerja Berdasarkan Jarak Kesesuaian Prediksi

```
void Start()
{
      ... [kode lainnya] ...
      var fileName = (isLocalPlayer ? "local" : "remote") +
".csv";
      if (System.IO.File.Exists(fileName))
      {
            Debug.Log("File Exist");
            System.IO.File.Delete(fileName);
      }
      sr = System.IO.File.CreateText(fileName);
}

void Update()
{
      ... [kode lainnya] ...
      sr.WriteLine(System.DateTime.Now.Ticks+";"+transform.posit
ion.x + ";" + transform.position.y + ";" + transform.position.z
+ ";");
}
```

Contoh data hasil pengujian jarak kesesuaian prediksi ini ditunjukkan pada Tabel L-B.3.

Tabel L-B.3 Contoh Data Hasil Pengujian Kinerja Berdasarkan Kesesuaian Prediksi

Ticks (nano seconds)	Posisi Objek Asli			Posisi Objek *Remote*			Jarak
	x	y	z	x	y	z	
635859918539932701	2.30	-7.10	0.00	0.54	-7.65	0.00	2.07630
635859918540430063	2.68	-6.95	0.00	0.54	-7.65	0.00	2.63760
635859918548898859	8.18	-3.09	0.00	7.24	-4.04	0.00	1.82866
635859918549539775	8.52	-2.70	0.00	7.24	-4.04	0.00	3.05676
635859918549709417	8.61	-2.60	0.00	7.24	-4.04	0.00	3.42304
635859918552741194	10.00	-0.62	0.00	9.58	-1.29	0.00	0.87636
635859918552912983	10.07	-0.51	0.00	9.58	-1.29	0.00	1.11044
635859918585288094	2.86	21.23	0.00	4.05	20.66	0.00	1.52212
635859918585458298	2.73	21.29	0.00	4.05	20.66	0.00	1.70603
635859918587647151	1.08	21.87	0.00	2.24	21.48	0.00	1.31558