BAB 2
TINJAUAN PUSTAKA

2.1 Game

Game atau dalam bahasa Indonesia artinya permainan video adalah suatu bentuk permainan yang dapat dimainkan dengan aturan tertentu yang mana ada pihak yang menang dan ada pihak yang kalah. Game biasanya dilakukan dengan tidak serius atau dengan tujuan menghibur.

Menurut Agustinus Nilwan [5]. Game merupakan permainan komputer yang dibuat dengan teknik dan metode animasi. Jika ingin mendalami penggunaan animasi haruslah memahami pembuatan game. Atau jika ingin membuat game, maka haruslah memahami teknik dan metode animasi, sebab keduanya saling berkaitan.

2.2 Game shooter multiplayer

Game Shooter atau dalam Bahasa Indonesia adalah game tembak-tembakan. *game Shooter* adalah salah satu jenis game dimana pemain menggunakan senjata api seperti pistol senapan untuk menjatuhkan lawan dalam sebuah pertandingan untuk memperebutkan suatu hal dalam permainan [6]. Game shooter ini dibagi menjadi dua yaitu First Person Shooter (FPS) dan Third Person Shooter (TPS) Dimana pada game First Person Shooter melihat dunia melalui mata merka sedangkan pada game Third Person Shooter pemain melihat dunia dari belakang karakter atau seperti sudut pandang orang ketiga.

Game Multiplayer adalah game yang memungkinkan game dimainkan oleh pemain sebanyak dua orang atau lebih untuk bermain bersama secara simultan [6]. Dalam game *multiplayer* dibagi menjadi dua kategori diantaranya *local multiplayer* dan *online multiplayer*. *Local multiplayer* artinya pemain bermain di perangkat yang sama seperti di depan TV contohnya Mario Kart, sedangkan *online*

multiplayer artinya pemain bermain secara terpisah seperti melalui jaringan *Local Area Network* (LAN) dan internet.

Dari pengertian diatas, *game shooter multiplayer* dapat diartikan sebagai game tembak-tembakan yang dimainkan bersama orang lain melalui suatu jaringan tertentu.

2.3 Server

Server adalah komputer yang menyediakan layanan untuk komputer untuk komputer lain pada sebuah jaringan [7]. Layanan yang diberikan bisa berupa file sharing, database dan lain sebagainya. Dalam konteks *game multiplayer*, server merupakan komputer yang mengatur semua interaksi jaringan dalam game terkait mengelola gameplay, menyimpan data game dan menyediakan komunikasi.

2.4 Klien

Klien dalam konteks jaringan adalah sebuah perangkat komputer yang melakukan pengiriman data ke server untuk mengirim informasi serta mendapatkan data dari server untuk menerima informasi [7]. Dalam *game multiplayer* klien adalah user atau pemain yang bermain dalam game tersebut.

2.5 Latensi

Latensi adalah waktu yang dibutuhkan untuk mengirimkan paket data dari sisi pengirim ke penerima dan begitupun sebaliknya. Latensi adalah waktu yang dibutuhkan sebuah paket untuk melakukan perjalanan dari satu titik ke titik lain dalam jaringan. Dalam konteks game online, latensi mengacu pada waktu yang dibutuhkan data untuk melakukan perjalanan dari klien pemain ke server game dan kembali lagi [1][8] [5][6][7][8][9][13][14][15]. Latensi yang rendah berarti hanya ada sedikit penundaan pada proses transfer data (lebih cepat), sedangkan latensi tinggi berarti penundaannya lebih lama (lebih lambat).

Banyak hal yang dapat menyebabkan latensi menjadi tinggi, mulai dari arsitektur jaringan yang kurang bagus, lokasi server yang jauh, hingga mekanisme jaringan dalam game itu sendiri dapat menyebabkan tingginya latensi dalam sebuah game.

2.6 Lag

Lag dalam konteks game merujuk pada penundaan antara input pemain dan output sistem [9][7][18]. Lag biasanya disebabkan oleh berbagai faktor diantaranya karena keterbatasan perangkat keras, kualitas koneksi internet, atau optimasi perangkat game yang buruk. Lag sendiri dapat menyebabkan ketidaknyamanan dalam bermain game, dan dapat membuat game menjadi tidak menyenangkan atau bahkan tidak dapat dimainkan [11][13].

Dalam kasus lag yang diakibatkan oleh jaringan yang buruk dapat diminimalisir dengan peningkatan Quality Of Service (QoS), metode transmisi seperti Multiplexing, hingga pergantian protokol jaringan. Namun, perlu adanya kemampuan game itu sendiri untuk menghadapi lag akibat jaringan yang biasanya menggunakan teknik kompensasi lag.

2.7 C++

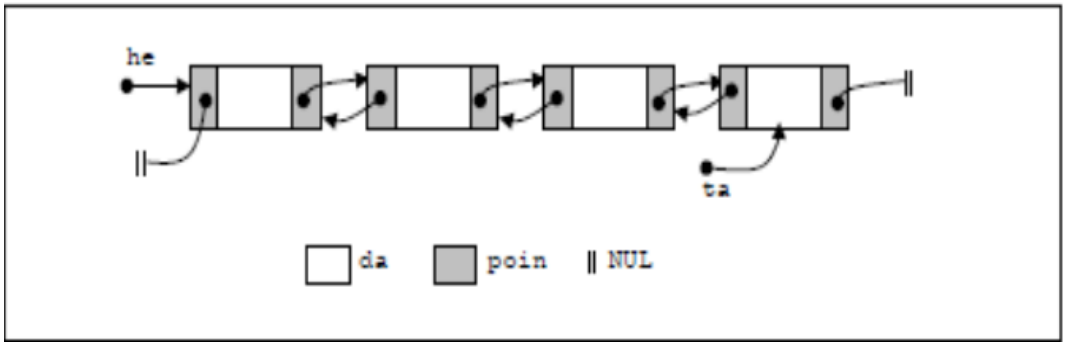
C++ merupakan bahasa pemrograman yang merupakan pengembangan dari bahasa pemrograman C. C++ disebut juga dengan bahasa *"C with Classes"*. Bahasa ini mendukung pemrograman berorientasi objek (OOP) dan dapat digunakan untuk membuat berbagai macam aplikasi. Bahasa C++ memiliki fungsi yang mirip dengan bahasa C, namun mendukung pemrograman berorientasi objek. C++ merupakan bahasa yang sangat populer yang sering digunakan dalam pengembangan perangkat lunak, termasuk pembuatan aplikasi desktop, game, dan sistem operasi [19].

Bjarne Stroustrup, seorang ilmuwan komputer Denmark, memelopori pengembangan bahasa C++ pada tahun 19791. Bahasa C++ berasal dari bahasa

Simula dan bahasa pemrograman lain seperti ALGOL 68, Ada, CLU dan ML1. Bahasa C++ mulai mendapatkan popularitas ketika Bjarne Stroustrup menambahkan fitur baru ke bahasa C, termasuk dukungan kelas untuk pemrograman berorientasi objek [20].

2.8 Double Linked List

Linked List atau juga disebut sebagai senarai berantai, adalah jenis struktur data yang berisi kumpulan data yang disusun secara linear, dengan setiap data disimpan dalam sebuah simpul dan dihubungkan melalui pointer antara simpul dan simpul lainnya [21]. Struktur data ini memiliki bentuk dasar, dengan sifat data dimasukkan ke dalam senarai melalui salah satu ujungnya. Sedangkan *double linked list* adalah *linked list* yang memiliki 2 pointer yang merujuk pada list sebelum dan list sesudahnya [21]. *Double linked list* memiliki pointer *head* (kepala) dan *tail* (ekor) untuk sebuah aksesnya. Contohnya dapat dilihat pada gambar berikut.



Gambar 2. 1 Ilustrasi Double Linked List [21]

Double linked list memiliki beberapa operasi dasar, diantaranya:

1. Penambahan: Menambahkan node baru ke dalam daftar.
2. Penghapusan: Menghapus node dari daftar.
3. Pencarian: Mencari node dengan nilai tertentu dalam daftar.
4. Traversal: Mengunjungi setiap node dalam daftar.

Double linked list memiliki beberapa keuntungan dibandingkan *linked list*, diantaranya:

1. Navigasi dua arah: Memungkinkan navigasi bolak-balik dalam daftar.
2. Memudahkan penyisipan dan penghapusan: Memudahkan penyisipan dan penghapusan node di tengah daftar tanpa perlu mencari node sebelumnya atau berikutnya.
3. Efisiensi memori: Memungkinkan penghapusan node tanpa perlu mencari node sebelumnya.

Namun double linked list sendiri memiliki beberapa kekurangan, diantaranya:

1. Lebih kompleks: Implementasi lebih kompleks dibandingkan *linked list* karena adanya dua pointer.
2. Membutuhkan lebih banyak memori: Membutuhkan lebih banyak memori karena adanya dua pointer.

2.9 Unreal Engine

Unreal Engine adalah sebuah aplikasi pengembangan permainan (game engine) yang dibuat oleh Epic Games. Pertama kali muncul pada tahun 1998 dengan game bertema *First Person Shooter.* Aplikasi ini awalnya dirancang untuk digunakan dalam berbagai jenis permainan, termasuk permainan bersembunyi (stealth), permainan pertempuran (fighting games), permainan peran daring multipemain masif (MMORPG), dan beberapa permainan peran (RPG). Unreal Engine ditulis dalam bahasa pemrograman C++ dan versi terbarunya, Unreal Engine 5 yang dirilis pada April 2022, mendukung berbagai platform desktop, ponsel, konsol, dan VR [22].

Dalam Unreal Engine ada 3 fungsi utama yang biasanya ada pada setiap kelas yang dibuat, diantaranya:

1. Kelas Konstruktor. Kelas Konstruktor adalah fungsi yang dipanggil ketika objek dibuat pertama kali. Pada fungsi ini dilakukan inisialisasi variabel dan, mengatur properti.
2. *Begin Play*. Fungsi *Begin Play* adalah fungsi yang dipanggil tepat setelah konstruktor dan objek telah sepenuhnya dibuat. Fungsi biasa ini digunakan

untuk membuat referensi ke objek lain dan menelurkan atau spawning suatu actor yang akan digunakannya misalnya senjata.

3. *Event Tick*. Fungsi *Event Tick* adalah sebuah fungsi yang dipanggil pada setiap frame untuk memperbaharui status sebuah objek. Fungsi ini biasanya digunakan untuk merespon input pemain, menggerakan objek, dan melakukan perhitungan yang dilakukan secara terus menerus.

Pada Unreal Engine juga memiliki konsep dalam interaksi pengiriman data dan teknis antara server dan klien dengan menggunakan replikasi variabel *dan Remote Procedure Calls* (RPC), yang mana akan dijelaskan lebih lanjut pada penjelasan dibawah ini.

2.9.1 Replikasi Variabel

Replikasi variabel dalam Unreal Engine adalah mekanisme yang memungkinkan variabel atau properti untuk dibagikan dan diperbarui secara otomatis di antara klien dan server dalam lingkungan permainan berbasis jaringan secara konsisten [23]. Tujuan replikasi variabel sendiri adalah untuk memastikan nilai variabel di semua klien dan server selalu konsisten. Misalnya, jika posisi karakter berubah di satu klien, maka informasi ini harus diteruskan ke semua klien lainnya.

Berikut adalah Langkah-langkah bagaimana cara kerja replikasi variabel di Unreal Engine:

1. Server Menandai Data. Server pertama-tama akan menandai data yang perlu direplikasi. Data ini dapat berupa aktor, komponen, variabel, atau data lainnya yang penting untuk menjaga konsistensi game di semua klien.

2. Server Mengemas Data. Setelah data ditandai, server akan mengemasnya menjadi pesan. Pesan ini berisi informasi tentang data yang direplikasi, seperti aktor yang direplikasi, komponen yang direplikasi, variabel yang direplikasi, dan nilai-nilainya.

3. Server Mengirim Pesan. Server kemudian akan mengirim pesan yang berisi data yang direplikasi ke semua klien di jaringan.

4. Klien Menerima Pesan. Klien kemudian akan menerima pesan dari server dan membongkar data yang terkandung di dalamnya.

5. Klien Menerapkan Data. Terakhir, klien akan menerapkan data yang direplikasi ke dunia game mereka. Ini melibatkan memperbarui aktor, komponen, dan variabel yang direplikasi agar sesuai dengan nilai yang diterima dari server.

2.9.2 Remove Procedure Calls

Remote Procedure Calls (RPC) dalam Unreal Engine adalah fungsi yang dipanggil secara lokal, tetapi dieksekusi secara remote di mesin lain (terpisah dari mesin yang memanggil) [23]. Fungsi RPC sangat berguna dan memungkinkan klien atau server untuk mengirim pesan satu sama lain melalui koneksi jaringan. Berikut adalah jenis-jenis fungsi RPC di Unreal Engine:

1. Server RPC. Dipanggil dari server dan dieksekusi di klien. Contoh penggunaan: mengirim pesan dari server ke klien untuk memainkan suara atau efek sementara.
2. Client RPC. Dipanggil dari klien dan dieksekusi di server. Contoh penggunaan: mengirim pesan dari klien ke server untuk mengubah status objek di dunia game.
3. Multicast RPC. Dipanggil dari server dan dieksekusi di semua klien yang terhubung. Berguna untuk mengirim pesan yang harus diterima oleh semua pemain.

2.9.3 Client-Server Programming

Sistem *Client-Server* dalam konteks jaringan adalah teknik untuk komunikasi antara server klien dan server yang mana dalam sistem ini salah satu pemain yang berperan sebagai *host*, maka ia berperan juga sebagai server dan komputer lainnya

yang memainkan permainan tersebut berperan sebagai klien [1][2]. Berbeda dengan sistem *Peer-to-Peer* (P2P) pada *game multiplayer* adalah model jaringan yang mana semua komputer yang berinteraksi secara langsung tanpa ada server pusat terpisah contohnya GTA Online.

2.9.4 Network Emulation Profile

Dalam Unreal Engine terdapat fitur yang menyajikan untuk melakukan kustomisasi latensi. Dalam fitur tersebut ada 2 profil yang bisa dipakai yaitu *average* dan *bad* dan juga bisa kustomisasi sendiri juga [24]. Berikut adalah kondisi jaringan pada profil *average* dan *bad.*

1. *Average*

   Incoming traffic : *Minimum Latency* : 30 ms
   Maximum Latency : 60 ms
   Packet Loss Percentage : 1%

   Outgoing traffic : *Minimum Latency* : 30 ms
   Maximum Latency : 60 ms
   Packet Loss Percentage : 1%

2. *Bad*

   Incoming traffic : *Minimum Latency* : 100 ms
   Maximum Latency : 200 ms
   Packet Loss Percentage : 5%

   Outgoing traffic : *Minimum Latency* : 100 ms
   Maximum Latency : 200 ms
   Packet Loss Percentage : 5%

Incoming traffic adalah penundaan jaringan atau kehilangan paket saat menerima paket data, sedangkan *outgoing traffic* adalah penundaan jaringan atau kehilangan paket saat mengirim data [24]. Maka latensi dapat disimulasikan dengan *incoming traffic* ditambah dengan *outgoing traffic*.

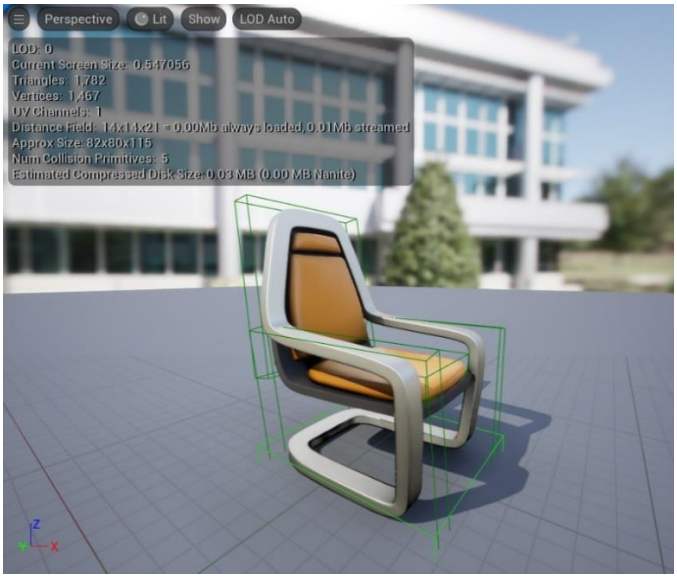
2.9.5 Collision Component

Collision dalam konteks game adalah objek semu atau sebuah objek yang tidak terlihat namun memiliki lokasi, rotasi dan ukuran yang jelas untuk proses interaksi antar benda dalam sebuah game [25]. Biasanya digunakan untuk menentukan interaksi suatu objek dengan objek lainnya.

Interaksi objek dengan collision biasanya dibagi menjadi tiga, diantaranya:

1. *Block*, jika collision memiliki interaksi *block* maka objek yang bertabrakan dengan collision tersebut tidak akan menembusnya, misalnya untuk kursi dan meja.
2. *Overlap,* jika collision memiliki interaksi *overlap* maka objek tersebut bisa menembus objek lainnya. Namun jika ada objek yang bertabrakan dengannya maka akan memanggil sebuah fungsi. Misalnya jika pemain melewati garis finish maka akan memanggil fungsi finish game misalnya.
3. *Ignore,* jika *collision* memiliki interaksi *Ignore* maka objek tersebut akan menembus objek lainnya dan tidak memanggil fungsi apa-apa. Misalnya jika karakter mempunyai skill untuk menembus objek manapun.

Collision Component adalah komponen *collision* yang menempel pada objek itu sendiri seperti kursi atau objek lainnya yang merepresentasikan jenis interaksi objek tersebut dengan objek lainnya, contohnya kursi akan bertabrakan dengan meja maka kursi tidak akan menembus meja. Contohnya seperti pada gambar 2.1.



Gambar 2. 2 Objek Kursi dengan Collision [26]

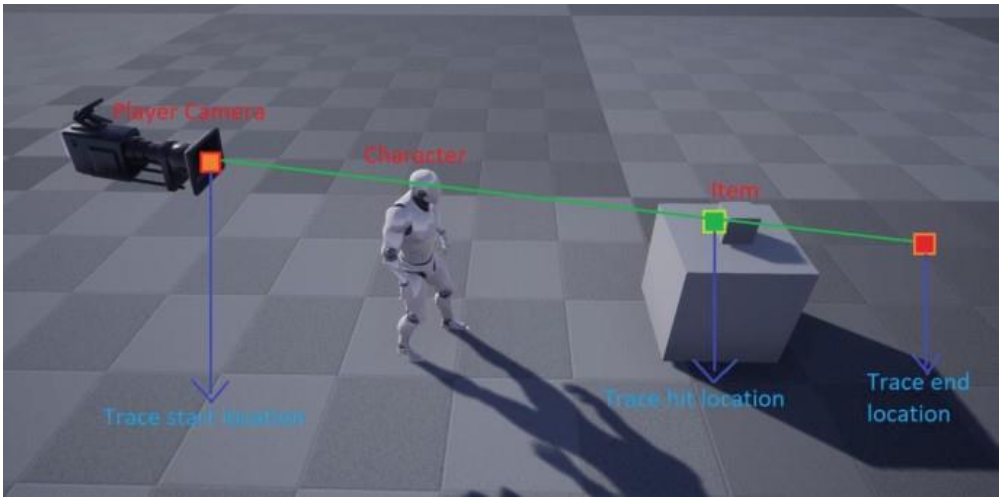
Objek berwarna hijau pada gambar tersebut merupakan *collision component* yang ada pada objek kursi tersebut. Sehingga interaksi antar objek pada kursi tersebut akan berdasarkan dari lapisan *collision* tersebut, yang artinya jika *collision* memiliki interaksi *block* maka objek kursi sebenarnya tidak akan menyentuh objek lainnya karena interaksi objek berdasarkan *collision component* yang ada padanya.

2.9.6 Raycasting

Raycasting adalah teknik yang memproyeksikan sinar dalam perspektif tiga dimensi ke dalam dunia dua dimensi (dalam kasus ini monitor), sinar dari mata diproyeksikan ke objek dalam dunia tiga dimensi yang selanjutnya sinar tersebut dilacak untuk mengidentifikasi titik potongnya dengan objek [27]. *Raycasting* pada dasarnya terdiri dari dua langkah utama. Sinar diproyeksikan dari mata ke objek dalam dunia tiga dimensi. Biasanya, dua titik menunjukkan sinar ini: titik awal dan titik akhir. Titik awal biasanya terletak di mata, dan titik akhir adalah di mana sinar akan berhenti. Pencarian titik potong, potongan sinar dengan objek yang dicari Untuk menentukan titik potong ini, biasanya digunakan persamaan matematika yang menjelaskan bagaimana sinar berinteraksi dengan objek.

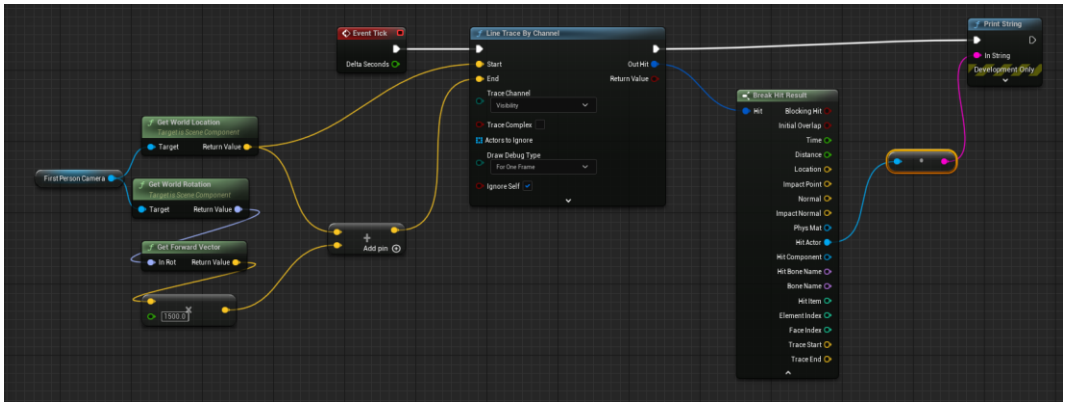
Raycasting dalam konteks *game shooter* adalah sebuah simulasi lintasan peluru yang ditembakan oleh pemain. Dengan proses *raycasting* sistem dapat mengetahui apakah lintasan tersebut mengenai objek atau tidak [2]. Dalam pendeteksian objek tersebut merupakan implementasi dari *hit detection* dan *collision detectio*n. *Hit detection* yaitu proses mendeteksi apakah dalam lintasan tersebut ada objek seperti bagian tubuh target yang berpotongan atau tida. *Collision detection* adalah proses yang mendeteksi apakah dalam lintasan tersebut ada *hit box* atau salah satu dari *collision*.

Dalam Unreal Engine proses raycast biasanya hanya dimasukan lokasi awal dan lokasi akhir dari raycast. Contohnya dalam *game third person sooter* proses raycast akan dimulai dari kamera yang lurus ke depan kamera, jika ditengah-tengah terdapat objek atau mengenai objek maka akan didapatkan lokasi objek tersebut seperti pada gambar 2.2.



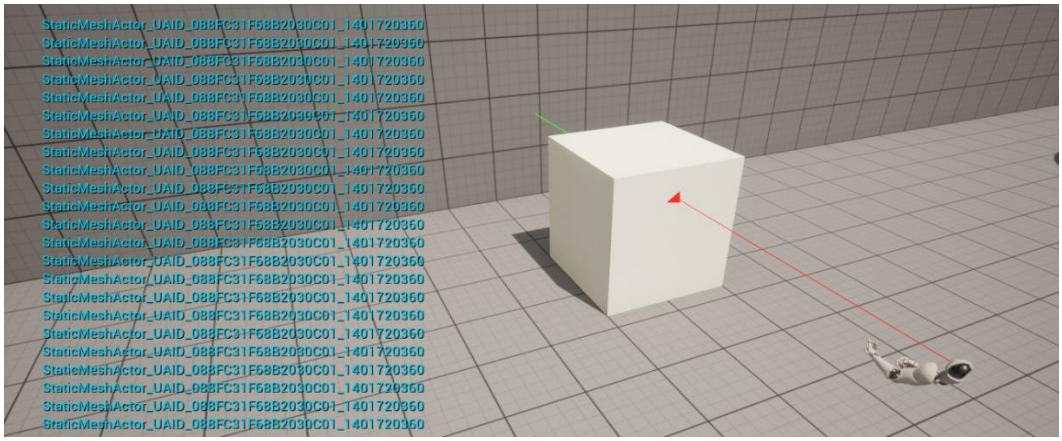
Gambar 2. 3 Simulasi *Raycasting* Dalam Unreal Engine [28]

Sebagai contoh, gambar berikut adalah contoh fungsi yang ada di *event tick* untuk melakukan proses *raycasting* di Unreal Engine menggunakan Blueprint.



Gambar 2. 4 Contoh Fungsi Raycasting Dalam Blueprint [29]

Gambar 2.3 memperlihatkan fungsi raycast di Unreal Engine dengan *line trace by channel*, dengan input dasar *start location* yang berasal dari posisi kamera ke end location yang berasal dari posisi depan kamera sejauh 15 m. Lalu outputnya sendiri merupakan sturct FHitResult dengan isi seperti pada gambar dan hanya diambil *hit actor* saja. Hasil dari proses raycast tersebut dapat dilihat pada gambar dibawah ini.



Gambar 2. 5 Hasil *Raycasting* Dari Fungsi Blueprint [29]

Gambar 2.4 memperlihatkan hasil dari fungsi sebelumnya. Dengan sinar warna merah menandakan proses raycasting berjalan, dan jika mengenai objek maka sinar akan berubah menjadi warna hijau dan sesuai dengan fungsi sebelumnya, objek yang dikenai akan di print sesuai dengan namanya.

2.10 Kompensasi Lag

Kompensasi lag adalah sebuah teknik untuk memberikan sebuah kompensasi terhadap pemain game *multiplayer* yang memiliki lag agar merasakan permainan yang setara [1][8][5][6][7][8][14][15] . Dalam *game shooter multiplayer* teknik kompensasi lag yang biasa digunakan adalah *Client-Side Prediction* dan *Server-Side Rewind* [1][8][5].

2.10.1 Client-Side Prediction

Dalam konteks game, *client-side prediction* adalah Teknik pemrograman jaringan yang digunakan untuk game multipemain untuk mengurangi dampak negatif dari koneksi latensi yang tinggi [1][14]. Teknik ini bertujuan agar input pemain menjadi lebih responsif sekaligus mengontrol tindakan pemain pada server jarak jauh.

Proses *client-side prediction* mengacu pada reaksi lokal klien terhadap masukan pengguna sebelum server mengkonfirmasi masukan dan memperbarui status permainan. Dengan demikian, klien tidak hanya mengirimkan input kontrol ke server dan menunggu pembaruan status permainan, tetapi juga memprediksi status permainan secara lokal dan memberikan umpan balik kepada pengguna tanpa harus menunggu pembaruan status permainan dari server. Prediksi sisi klien mengurangi masalah latensi karena tidak ada lagi penundaan waktu di sisi klien antara input dan umpan balik visual karena waktu ping jaringan. Namun, teknik ini juga mengakibatkan inkonsistensi antara status game di sisi klien dan sisi server, yang harus diatasi agar game tetap dapat dimainkan.

Singkatnya, *client-side prediction* adalah teknik untuk mengurangi masalah latensi karena tidak ada lagi penundaan waktu di sisi klien antara input dan umpan balik visual karena waktu ping jaringan. Namun, teknik ini juga mengakibatkan desinkronisasi antara status game di sisi klien dan sisi server, yang harus diatasi agar game tetap dapat dimainkan.

2.10.2 Server-Side Rewind

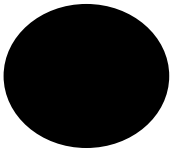
Dalam konteks game, *server-side rewind* adalah teknik yang digunakan dalam pengembangan game untuk mengatasi masalah latensi dan desinkronisasi dalam game bertipe multiplayer. Dengan teknik ini, memungkinkan server untuk "memutar ulang" status permainan sebelumnya dan menghitung ulang pergerakan dan tindakan pemain berdasarkan masukan dari klien. Pemutaran ulang sisi server memungkinkan server memperbaiki perbedaan apa pun antara apa yang terjadi di sisi klien dan apa yang sebenarnya terjadi di sisi server [2].

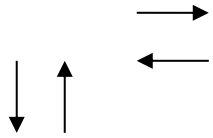
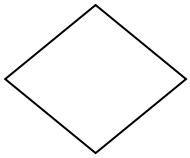
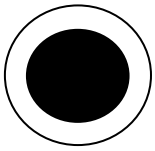
Singkatnya, *server-side rewind* adalah salah satu teknik yang digunakan dalam pengembangan game berbasis jaringan untuk meningkatkan pengalaman bermain dan memastikan bahwa semua pemain memiliki pengalaman yang serupa meskipun memiliki kualitas jaringan yang berbeda.

2.11 Flowchart

Flowchart merupakan diagram yang menggambarkan langkah-langkah dan keputusan sebuah proses dari sebuah sistem atau juga proses bisnis. *Flowchart* menggunakan simbol-simbol standar untuk mewakili berbagai jenis langkah, seperti keputusan, masukan, keluaran, dan proses [30]. Berikut adalah simbol *flowchart* yang digunakan dalam penelitian ini.

Tabel 2. 1 Simbol Flowchart

Simbol	Pengertian	Keterangan
	Status Awal	Menunjukan awal dari sebuah diagram aktivitas.
	Aktivitas	Menunjukan aktivitas yang dilakukan oleh sistem, yang biasanya diawali oleh kata kerja.

Simbol	Pengertian	Keterangan
	Input/Output	Menunjukan proses memasukan data dan sekaligus proses keluaran data.
	Garis aliran	Menunjukan arus sata antar simbol/proses.
	*Decision*	Menunjukan pilihan yang akan dikerjakan atau keputusan yang harus dibuat dalam proses pengolahan data.
	Penggabungan	Menunjukan penggabungan dimana lebih dari satu aktivitas digabung menjadi satu
	Status Akhir	Menunjukan status akhir yang dilakukan oleh sistem.

2.12 Interpolasi

Interpolasi adalah proses memperkirakan sebuah nilai dari suatu fungsi yang grafiknya melewati sekumpulan titik-titik. Titik-titik tersebut merupakan sebuah angka hasil dari sebuah eksperimen atau percobaan, atau bisa dari suatu fungsi yang diketahui [31].

Interpolasi biasanya digunakan untuk mencari sebuah nilai diantara titik – titik yang ada dikarenakan ada data yang tidak tersaji dengan lengkap atau hilang, dan bisa juga digunakan untuk memperkirakan nilai dari dari sebuah fungsi yang telah diketahui.

Berikut adalah rumus interpolasi linier untuk mencari dengan dua variabel (data dan waktu).

$$Y = y1 + (t - t_1) * \frac{y_2 - y_1}{t_2 - t_1} \quad (2.1)$$

Keterangan:

Y: Nilai yang ingin diperkirakan

y_1: Nilai data pada titik x1

y_2: Nilai data pada titik x2

t: Nilai waktu di mana ingin memperkirakan y

t_1: Nilai waktu pada titik pertama

t_2: Nilai waktu pada titik kedua

2.13 Hitscan

Hitscan adalah Sebuah teknik untuk mensimulasikan tembakan senjata melalui uji *raycasting*. Dalam kasus ini Hitscan ini adalah jenis peluru yang biasa digunakan suatu game yang tidak menggunakan konsep fisika dalam peluncuran pelurunya seperti pada game Counter-Strike [2][13].

2.13 Registrasi Hit

Registrasi Hit dalam konteks *game shooter multiplayer* mengacu pada proses di mana server game menentukan apakah tindakan pemain, seperti menembakkan senjata, menghasilkan keberhasilan mengenai target dalam dunia game. Saat pemain menembakkan senjata dalam game, game tersebut perlu mencatat secara akurat apakah tembakannya mengenai target, seperti pemain lain atau suatu objek [1][2].

2.14 Frame Rate

Frame rate atau tingkat *frame*, adalah ukuran seberapa cepat suatu perangkat digital menghasilkan gambar bergerak. Biasanya Frame rate diukur dalam satuan detik yang biasa disebut *Frame Per Second* (FPS) yang artinya berapa banyak gambar yang dimuat dalam satu detik [4].

Semakin tinggi *frame rate* maka semakin rendah delay antara 2 gambar, sehingga pergerakan pemain akan semakin mulus. Nilai frame rate dalam sebuah game ditentukan oleh banyak faktor, mulai dari kemampuan perangkat keras, jumlah perhitungan dalam game yang diperlukan, artinya semakin realistis dan semakin banyak objek yang dimuat dalam game maka akan semakin banyak komputasi yang diperlukan yang berimbas pada jumlah *frame rate* yang dimuat.

Dalam penelitian ini *frame rate factor* merujuk pada delay antar frame, yang mana dapat dihitung dengan rumus.

$$FRFactor = \frac{1}{\text{FPS}} \tag{2.2}$$

2.15 Round Trip Time

Round Trip Time (RTT) adalah waktu yang dibutuhkan oleh paket untuk dikirim dari pengirim, mencapai penerima, dan kemudian kembali ke pengirim [7]. Dalam jaringan komputer, RTT, yang juga disebut sebagai waktu ping, digunakan untuk mengevaluasi stabilitas koneksi jaringan.

2.16 Sinkronisasi Waktu

Dalam penelitian ini sinkronisasi waktu merujuk pada kondisi untuk dilakukan sinkronisasi waktu antara klien dan server. Hal ini dilakukan untuk perlu dilakukan dikarenakan waktu dan server selalu memiliki perbedaan, bahkan seiring dengan berjalannya waktu akibat latensi akan membuat waktu antara server dan setiap klien berbeda, sehingga perlu sinkronisasi waktu untuk mendapatkan waktu server yang akurat [8][9][32][33].

Dikarenakan setiap Game Engine memiliki arsitektur jaringan untuk pembuatannya, maka diperlukan sebuah metode khusus untuk melakukannya. Berikut adalah rumus yang digunakan oleh Josh Sutphin [33] dalam mensinkronkan waktu di Unreal Engine. Pertama klien perlu melakukan *request* ke server dengan memasukan data waktu kapan melakukan request, lalu server akan mengirim data

waktu perintah diterima di server dan waktu klien melakukan request kepada klien lagi itu sendiri, setelah itu klien tersebut akan mendapatkan data terkait kapan waktu *request* klien sebelumnya dan kapan waktu saat sampai ke server. Setelah mendapat itu waktu pada klien menerima sekarang akan dikurangi waktu saat klien menerima *request* sebelumnya dan didapatkan *Round Trip Time* (RTT), setelah RTT didapatkan maka nilai tersebut akan dibagi dua untuk mengetahui waktu yang dibutuhkan untuk mencapai server saja. Untuk mendapatkan waktu server sekarang ketika perintah diterima kembali di klien maka waktu di server menerima perintah akan ditambah dengan *½ RTT*. Lalu, didapat perbedaan juga waktu klien dan server dengan cara menghitung kapan waktu server saat itu dikurangi waktu klien menerima *request* . Perbedaan waktu klien dan sever ini diperlukan untuk mengetahui waktu server saat.

$$Round\ Trip\ Time = Waktu\ Klien\ Menerima\ Request - Waktu\ Klien\ Request \quad (2.3)$$

$$Waktu\ Server\ Sekarang = Waktu\ Server\ Menerima\ Request + \frac{Round\ Trip\ Time}{2} \quad (2.4)$$

$$Perbedaan\ Waktu\ Server\ dan\ Klien = Waktu\ Server\ Sekarang - Waktu\ Klien\ Menerima\ Request \quad (2.5)$$

Jika Suatu pemain ingin mendapatkan waktu server pada suatu waktu maka rumusnya menjadi

$$Waktu\ Server = Waktu\ Klien\ Request + Perbedan\ Waktu\ Server\ dan\ Klien \quad (2.6)$$

Frekuensi sinkronisasi yang ideal untuk game tergantung pada kebutuhan game itu sendiri [34]. Jika sinkronisasi dilakukan terlalu cepat seperti setiap 0,1 detik namun memiliki RTT 0,4 detik maka dikhawatirkan server bisa *overload* atau terjadi kehilangan paket [32].

2.17 Hit Time

Hit time dalam penelitian ini adalah sebuah variabel yang menggambarkan waktu hit atau waktu tembakan mengenai lawan dengan kondisi tembakan tersebut hasil prediksi klien [2]. *Hit time* dibuat ketika proses *Server-Side Rewind* dilakukan. Pada penelitian ini *Hit Time* didapat dari waktu server saat ini lalu dikurangi ½ RTT. Waktu server diperlukan dikarenakan proses registrasi hit dilakukan di server sehingga waktu server lah yang dipakai, lalu dikarenakan saat proses registrasi hit akan terjadi delay karena proses verifikasinya dilakukan di server, maka nilai waktu server harus dikurangi ½ RTT agar waktu hit adalah waktu saat tembakan itu dilakukan dan bukan ketika tembakan tersebut dilakukan di server [2]. Sehingga rumusnya menjadi seperti berikut.

$$Hit\ Time = Server\ Time - \frac{1}{2}RTT \tag{2.7}$$

2.18 Confusion Matrix

Confusion Matrix atau matriks kebingungan adalah sebuah tabel yang menggambarkan kinerja dari sebuah model klasifikasi [35]. Matriks ini menunjukkan jumlah prediksi yang benar dan juga salah yang dibuat oleh model untuk setiap kelas.

Berikut adalah contoh *confusion matrix* dengan model klasifikasi biner yang mengklasifikasikan contoh sebagai positif atau negatif.

Tabel 2. 2 Contoh *Confusion Matrix*

		Nilai Aktual		
		True	*False*	Total
Nilai Prediksi	*True*	TP (*True* Positif)	FP (*False* Positif)	
	False	FN (*False* Negatif)	TN (*True* Negatif)	
Total				

Berikut adalah penjelasan terkait TP, FN, FP, dan TN.

1. TP (*True* Positif) adalah model memprediksi positif dan aktualnya bernilai benar.
2. FN (*False* Negatif) adalah model memprediksi negatif dan aktualnya bernilai positif.
3. FP (*False* Positif) adalah model memprediksi positif dan aktualnya bernilai negatif.
4. TN (*True* Negatif) adalah model memprediksi negatif dan nilai aktualnya bernilai negatif.

Dari *confusion matrix* tersebut bisa digunakan untuk menghitung berbagai metrik evaluasi model klasifikasi contohnya presisi. Presisi adalah proporsi prediktif positif yang benar dari semua prediksi yang diklasifikasikan sebagai positif oleh model klasifikasi [35]. Mudahnya nilai presisi menggambarkan seberapa tinggi model membuat prediksi benar. Presisi dibuat dengan rumus.

$$Presisi = \frac{TP}{TP + FP} \tag{2.8}$$

2.19 Visual Studio

Visual Studio adalah *Integrated Development Environment* (IDE) yang berfungsi untuk memfasilitasi berbagai tool pemrograman seperti *code editor, interpreter, debugger* dan *compiler*. Aplikasi ini dibuat oleh Microsoft untuk menunjang berbagai kebutuhan pengembangan aplikasi dan salah satunya sebagai *code editor* untuk pengembagan game menggunakan Unreal Engine.

2.20 Studi Literatur

Studi literatur diperlukan untuk menganalisa penelitian sebelumnya untuk dijadikan pedoman penelitian ini yang kemudian akan menjadi acuan dan perbandingan dalam melakukan penelitian ini. Penelitian yang digunakan berasal dari jurnal, tesis, hingga hasil konferensi. Daftar penelitian tersebut antara lain.

1. Dalam Sebuah konferensi *Game Developer Conference* (GDC) tahun 2001. Yahn Benrier selaku pengembang game dalam studio Valve menjelaskan bagaimana studio Valve menggunakan metode kompensasi lag untuk game-game yang dibuat oleh mereka seperti Half Life, Team Fortress, dan Counter-Strike yang mana semuanya merupakan *game shooter mulitplayer*. Dalam konferensi tersebut dijelaskan bahwa dari sisi klien pengembang menggunakan metode *client-side prediction*. *Client-side prediction* adalah sebuah teknik agar sebuah perintah langsung dieksekusi tanpa otoritatif server, sehingga perintah langsung dieksekusi tanpa menunggu server terlebih dahulu sehingga tidak memiliki delay akibat latensi. Hal ini akan membuat game menjadi responsif meskipun memiliki latensi yang tinggi Namun hal ini akan membuat inkonsistensi antara status server dan klien. Oleh karena itu pengemang menggunakan metode *server-side rewind* untuk mengatasinya. *Server-side rewind* adalah teknik untuk mereplikasi status permainan atau menyamakan status permainan oleh klien kepada server dan semua klien akibat inkonsistensi seperti lokasi player akibat latensi yang tinggi dengan cara meminta klien meminta pose mundur untuk suatu target dan membandingkannya dengan pose lokal yang diamati pada klien. Hal ini akan membuat perintah yang dilakukan dengan *Client-Side Prediction* menjadi konsisten dengan server. Berdasarkan penelitian ini teknik *Client-Side Prediction* dan *Server-Side Rewind* dapat membantu pemain yang memiliki lag akibat latensi yang tinggi agar bermain game dengan lebih baik pada *game shooter multiplayer*.
2. Dalam sebuah penelitian yang berjudul A Time Rewind System for Multiplayer Games. Penelitian yang ditulis oleh H. Rahimi dan S. Ratti, A. A. N. Shirehjini pada tahun 2013. Penelitian ini membahas bagaimana menggunakan teknik time rewind untuk mengatasi lag akibat latensi pada game multiplayer 2D seperti game Prince Of Persia. Dalam penelitian tersebut menceritakan bagaimana sulitnya menggunakan *time rewind* untuk game multiplayer dikarenakan jika sebuah entitas yang terpengaruh atau sedang memakai *time rewind* berinteraksi satu sama lain, utamanya Ketika

entitas yang terpengaruh time rewind secara fisik bertabrakan dengan entitas yang tidak terpengaruh atau memakai time rewind dan keduanya menempati ruang fisik yang sama. Dari hasil penelitiannya didapat cara bagaimana cara mengatasinya. Pertama menggunakan *best-effort corrective algorithm*. Algoritma ini bekerja dengan cara membiarkan sebuah entitas tumpang tindih sejenak dan memperbaiki posisi mereka setelah mereka melakukannya. Kedua menggunakan *Time Delineation Mechanism*. Yaitu sebuah teknik yang bekerja dengan cara membagi dunia game menjadi beberapa wilayah, yang masing-masing memiliki batas penggambaran waktu (time delineation). Artinya jika player 1 menggunakan *time rewind* maka hanya di wilayah player 1 saja yang terpengaruh, sedangkan player 2 tidak sehingga dapat mencegah konflik dalam penggunaan *time rewind*. Ketiga menggunakan *Time Synchronization Mechanism* atau teknik sinkronisasi waktu bekerja dengan memastikan semua pemain mengalami timeline dan peristiwa yang sama, terlepas dari latensi jaringan atau faktur lain yang dapat menyebabkan perbedaan. Secara teknis teknik ini menggunakan protokol sinkronisasi waktu yang memastikan semua jam pemain disinkronkan dan semua kejadian diproses dalam urutan yang sama untuk semua pemain. Hasilnya terjadi peningkatan persentase pemain dalam mempertahankan *health player* (HP) dan lawan yang terbunuh, sehingga membuat permainan jauh lebih baik dan lebih adil meskipun memiliki latensi yang tinggi. Kesimpulannya dengan menggunakan metode, teknik dan algoritma yang dijelaskan sebelumnya dapat membuat permainan pemain yang lag akibat latensi menjadi lebih baik.

3. Dalam sebuah tesis yang berjudul Implementation and Evaluation of Hit Registration in Networked First Person Shooters. Ditulis oleh Jonathan Lundgren pada tahun 2021. Tesis ini menjelaskan bagaimana mengimplementasi dan mengevaluasi metode kompensasi lag pada keberhasilan registrasi hit. Game yang digunakan adalah *Multiplayer* dengan sudut pandang orang pertama atau *First Person Shooter* (FPS). Penelitian ini menjelaskan mengimplementasikan teknik *Client-Side Prediction* dan *Server-Side Rewind* untuk game multiplayer dengan Unreal Engine 4.

Berdasarkan pengujian yang dilakukannya dengan peluru jenis hitscan yang dilakukan sebanyak 300 kali tembakan lalu sudut dan jarak yang acak, dan juga latensi 0ms, 50ms, 100ms, 200ms, dan 400ms. Didapat bahwa dalam suatu waktu bisa mendapat hasil dibawah 50%, bahkan untuk pengujian dengan latensi 200ms bisa mendapat hasil di bawah 10%. Dalam penelitian tersebut dicoba ditambahkan *fudge factor* atau angka konstan pada proses *rewind*. Hasilnya terjadi peningkatan keberhasilan registrasi hit setelah ditambahkan *fudge factor*, namun menurutnya peningkatan tersebut tergantung pada *frame ratenya*. Berdasarkan kesimpulan dari penelitian tersebut didapat bahwa keberhasilan registrasi hit pada pemain yang lag meskipun sudah memakai teknik kompensasi lag hasilnya tidak terlalu baik, oleh karena itu perlu diperlukan perbaikan dari segi metode yang dipengaruhi suatu hal misalnya *frame rate*.

4. Dalam sebuah penelitian yang berjudul A Survey and Taxonomy of Latency Compensation Techniques for Network Computer Games. Ditulis oleh S. Liu, X. Xu, dan M. Claypool pada tahun 2022 menulis tentang bagaimana bentuk taxonomi kompensasi lag dalam *game multiplayer* hingga saat ini. Ada berbagai macam teknik taksonomi yang selama ini digunakan untuk memberikan kompensasi terhadap player yang memiliki latensi yang buruk diantaranya: Feedback : terdiri dari Latency Concealment dan Latency Exposure. Prediction Terdiri dari Self Prediction, Speculative Prediction, dan Other Prediction yang mana Other prediction diturunkan lagi menjadi Interpolasi dan Ekstrapolasi. Time Manipulation terdiri dari Time Wrap dan Time Delay. Time Delay terdiri dari Incoming Delay dan Outgoing Delay. World Adjustment terdiri dari Control Assistance dan Attribute Scaling. Kesimpulannya ada banyak sekali teknik yang biasa digunakan dalam metode kompensasi lag.
5. Dalam sebuah penelitian yang berjudul Client-Side Network Delay Compensation for Online Shooting Games. Ditulis oleh T. Motoo, J. Kawasaki, T. Fujihashi, S. Saruwatari, and T. Watanabe tahun 2021. Penelitian ini meneliti Bagaimana memberikan kompensasi delay dari sisi

klien untuk *game shooter multiplayer* dengan menggunakan metode regresi. Didapat dua metode yaitu prediksi *deep reinforcement learning* (DRL) tunggal dan *deep reinforcement learning* (DRL) beruntun. Hasilnya terjadi peningkatan akurasi prediksi dari metode sebelumnya dan untuk DRL beruntun hasilnya jauh lebih baik namun perlu komputasi yang lebih besar sedangkan DRL tunggal hasilnya tidak sebaik yang beruntun namun biaya komputasinya lebih sedikit. Kesimpulannya metode regresi menggunakan *deep reinforcement learning* (DRL) membuat prediksi dari sisi klien menjadi lebih akurat.

6. Dalam sebuah paper yang berjudul Timelines: simplifying the programming of lag compensation for the next generation of networked games. Ditulis oleh Cheryl Savery dan T. C. Nicholas Graham tahun 2013. Penelitian ini meneliti tentang bagaimana menyederhanakan pemrograman kompensasi lag untuk game jaringan generasi berikutnya dengan timeline programming. Hasilnya dengan menggunakan *timeline programming model* dapat menyederhanakan bagaimana metode kompensasi lag diimplementasikan untuk game multiplayer dikarenakan dengan dengan mengintegrasikan waktu sebagai aspek inti dari model pemrograman. Hal ini memungkinkan terjadinya manipulasi nilai-nilai masa lalu dan masa depan serta kendali atas perbedaan keadaan bagi para pemain yang berbeda. Kesimpulannya menggunakan *timeline programming model* dapat mempermudah dalam melakukan implementasi metode kompensasi lag untuk game multiplayer di masa mendatang.
7. Dalam sebuah penelitian yang berjudul Evaluation of Lag-Related Configurations in First-Person Shooter Games. Ditulis oleh Wai-Kiu Lee dan Rocky K. C. Chang tahun 2015. Dalam penelitian ini berfokus pada pengaruh Tick-Rate, Vertical Synchronization, dan kompensasi lag pada game *multiplayer first person shooter*. Hasilnya didapat bahwa Game dengan kompensasi lag akan meningkatkan akurasi rata-rata sebanyak 5% dibandingkan dengan yang tidak menggunakan kompensasi lag. Game dengan *tickrate* 128 Hz akan meningkatan akurasi rata-rata sebanyak

sebanyak 4% dibandingkan dengan game dengan tick rate 64 Hz dan game dengan *Vertical Synchronization* (V-Sync) akan meningkatkan akurasi rata-rata sebanyak 3% jika dibandingakn dengan game tanpa *V-Sync*.

8. Dalam sebuah penelitian yang berjudul On the Objective Evaluation of Real-Time Networked Games. Ditulis oleh Arnaud Kaiser, Dario Maggiorini, Nadjib Achir dan Khaled Boussetta tahun 2009. Penelitian ini berfokus pada bagaimana korelasi kualitas koneksi internet terhadap pengalaman pemain terhadap game First-Person-Shooter. Hasilnya jika penundaan jaringan lebih dari 60 milidetik dapat mengganggu pengguna game berjaringan real-time, khususnya game FPS (First-Person Shooter). Pemain dapat merasakan latensi serendah 75 milidetik dan merasa gameplay menjadi kurang menyenangkan pada latensi di atas 100 milidetik, namun penundaan tersebut mungkin tidak memengaruhi skor game secara signifikan. Disarankan untuk menjaga penundaan di bawah 150 milidetik untuk pertandingan yang dapat dimainkan. Lalu, Jitter tidak memainkan peran penting dalam mempengaruhi pengalaman bermain game dibandingkan dengan penundaan. Selain itu penting untuk melakukan rekayasa lalu lintas preemptive, mengevaluasi infrastruktur jaringan, dan mencocokkan pemain berdasarkan handicap jaringan untuk meningkatkan pengalaman bermain game di lingkungan jaringan. Kesimpulannya bahwa latensi dapat mengganggu permainan jika memiliki latensi diatas 60ms dan perlu adalanya perbaikan *Quality Of Servi*ce (QoS) untuk mengatasinya.
9. Dalam penelitian yang berjudul Online Games Traffic Multiplexing: Analysis and Effect in Access Networks. Ditulis oleh J. Saldana tahun 2012. Penelitian ini berfokus pada apa saja yang mempengaruhi lalu lintas jaringan pada pada game online dan bagaimana pengaruh teknik multiplexing untuk mengatasinya. Hasilnya lalu-lintas seperti bandwidth, ukuran paket, Quality Of Service, *traffic* dalam server server dan klien berpengaruh terhadap kualitas game online. Selain itu diperlukan penghematan bandwidth dengan menerapkan teknik multiplexing paket, kompresi header, dan pengelompokan paket, penelitian ini mencapai penghematan bandwidth yang signifikan dalam

game online. Penghematan berkisar antara 30% hingga 50% untuk lalu lintas klien-ke-server dari berbagai game. Kesimpulannya dengan menggunakan teknik multiplexing untuk penghematan bandwidth dapat membantu kualitas pengalaman bermain dalam bermain game online.

10. Dalam penelitian yang berjudul Effect of Network Quality on Player Departure Behavior in Online Games. Ditulis oleh Kuan-Ta Chen, P. Huang, and Chin-Laung Lei tahun 2009. Penelitian ini meneliti bagaimana hubungan antara kualitas jaringan dengan perilaku keluar pemain secara prematur dalam game online. Hasilnya latensi jaringan yang tinggi dan packet loss mempunyai pengaruh besar terhadap keputusan pemain untuk meninggalkan permainan sebelum waktunya atau sebelum permainannya selesai.
11. Dalam penelitian yang berjudul Is 60 FPS better than 30?: the impact of frame rate and latency on moving target selection. Ditulis oleh B. F. Janzen and R. J. Teather tahun 2014. Penelitian ini berfokus pada. bagaimana pengaruh *frame rate* dan latensi dalam menembak target yang bergerak pada game shooter multiplayer. Hasilnya Performa hit rates pemain dengan dengan frame rate 60 FPS 14% lebih tinggi dari 30 FPS, namun selisihnya antara 45 dan 60 FPS tidak signifikan. Latensi saja memiliki dampak yang lebih rendah daripada perbedaan frame rate yang sesuai dengan syarat dibawah 100ms. Ketika kedua faktor tersebut mempengaruhi kinerja, frame rate memiliki efek yang lebih besar daripada latensi yang ditimbulkannya. Kesimpulannya bahwa latensi dan dan *frame rate* mempengaruhi dalam menembak target yang bergerak dalam sebuah game.

Tabel 2. 3 Studi Literatur

No	Judul dan penulis	Tahun dan tempat	Objek penelitian	Perbandingan yang dijadikan alasan tinjauan penelitian
1	Latency Compensating Methods in	2001, Amerik	Metode kompensasi lag pada	Hasil dari penelitian ini menjadi acuan utama dalam metode

No	Judul dan penulis	Tahun dan tempat	Objek penelitian	Perbandingan yang dijadikan alasan tinjauan penelitian
	Client/Server In-game Protocol Design and Optimization Penulis: Yahn Benrier	a Serikat	game yang dibuat oleh studio Valve.	kompensasi lag hingga saat ini dan digunakan pada penelitian ini.
2	A Time Rewind System for Multiplayer Games Penulis: H. Rahimi, S. Ratti, A. A. N. Shirehjini	2013, Kanada	Time Rewind untuk game multiplayer.	Hasil penelitian ini digunakan sebagai referensi untuk implementasi time *rewind* untuk *game multiplayer* lainnya.
3	Implementation and Evaluation of Hit Registration in Networked First Person Shooters Penulis: Jonathan Lundgren	2021, Swedia	Keberhasilan registrasi hit pada game shooter multiplayer menggunakan metode kompensasi lag saat ini.	Penelitian ini digunakan sebagai patokan utama peneliti dalam menggunakan metode kompensasi lag terbaru dan mendapatkan masalah dari penelitian tersebut.
4	A Survey and Taxonomy of Latency Compensation	2022, Amerika Serikat	Taksonomi kompensasi lag akibat latensi untuk	Hasil penelitian ini digunakan peneliti sebagai referensi teknik apa saja yang digunakan

No	Judul dan penulis	Tahun dan tempat	Objek penelitian	Perbandingan yang dijadikan alasan tinjauan penelitian
	Techniques for Network Computer Games Penulis: S. Liu, X. Xu, dan M. Claypool		*game multiplayer*	dalam metode kompensasi lag hingga saat ini sehingga menjadi pertimbangan penggunaan teknik apa yang cocok dalam *game shooter multiplayer*
5	Client-Side Network Delay Compensation for Online Shooting Games Penulis: T. Motoo, J. Kawasaki, T. Fujihashi, S. Saruwatari, and T. Watanabe	2021, Jepang	Teknik *Client-Side Prediction*	Hasil penelitian ini menjadi referensi peneliti untuk melihat apakah ada metode yang lebih baik dalam menggunakan teknik *client-side prediction*.
6	Timelines: simplifying the programming of lag compensation for the next generation of networked games	2013, Kanada	*Timelines Programming*	Hasil penelitian ini dijadikan referensi untuk penggunaan *timelines* dalam *cache pose*.

No	Judul dan penulis	Tahun dan tempat	Objek penelitian	Perbandingan yang dijadikan alasan tinjauan penelitian
	Penulis: Cheryl Savery dan T. C. Nicholas Graham			
7	Evaluation of Lag-Related Configurations in First-Person Shooter Games Penulis Wai-Kiu Lee dan Rocky K. C. Chang	2015, Hongkong	Tick-Rate, Vertical Synchronization, dan kompensasi lag.	Hasil penelitian ini dijadikan patokan untuk setting Tick-Rate pada game yang akan diteliti.
8	On the Objective Evaluation of Real-Time Networked Games Penulis: Arnaud Kaiser, Dario Maggiorini, Nadjib Achir dan Khaled Boussetta	2009, Prancis	Lag.	Hasil penelitian ini dijadikan referensi oleh peneliti untuk setting latensi saat pengujian.
9	Online Games Traffic Multiplexing:	2012, Spanyol	Multiplexing dan	Hasil penelitian ini dijadikan referensi oleh

No	Judul dan penulis	Tahun dan tempat	Objek penelitian	Perbandingan yang dijadikan alasan tinjauan penelitian
	Analysis and Effect in Access Networks Penulis: J. Saldana		penghematan *bandwidth.*	peneliti untuk menghemat bandwith.
10	Effect of Network Quality on Player Departure Behavior in Online Games Penulis: Kuan-Ta Chen, P. Huang, and Chin-Laung Lei	2009, Taiwan	Efek dari kualitas jaringan.	Hasil penelitian ini dijadikan referensi pentingnya latensi dan pengaturan latensi yang baik dalam *game multiplayer.*
11	Is 60 FPS better than 30?: the impact of frame rate and latency on moving target selection Penulis: B. F. Janzen and R. J. Teather	2014, Kanada	*Frame rate* dan latensi.	Hasil penelitian ini dijadikan referensi oleh peneliti bagaimana jumlah FPS minimal yang baik dalam *game shooter multiplayer.*