Rancang Bangun Sistem Deteksi Pelanggaran Lalu Lintas dalam Simulator Mengemudi menggunakan Unity Game Engine

Mohammad Iqbal[1], Karmilasari[2], Yuli Karyanti[3] dan Yulia Chalri[4]
Fakultas Ilmu Komputer & Teknologi Informasi, Universitas Gunadarma[1,2, 3, 4]
Jl. Margonda Raya 100 - Pondok Cina - Depok, 021-78881112
E-mail : mohiqbal@staff.gunadarma.ac.id[1], karmila@staff.gunadarma.ac.id[2]
yuli@staff.gunadarma.ac.id[2], liapsa@staff.gunadarma.ac.id[3]

Abstract **-- Faktor penyebab kecelakaan lalu lintas secara umum, terdiri dari beberapa hal, yaitu kondisi sarana transportasi, faktor kelalaian manusia, dan alam, serta belum optimalnya penegakan hukum lalu lintas. Namun demikian, hal yang menjadi penyebab utama tingginya angka kecelakaan lalu lintas adalah kelalaian manusia, sehingga diperlukan kesadaran berlalu lintas yang baik bagi masyarakat terutama untuk pengendara kendaraan bermotor kalangan usia produktif. Fakta bahwa pengemudi kendaraan yang memiliki sedikit pengalaman mengemudi lebih berisiko mengalami kecelakaan dibandingkan pengemudi yang telah berpengalaman. Pengalaman tersebut sangat mempengaruhi keterampilan kontrol, keterampilan pemrosesan, kalibrasi diri, persepsi risiko, dan motivasi atau sikap yang berhubungan dengan keselamatan. Dalam kegiatan ini, Sistem Simulator mengemudi menjadi sistem penting yang dapat digunakan meningkatkan keahlian. Tuntutan membuat situasi yang mirip dengan keadaan sesungguhnya di jalan raya menjadi tantangan bagi para pengembang untuk membuat desain simulator yang baik. Dalam penelitian ini dihasilkan rancang bangun sistem deteksi pelanggaran lalu lintas dalam simulator mengemudi dengan menggunakan alat bantu Unity Game Engine, yang telah diuji secara sistem black box.**

Kata Kunci**: Simulator mengemudi, platform perangkat lunak, Sistem Deteksi Pelanggaran Lalu lintas, 3D Model**

I. PENDAHULUAN

Dalam beberapa dekade, simulator mengemudi adalah suatu alat umum digunakan untuk membantu para pengemudi pemula agar mereka dapat belajar mengendarai kendaraan di suasana yang mirip dengan jalan raya. simulator dapat memberikan pelatihan yang tidak bisa didapatkan dengan praktik mengemudi secara langsung. Metode latihan mengemudi tradisional umumnya mengajarkan instruksi mengenai lingkungan mengemudi seperti peraturan di jalan, alat pemandu lalu lintas, dan bahaya-bahaya saat mengemudi dan langsung dipraktekkan ke jalan raya [2]. Namun, metode tradisional ini tidak dapat diterapkan dan memposisikan pengemudi pemula dalam keadaan bahaya di dunia nyata dengan alasan keselamatan berkendara. Simulator mengemudi ini dapat membantu dalam hal tersebut, sehingga pengemudi pemula dapat diberikan pengajaran dan pengalaman untuk menangani skenario mengemudi di jalan raya yang kompleks dan berbahaya untuk dapat melatih kemampuan mereka terkait dengan kesadaran akan situasi yang cepat dan kritis, melatih insting penilaian risiko dengan cepat, dan serta melatih pengambilan keputusan dalam situasi kritis [4].

Tujuan dari penelitian ini adalah membuat sebuah simulator mengemudi yang memiliki sistem deteksi pelanggaran lalu lintas yang terintegrasi dengan sistem penilaian kualitas mengemudinya berdasarkan banyak atau tidaknya pelanggaran yang dilakukan saat mengemudi. Simulator ini kembangkan selain untuk dapat membantu para pengemudi pemula untuk melatih kemampuan mengemudinya, juga untuk melakukan berbagai macam tes psikologi kognitif pada supir yang berkaitan dengan keselamatan berkendara.

Secara umum, perangkat keras *driving simulator* terdiri dari tampilan, sistem gerak, sistem audio dan untuk meningkatkan kesan biasanya dibangunkan kabin buatan dari suatu kendaraan. Aktivitas mengemudi adalah tugas visual karena pengemudi menerima sebagian besar informasi melalui matanya, sehingga konfigurasi tampilan terbaik paling berpengaruh untuk persepsi lingkungan yang akurat [2]. Pada penelitian ini mengembangkan perangkat lunak simulator mengemudi dan mengevaluasinya menggunakan sistem perangkat keras *driving simulator* yang telah dibuat oleh Iqbal dkk tahun 2020 [3].

Simulasi mengemudi yang dikembangkan ini, akan memfasilitasi pengguna untuk berkendara pada lingkungan yang telah disediakan. Terdapat dua mode yaitu *Mission Mode* dan *Free Mode.* Pada *Mission Mode,* pengguna akan menjelajahi lingkungan 3D yang telah dibuat dengan misi tertentu. Pengguna simulator harus menyelesaikan misi tersebut sesuai dengan ketentuan yang disebutkan pada misi. Sedangkan pada *Free Mode,* pengguna dapat mengendarai kendaraan virtualnya menjelajah lingkungan simulasi secara bebas. Beberapa pilihan keadaan

lingkungan simulasi pun disediakan yaitu dapat memilih satu dari enam tingkat kepadatan lalu lintas. Setelah itu pengguna simulator dapat melakukan proses simulasi berkendara dengan memanfaatkan dua macam *controller input* yaitu *keyboard* dan *mouse* serta *driving wheel* dan *pedals*.

Ketika simulasi sedang berlangsung, sebuah objek bernama *GameManager* akan menghitung skor berkendara pemain. Skor akan terus bertambah apabila pemain tidak melakukan pelanggaran lalu lintas. Di samping itu juga terdapat *combo multiplier* yang akan memperbesar pertambahan skor seiring dengan berjalannya waktu. Namun, saat pemain melakukan pelanggaran, *combo multiplier* akan dikembalikan menjadi satu dan program akan mulai mengurangi skor pemain. Selain itu, pelanggaran lalu lintas yang dilakukan pemain selama simulasi juga akan disimpan oleh GameManager dalam sebuah *file* log. Setelah pemain berhasil menyelesaikan misi yang diberikan pada *mission mode* atau menghentikan simulasi pada *free mode*, *GameManager* akan menyimpan skor akhir dalam *file* log dan pemain akan dibawa kembali ke menu utama. Pemain dapat melihat pelanggaran apa saja yang dilakukan selama simulasi dengan membuka menu laporan yang ada pada menu utama dan memilih *file* yang memiliki waktu yang sama saat pemain memulai simulasi.

II. METODOLOGI PENELITIAN

Penelitian ini adalah perekayasaan perangkat lunak simulator mengemudi yang akan digunakan lebih lanjut untuk melakukan berbagai pengukuran yang berkaitan dengan keselamatan berkendara secara virtual. Maka agar simulator yang dibuat dapat mendeskripsikan dengan baik lingkungan jalan raya maka perlu disusun metodologi yang digunakan dalam penelitian ini sebagai berikut :

1) Perencanaan Penelitian. Pada tahap ini perencanaan dimulai dengan menentukan tujuan penelitian lalu melakukan pencarian referensi yang berkaitan dengan topik penelitian dilakukan. Referensi yang dicari merupakan referensi yang berhubungan dengan penelitian driving simulator baik penelitian di Indonesia mapun di luar Indonesia. Sumber literatur diperoleh dari penelitian-penelitian sebelumnya baik yang berupa jurnal, prosiding, ebook, technical report, maupun situs web lainnya dengan sumber informasi yang valid. Informasi yang valid dan relevan merupakan hal yang paling utama dalam membuat kajian penelitian. Pada tahap ini juga dilakukan pemilihan perangkat lunak bantu yang digunakan, menentukan fitur-fitur dari simulator yang akan dibuat, dan menentukan batasan masalah yang dibahas. Selanjutnya adalah membuat sketsa lingkungan mengemudi dalam simulator beserta konsep peletakan rambu-rambu, marka jalan, dan lampu lalu lintas.
2) Perancangan dan Pembuatan / Penyediaan Aset-Aset untuk Simulator Mengemudi. Pada tahap ini, aset-aset yang dibutuhkan seperti rambu-rambu lalu lintas dirancang dan dibuat. Beberapa aset yang ada seperti bangunan dipersiapkan dengan menggunakan aset yang sudah tersedia pada Unity *Asset Store*.
3) Pembuatan Lingkungan Simulator Mengemudi. Pada tahap ini, setelah semua aset-aset tersedia, lingkungan dari simulator mengemudi dibuat sesuai sketsa yang dibuat saat perencanaan.
4) Pembuatan Program Simulator Mengemudi. Setelah lingkungan simulator telah selesai dibuat, sistem-sistem (seperti sistem lalu-lintas, sistem misi, sistem deteksi pelanggaran lalu lintas, dan lain-lain) untuk simulator mengemudi akan dibuat.
5) Uji Coba. Pada tahap ini simulator mengemudi yang telah dirancang akan diuji fitur-fiturnya. Perbaikan dan penyesuaian akan dilakukan apabila hasil uji coba yang telah dilakukan belum sesuai dengan tujuan yang ingin dicapai.

III. HASIL DAN PEMBAHASAN

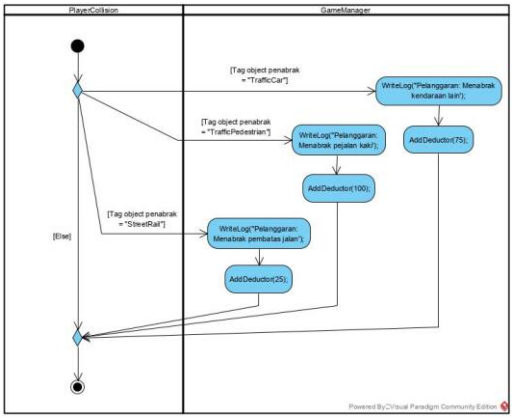
Aplikasi simulator mengemudi ini perancangannya dilakukan dengan menggunakan Unity sebagai *game engine* dan C# sebagai bahasa pemrogramannya. Tahap ini terdiri dari beberapa langkah yaitu Perancangan Lingkungan Simulasi, Persiapan/Perancangan Model 3D Mobil dan Pejalan Kaki, Perancangan Sistem Deteksi Pelanggaran Lalu Lintas, Perancangan sistem penilaian simulasi, perancangan sistem Lalu Lintas, Perancangan Mode Simulasi, dan Perancangan Sistem Kontrol Mobil. Pada artikel ini dibahas dua langkah perancangan yaitu Sistem Deteksi Pelanggaran Lalu Lintas dan Sistem Penilaian Simulasi.

III.1. Perancangan Sistem Pendeteksi Pelanggaran Lalu lintas

Dalam mendeteksi pelanggaran, beberapa fitur pada Unity digunakan. Fitur-fitur tersebut adalah *Prefab*, *Collider*, dan *Trigger*. *Prefab* [5] [6] [7] digunakan untuk membuat objek-objek pada lingkungan simulasi terhubung pada satu cetak biru, sehingga apabila *Prefab* (cetak biru) diubah, seluruh objek akan berubah juga. *Trigger* digunakan untuk membuat sebuah pembangkit kejadian. *Trigger* ini akan membangkitkan suatu kejadian saat pemain melakukan pelanggaran berkendara. Kejadian yang dibangkitkan berupa perekaman pelanggaran dan pesan bagi pemain. Sistem deteksi pelanggaran lalu lintas yang dibuat ada 4 macam yaitu:

1) Saat pemain menabrak mobil, pejalan kaki, dan pembatas jalan.

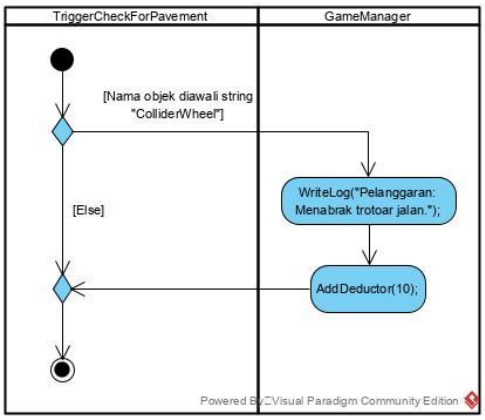
2) Saat pemain keluar dari area jalan (masuk ke area trotoar).
3) Saat pemain melanggar lampu lalu lintas di persimpangan.
4) Saat pemain berkendara melawan arus lalu lintas.



Gambar 1. Diagram aktivitas PlayerCollision

1. Deteksi Saat pemain menabrak mobil, pejalan kaki, dan pembatas jalan.

Untuk sistem deteksi pelanggaran lalu lintas ini memanfaatkan fungsi *Collider*. *Collider* akan terpasang pada setiap objek yang ada dalam simulasi. Pada *collider* untuk objek mobil, pejalan kaki, dan pembatas jalan, *script* PlayerCollision ditambahkan. *Script* PlayerCollision memiliki metode OnCollisionEnter yang merupakan salah satu metode *built-in* untuk kelas yang diturunkan dari kelas MonoBehaviour. Metode ini akan dipanggil saat *collider* objek tersebut menabrak *collider* objek lain. Saat metode OnCollisionEnter dipanggil, *tag* dari objek yang ditabrak akan diperiksa. Jika *tag* objek yang ditabrak adalah "TrafficCar" maka metode WriteLog dan AddDeductor dari GameManager akan dipanggil (Ilustrasi pada gambar 1). Metode WriteLog berfungsi untuk merekam tabrakan tersebut ke dalam *file* log. Pada program, teks yang bertuliskan "Pelanggaran: Menabrak kendaraan lain." akan disimpan dalam *file* log. Metode AddDeductor berfungsi untuk menambahkan nilai pengurang pada variabel deductor di GameManager yang berfungsi untuk menyimpan nilai pengurang skor. Nilai sebesar 100 akan ditambahkan bila objek yang ditabrak memiliki *tag* "TrafficCar". Selanjutnya, jika objek yang ditabrak memiliki *tag* "TrafficPedestrian", teks "Pelanggaran: Menabrak pejalan kaki." akan disimpan pada *file* log oleh metode WriteLog dan nilai pengurang sebesar 200 akan ditambahkan oleh metode AddDeductor. Terakhir, jika objek yang ditabrak memiliki *tag* "StreetRail", teks "Pelanggaran: Menabrak pembatas jalan." akan disimpan pada *file* log oleh metode WriteLog dan nilai pengurang sebesar 25 akan ditambahkan oleh metode AddDeductor.



Gambar 2. Diagram Aktivitas TriggerCheckForPavement

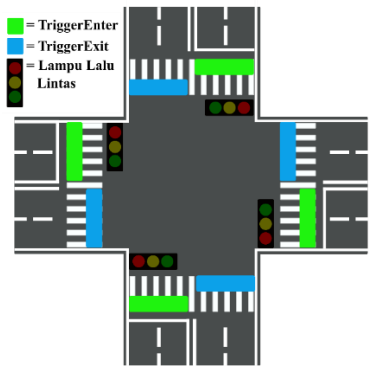
2. Deteksi Saat pemain keluar dari area jalan (masuk ke area trotoar)

Untuk sistem deteksi pelanggaran lalu lintas tipe kedua ini, *Trigger* digunakan. *Trigger* akan terpasang pada *prefab* trotoar jalan. Selain itu pada *prefab* trotoar jalan akan dipasangkan *script* TriggerCheckForPavement yang berfungsi untuk mendeteksi saat mobil pemain masuk ke daerah trotoar jalan (Ilustrasi pada gambar 2). *Script*

tersebut memiliki sebuah metode yang bernama OnTriggerEnter yang juga merupakan metode *built-in* Unity untuk anak kelas Monobehaviour. OnTriggerEnter akan dipanggil saat *collider* dari sebuah objek masuk ke area *trigger*. Pada mobil pemain akan dipasangkan 4 *collider* yang terletak di objek roda-roda mobil pemain. Nama objek dari *collider-collider* ini diawali dengan *string* "ColliderWheel" dan *script* TriggerCheckForPavement akan mengenali objek ini dari awalan *string* tersebut.
Saat pemain masuk ke daerah trotoar jalan, *Collider* dari mobil pemain akan masuk ke dalam *trigger* sehingga metode OnTriggerEnter pada *script* yang ada pada *trigger* akan aktif. Kode yang ada di dalam fungsi OnTriggerEnter ini memiliki alur seperti diagram di atas. Pertama program akan mengecek apakah *collider* yang masuk ke area *trigger* mengandung awalan *string* "ColliderWheel". Jika iya, teks "Pelanggaran: Menabrak trotoar jalan." akan disimpan pada *file* log oleh metode WriteLog dan nilai pengurang sebesar 20 akan ditambahkan oleh metode AddDeductor.

3. Deteksi Saat pemain melanggar lampu lalu lintas di persimpangan

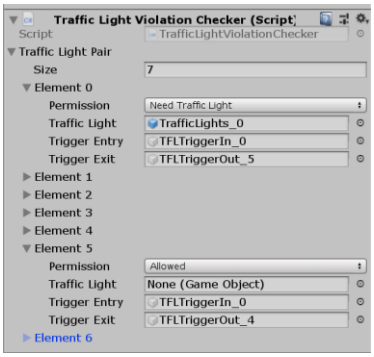
Untuk sistem deteksi pelanggaran lalu lintas tipe ketiga ini, sistem tersebut juga menggunakan *trigger*. *Trigger* ini akan dipasangkan di daerah perempatan yang memiliki lampu pengatur lalu lintas. Ada dua macam *trigger*, *trigger* saat pemain memasuki daerah persimpangan (*trigger entry*) dan *trigger* saat pemain keluar dari daerah persimpangan (*trigger exit*). Ilustrasi pada gambar 3.



Gambar 3. Rancangan peletakan trigger enter, trigger exit, dan lampu-lampu lalu lintas pada salah satu perempatan.

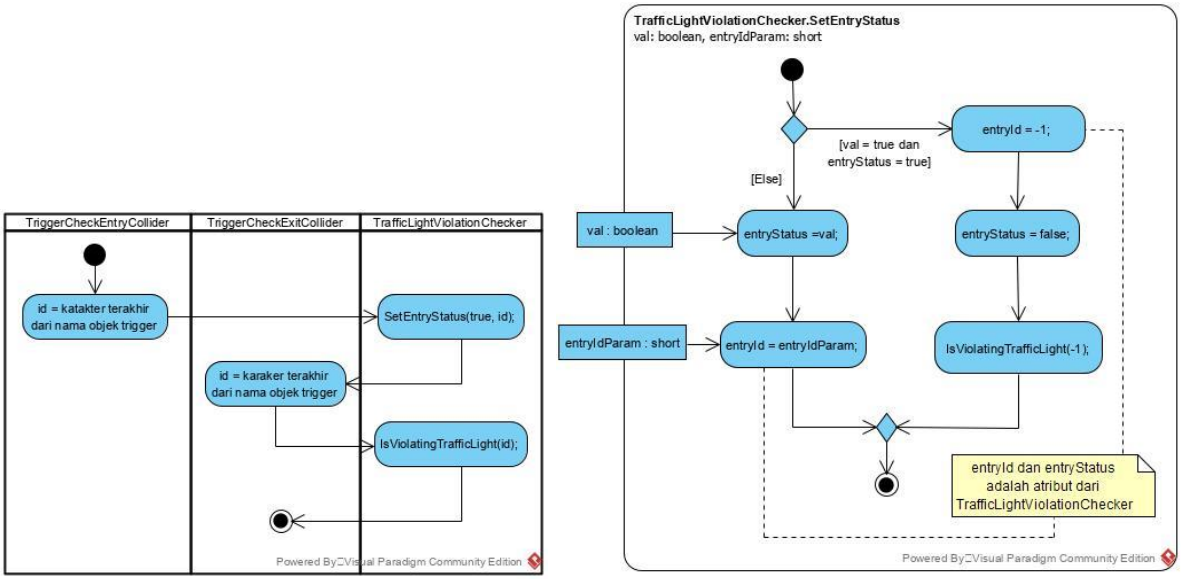
Pada sistem deteksi ini terdapat beberapa modul yang saling bekerja sama. Ketiga modul tersebut adalah TrafficLightViolationChecker (mendeteksi pelanggaran), TriggerCheckEntryCollider (merekam dari jalur mana mobil pemain datang), dan TriggerCheckExitCollider (merekan ke jalur mana mobil pemain pergi). Pasangan *trigger entry* dan *trigger exit* yang aktif karena dilewati mobil pemain akan menentukan apakah pemain melakukan pelanggaran atau tidak. Setiap pasangan ini akan dibuat dalam bentuk objek yang bernama TFLightPair. TFLightPair ini akan diinisialisasikan di dalam TrafficLightViolationChecker (gambar 4). Objek tersebut berfungsi untuk menyimpan objek *trigger entry*, *trigger exit*, lampu lalu lintas yang mengatur jalur tersebut, dan mengenai tipe izin dari jalur tersebut. Ada tiga tipe izin untuk objek TFLightPair yaitu:

1. *Need Traffic Light* : pelanggaran atau tidaknya ditentukan oleh lampu lalu lintas.
2. *Allowed* : tidak memerlukan lampu lalu lintas dan akan selalu dianggap tidak melanggar bila dilalui.
3. *Prohibited* : tidak memerlukan lampu lalu lintas dan akan selalu dianggap melanggar bila dilalui.



Gambar 4. Tampilan objek TFLightPair pada TrafficLightViolationChecker di Unity Editor

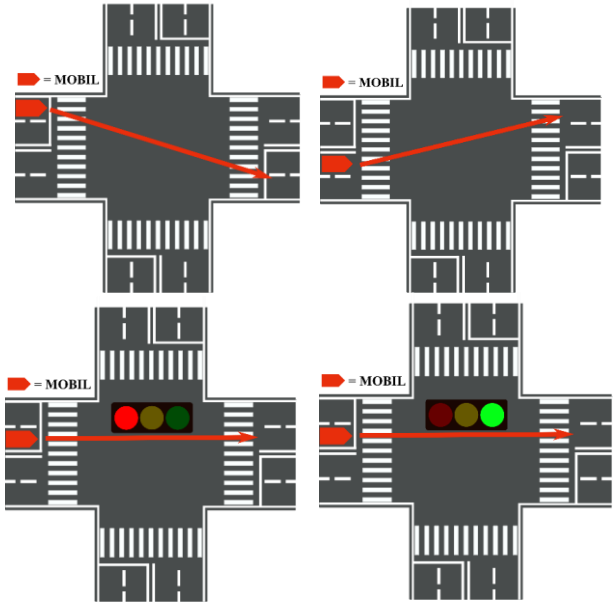
Saat *trigger entry* aktif (*trigger* dilewati mobil pemain), program akan merekam id dari objek *trigger enter* dan memanggil metode SetEntryStatus(true, id). Lalu saat mobil melewati trigger exit, program akan merekam id dari objek *trigger exit* dan memanggil metode IsViolatingTrafficLight(id). (Ilustrasi pada Gambar 5)



Gambar 5. Diagram aktivitas sistem untuk mengecek pelanggaran lampu lalu lintas (kiri) dan Diagram aktivitas metode SetEntryStatus kelas TrafficLightViolationChecker (kanan)

Metode *IsViolatingTrafficLight* berfungsi untuk menentukan apakah pemain melakukan pelanggaran atau tidak. Terdapat delapan macam kemungkinan yang dicek pada sistem deteksi di persimpangan ini yaitu (Ilustrasi dapat dilihat pada Gambar 6 dan Gambar 7)):

1) **Mobil masuk ke jalur yang salah.** Kemungkinan ini terjadi saat pemain melewati *trigger enter* dan kembali melewati *trigger enter* (artinya mobil masuk ke jalur yang berlawanan arahnya). Hal ini adalah pelanggaran dan skor akan dikurangi sebanyak 50 poin.
2) **Mobil keluar dari jalur yang salah.** Kemungkinan nomor 2 adalah kebalikan dari kemungkinan nomor 1. Mobil pemain masuk ke persimpangan melewati *trigger exit* dan keluar dari persimpangan melewati *trigger exit*. Hal ini dianggap sebagai pelanggaran dan akan dilakukan pengurangan skor sebesar 50 poin.

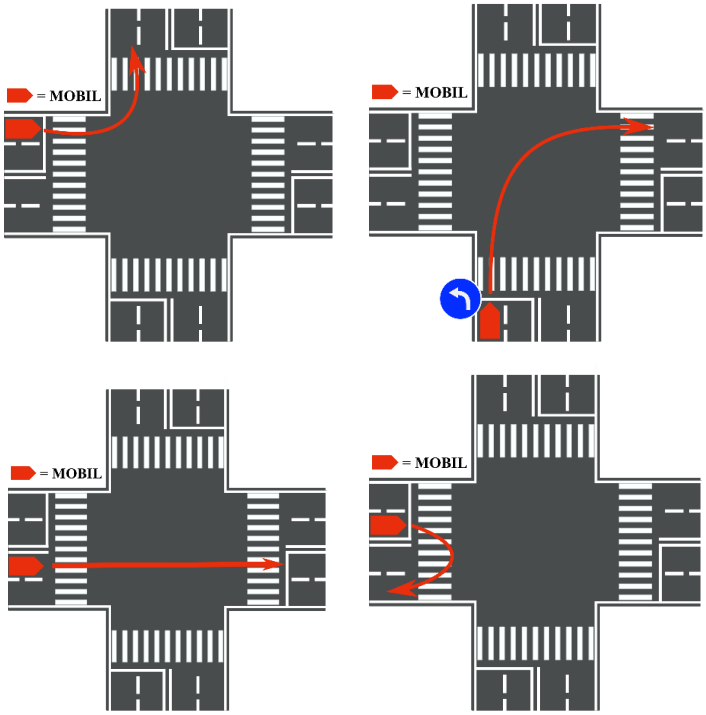


Gambar 6. Ilustrasi kemungkinan pelanggaran di persimpangan pertama (kiri atas), ke-2 (kanan atas), ke-3 (kiri bawah) dan ke-4 (kanan bawah)

3) **Mobil melewati jalur yang benar saat lampu lalu lintas sedang merah.** Kemungkinan ini terjadi saat mobil melewati *trigger enter* dan *trigger exit* dengan pengaturan izin "*Need Traffic Light*" serta lampu lalu

lintas yang mengatur pasangan *trigger* tersebut sedang merah. Kemungkinan ini adalah pelanggaran dan akan dilakukan pengurangan skor sebesar 50.

4) **Mobil melewati jalur yang benar saat lampu lalu lintas sedang hijau.** Kemungkinan ini terjadi saat mobil melewati *trigger enter* dan *trigger exit* dengan pengaturan izin "*Need Traffic Light*" serta lampu lalu lintas yang mengatur pasangan *trigger* tersebut sedang hijau. Kemungkinan ini bukan pelanggaran.
5) **Mobil melewati jalur yang diperbolehkan (tanpa diatur lampu lalu lintas).** Salah satu contoh skenario dari kemungkinan ini adalah belok kiri langsung. Pada simulasi, belok kiri langsung pada perempatan akan selalu diperbolehkan. Pada skenario seperti ini, pasangan *trigger enter* dan *trigger exit* akan memiliki izin "*Allowed*" yang artinya boleh dilalui. Skenario ini akan selalu dianggap sebagai bukan pelanggaran.
6) **Mobil masuk ke jalur yang dilarang dari jalur datang mobil tersebut.** Salah satu contoh dari skenario ini adalah pada saat di persimpangan terdapat tanda wajib belok dan mobil pemain tidak belok ke arah tersebut (dan belok ke jalur yang lain). Pada skenario seperti ini, pasangan *trigger enter* dan *trigger exit* akan memiliki izin "*Prohibited*" yang artinya dilarang untuk dilalui. Skenario ini akan selalu dianggap sebagai pelanggaran dan akan terjadi pengurangan skor sebesar 50.
7) **Mobil keluar dari dan masuk ke jalur yang salah saat melewati *trigger*.** Artinya mobil melewati *trigger exit* saat masuk ke persimpangan dan melewati *trigger enter* saat keluar dari persimpangan. Hal ini dianggap sebagai pelanggaran dan akan dilakukan pengurangan skor sebanyak 50 poin.
8) **Mobil melakukan putar balik di persimpangan.** Pada simulasi ini, putar balik pada persimpangan akan dianggap sebagai pelanggaran. Pelanggaran ini akan dikenai pengurangan poin sebesar 50 poin.



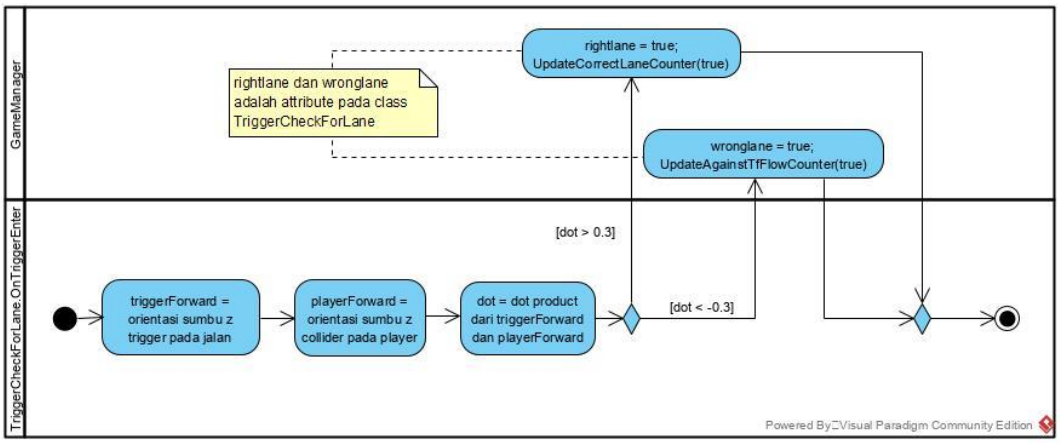
Gambar 7. Ilustrasi kemungkinan pelanggaran di persimpangan ke-5 (kiri atas), ke-6 (kanan atas), ke-7 (kiri bawah) dan ke-8 (kanan bawah)

Berikut adalah *pseudocode* dari metode IsViolatingTrafficLight pada TrafficLightViolationChecker.

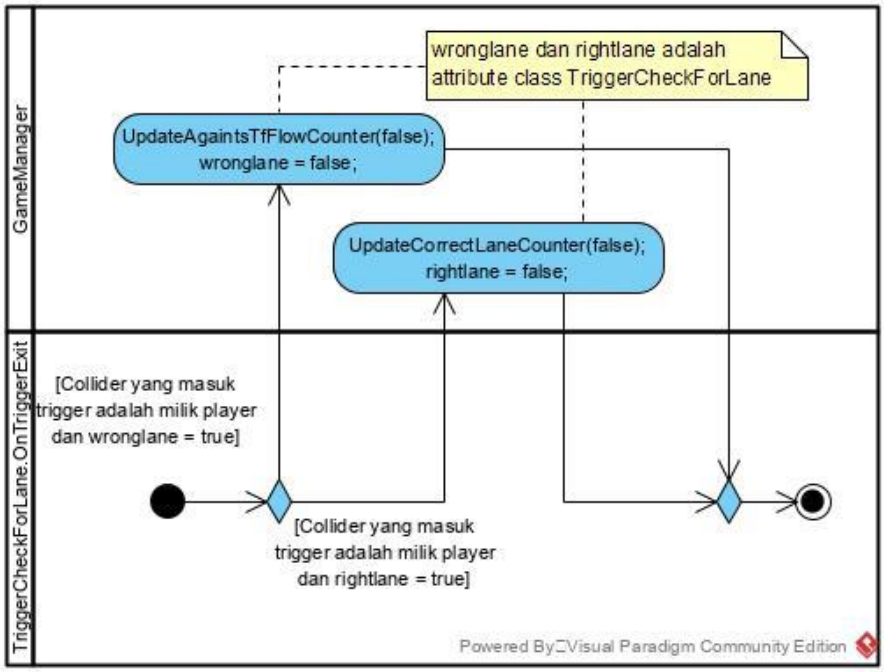
```
GLOBAL BOOLEAN entryStatus, INT entryId;

TrafficLightViolationChecker.IsViolatingTrafficLight(INT exitId) {
    tflPair = objek TFLightPair berdasarkan entryId dan exitId;
    BOOLEAN entryStat = entryStatus;
    INT idOfEntry = entryId;
    SetEntryStatus(FALSE, -1);

    IF exitId == -1:
        WriteLog("Mobil pemain masuk ke jalur yang salah.");
        Kurangi 50 skor dan atur ulang multiplier menjadi 1;
        RETURN;
    ELSEIF idOfEntry == -2:
        WriteLog("Mobil pemain keluar dari jalur yang salah.");
```

```
    Kurangi 50 skor dan atur ulang multiplier menjadi 1;
    RETURN;
ELSEIF myObject != NULL AND entryStat == TRUE:
    SetEntryStatus(FALSE);
    INT state = nilai variable state dari objek tflPair;
    IF state == 0:
        IF lampu lalulintas != hijau:
            WriteLog ("Mobil pemain melanggar lampu merah.");
            Kurangi 50 skor dan atur ulang multiplier menjadi 1;
            RETURN;
        ELSE:
            DebugLog("Lampu sedang hijau, bukan pelanggaran.")
        ENDIF.
    ELSEIF state == 1:
        DebugLog("Mobil melewati dari jalur yang" +
                 "diperbolehlan, bukan pelanggaran")
        RETURN;
    ELSE:
        WriteLog ("Mobil pemain melewati jalur yang dilarang.");
        Kurangi 50 skor dan atur ulang multiplier menjadi 1;
        RETURN;
    ENDIF.
ELSEIF entryStat == FALSE:
    WriteLog("Mobil pemain keluar dari dan masuk ke" +
             "jalur yang salah.");
    Kurangi 50 skor dan atur ulang multiplier menjadi 1;
    SetEntryStatus(TRUE, -2);
    RETURN;
ELSE:
    WriteLog("Mobil pemain melakuakan putar balik di perempatan.");
    Kurangi 20 skor dan atur ulang multiplier menjadi 1;
    RETURN;
ENDIF.
```

DebugLog adalah metode untuk mencetak teks pada *debugging console* di Unity sedangkan WriteLog adalah metode untuk menyimpan teks ke dalam *file* log yang mencatat pelanggaran yang dilakukan pemain. Objek *tflPair* adalah objek *TFLightPair* yang ada pada gambar 7 bagian kanan bawah, Variabel entryStat adalah variabel yang berfungsi untuk menandakan apakah mobil pemain telah melewati *trigger entry* atau belum. Nilai *true* artinya mobil sudah melewati *trigger entry* dan *false* berarti mobil belum melewati *trigger entry*. Variabel *idOfEntry* berfungsi untuk menyimpan id dari *trigger entry* yang dilewati. Hal ini dilakukan untuk mengetahui dari jalur mana mobil pemain berasal (dilihat dari *trigger entry* mana yang dilewati oleh mobil). Dengan menggunakan bantuan kedua variabel tersebut, sistem dapat menentukan apakah pemain melakukan pelanggaran saat melewati persimpangan atau tidak beserta pelanggaran apa yang dilakukan.



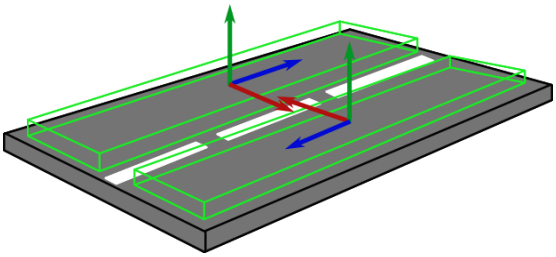
Gambar 8. Diagram aktivitas metode OnTriggerEnter pada kelas TriggerCheckForLane



Gambar 9. Diagram aktivitas metode OnTriggerExit pada kelas TriggerCheckForLane

4. Deteksi Saat pemain berkendara melawan arus lalu lintas.

Untuk sistem deteksi pelanggaran lalu lintas tipe keempat ini, *trigger* tetap digunakan. *Trigger* yang ada akan dipasang pada *prefab* jalan. *Trigger* akan dipasang dengan orientasi tertentu sehingga *Trigger* akan memiliki sumbu z yang searah dengan arah arus lalu lintas yang seharusnya. Pada Unity, sumbu z positif menandakan sisi depan objek. Sehingga saat sumbu z mobil memiliki orientasi yang berbeda dengan sumbu z *trigger* jalan, mobil dianggap melawan arus.



Gambar 10. Orientasi sumbu z (panah biru) *trigger* (balok hijau) pada jalan yang menunjukkan arah arus lalu lintas pada jalur jalan tersebut

Pada diagram ada dua metode yang digunakan untuk sistem deteksi pelanggaran lalu lintas. Dua metode itu adalah OnTriggerEnter dan OnTriggerExit yang ada pada kelas TriggerCheckForLane (Ilustrasi pada gambar 8 dan gambar 9). Saat mobil masuk ke dalam *trigger*, metode OnTriggerEnter akan aktif dan akan menentukan apakah mobil pemain sedang melawan arus atau tidak. Bila iya, variabel rightlane akan bernilai *true*. Bila tidak, variabel wronglane lah yang akan bernilai *true*. Selanjutnya sistem juga akan memanggil metode UpdateCorrectLaneCounter(true) jika tidak melawan arus dan memanggil metode UpdateAgaintsTfFlowCounter(true) jika melawan arus.

Metode UpdateCorrectLaneCounter akan memperbarui *counter* yang menghitung berapa lama pemain tidak melawan arah arus (yang nantinya akan berguna dalam penambahan skor tiap waktu dan pengali skor). Metode UpdateAgaintsTfFlowCounter akan memperbarui *counter* yang menghitung berapa lama pemain melawan arah arus (yang nantinya akan digunakan untuk pengurangan skor tiap waktu dan pengali skor).

Saat pemain keluar dari *trigger*, metode OnTriggerExit akan dipanggil. Metode ini akan melihat dan menentukan apakah mobil pemain melawan arus atau sudah memperbaiki posisinya saat keluar dari *trigger*. Jika pemain melawan arus dan sebelumnya pemain tidak melawan arus, metode ini akan mengubah status rightlane menjadi *false* dan akan memanggil metode UpdateCorrectLaneCounter(false) yang akan memperbarui *counter* ke

arah negatif (yang artinya pemain sedang melawan arus lalu lintas). Jika pemain tidak melawan arus dan sebelumnya pemain melawan arus, metode ini akan mengubah status wronglane menjadi *false* dan akan memanggil metode UpdateAgaintsTfFlowCounter(false) yang akan memperbarui *counter* ke arah negatif (yang artinya pemain sedan tidak melawan arus lalu lintas) (Ilustrasi pada gambar 10).

Sistem deteksi ini akan menambahkan skor dan meningkatkan pengali skor bila pemain berkendara tanpa melakukan pelanggaran apa pun. Sistem ini juga akan mengurangi skor dan meningkatkan pengali saat pengurangan skor apabila pemain tetap melawan arus, sehingga semakin lama pemain melawan arus, semakin besar pula poin skor yang dikurangi.

III.2. Perancangan Sistem Penilaian

Saat berkendara, skor akan terus bertambah jika pemain berkendara tanpa melakukan pelanggaran. Semakin lama pemain dapat berkendara tanpa melakukan pelanggaran, semakin besar nilai pengali skor. Saat pemain melakukan pelanggaran, sistem pengali skor akan dibuat kembali menjadi satu. Selain itu apabila pemain melakukan pelanggaran melawan arus lalu lintas, pengali skor akan menjadi minus dan bertambah selama pemain tetap berkendara melawan arus lalu lintas. Pengali skor memiliki nilai (berupa bilangan bulat) negatif empat sampai positif empat. Setiap kali pemain melakukan pelanggaran, skor akan langsung dikurangi. Setiap pelanggaran akan memiliki bobot yang berbeda. Misalnya bobot saat pemain menabrak pejalan kaki akan lebih besar dari pada menabrak pembatas jalan. Skor akhir akan didapatkan setelah simulasi selesai dan akan disimpan dalam *file* log yang ada.

Untuk mengurangi skor nilai, seluruh sistem deteksi pelanggaran yang ada pada simulasi akan menambahkan nilai pada variabel deductor di GameManager (kecuali untuk deteksi pelanggaran melawan arus). Hal tersebut dilakukan untuk membuat sistem pengurangan skor lebih terpusat. Setiap sistem deteksi pelanggaran tertentu harus memanggil metode AddDeductor milik GameManager untuk menambahkan nilai pada variabel deductor. Selanjutnya pengurangan skor akan diproses oleh GameManager melalui metode UpdateScore. Potongan kode metode UpdateScore yang telah dibuat adalah sebagai berikut.

```
private void UpdateScore() {

    int state = CheckIfAgaintsTfFlow();
    if (state == -1) {
        if (multiplier < 100 && isMultiplierPositive) {
            if (multiplier < 20f)
                multiplier = 20f;
            multiplier += 0.25f;
        }
        isMultiplierPositive = true;
        score += 1 * Mathf.FloorToInt(multiplier / 20f);
    } else if (state == 1) {
        if (multiplier > -100 && !isMultiplierPositive) {
            if (multiplier > -20f) {
                multiplier = -20f;
                WriteLog("Pelanggaran: Melawan arus lalu lintas.");
            }
            multiplier -= 0.25f;
        }
        isMultiplierPositive = false;
        score += 1 * Mathf.CeilToInt(multiplier / 20f);
    }

    score -= deductor;
    scoreText.text = "Score: " + score.ToString();
    if (isMultiplierPositive)
        multiplierText.text = "COMBO " + Mathf.FloorToInt(multiplier /
                20f).ToString() + "x";
    else
        multiplierText.text = "COMBO " + Mathf.CeilToInt(multiplier /
                20f).ToString() + "x";
    deductor = 0;
}
```

Sebelum mengurangi skor dengan nilai yang tersimpan pada variabel deductor, program akan melihat apakah mobil akan memanggil metode CheckIfAgainstTfFlow dan menyimpan nilai *return* metode tersebut ke dalam variabel state. Jika state sama dengan negatif satu, nilai variabel multiplier akan dibuat menjadi 20 (jika nilainya kurang dari 20) dan akan ditambahkan sebesar 0,25. Jika state sama dengan satu, variabel multiplier akan dibuat menjadi -20 (jika nilainya lebih dari -20) dan akan dikurangi nilai sebesar -0,25. Variabe multiplier memiliki

rentang -100,0 sampai 100,0. Pada saat akan dikalikan dengan penambahan / pengurangan skor, variabel multiplier akan dibagi dengan 20 dan dijadikan sebagai bilangan bulat. Hal ini membuat nilai pengali skor berada di rentang -5 sampai 5. Selanjutnya program akan mengurangi skor dengan seluruh nilai yang tersimpan dalam variabel deductor dan deductor akan dibuat menjadi 0 kembali. Lalu program akan memperbarui skor dan pengali skor pada antarmuka pengguna. Metode UpdateScore akan dipanggil 4 kali dalam 1 detik. Artinya untuk sistem deteksi pelanggaran melawan arus lalu lintas, skor akan bertambah atau berkurang (bergantung pada nilai multiplier) sebanyak $1 \times multiplier$ / 20 dalam ¼ detik.

III.2. Uji Coba Fungsionalitas Sistem

Pengujian sistem deteksi pelanggaran lalu lintas virtual secara fungsionalitas dilakukan dengan menggunakan teknik uji coba *black* box. Hasil uji coba fungsionalitas dapat dilihat pada Tabel 1 di bawah ini.

Tabel 1. Hasil Uji Fungsionalitas Sistem Deteksi Pelanggaran dan *Log File*

No	Fungsi yang Diuji	Input/Aksi	Hasil yang Diharapkan	Hasil Uji
1	Deteksi melawan arus	Mobil melaju di jalur sebelah kanan.	Terjadi pengurangan skor, pengubahan *combo* menjadi negatif, dan pencatatan pelanggaran pada log.	Berfungsi
2	Deteksi masuk ke area trotoar	Mobil masuk ke area trotoar.	Terjadi pengurangan skor, pengubahan *combo* menjadi 1, dan pencatatan pelanggaran pada log.	Berfungsi
3	Deteksi menabrak kendaraan	Mobil menabrak kendaraan.	Terjadi pengurangan skor, pengubahan *combo* menjadi 1, dan pencatatan pelanggaran pada log.	Berfungsi
4	Deteksi menabrak pejalan kaki	Mobil menabrak pejalan kaki.	Terjadi pengurangan skor, pengubahan *combo* menjadi 1, dan pencatatan pelanggaran pada log.	Berfungsi
5	Deteksi menabrak pembatas jalan	Mobil menabrak pembatas jalan.	Terjadi pengurangan skor, pengubahan *combo* menjadi 1, dan pencatatan pelanggaran pada log.	Berfungsi
6	Deteksi mobil masuk ke jalur yang salah di persimpangan.	Mobil keluar dari jalur yang benar namun masuk ke jalur yang melawan arus lalu lintas.	Terjadi pengurangan skor, pengubahan *combo* menjadi 1, dan pencatatan pelanggaran pada log.	Berfungsi
7	Deteksi mobil keluar dari jalur yang salah di persimpangan.	Mobil keluar dari jalur yang salah lalu kembali ke jalur yang benar.	Terjadi pengurangan skor, pengubahan *combo* menjadi 1, dan pencatatan pelanggaran pada log.	Berfungsi
8	Mobil melewati jalur yang benar saat lampu lalu lintas sedang merah di persimpangan.	Mobil melewati jalur yang benar namun melanggar lampu merah.	Terjadi pengurangan skor, pengubahan *combo* menjadi 1, dan pencatatan pelanggaran pada log.	Berfungsi
9	Mobil melewati jalur yang benar saat lampu lalu lintas sedang hijau di persimpangan	Mobil melewati jalur yang benar saat lampu hijau.	Tidak terjadi apa-apa.	Berfungsi
10	Mobil melewati jalur yang diperbolehkan (tanpa diatur lampu lalu lintas) di persimpangan	Mobil melewati jalur yang diperbolehkan untuk dilewati tanpa perlu lampu lalu lintas.	Tidak terjadi apa-apa.	Berfungsi

11	Mobil masuk ke jalur yang dilarang di persimpangan.	Mobil masuk ke jalur yang tidak diperbolehkan dari jalur asal mobil datang.	Terjadi pengurangan skor, pengubahan *combo* menjadi 1, dan pencatatan pelanggaran pada log.	Berfungsi
12	Mobil keluar dari dan masuk ke jalur yang salah di persimpangan.	Mobil melewati persimpangan dan melawan arus.	Terjadi pengurangan skor, pengubahan *combo* menjadi 1, dan pencatatan pelanggaran pada log.	Berfungsi
13	Mobil melakukan putar balik di persimpangan	Mobil melakukan putar balik.	Terjadi pengurangan skor, pengubahan *combo* menjadi 1, dan pencatatan pelanggaran pada log.	Berfungsi

Selanjutnya dilakukan juga pengujian fungsionalitas antar muka menu *reports*. Hal ini dilakukan untuk melihat apakah semua hasil pencatatan *score* dari pengguna simulator sudah benar tercatat dan tersimpan dengan baik. Hasil pengujian fungsionalitas dapat dilihat pada Tabel 2.

Tabel 2. Hasil Uji Fungsionalitas Antarmuka Menu *Reports*

No	Fungsi yang Diuji	Input/Aksi	Hasil yang Diharapkan	Hasil Uji
1	Daftar laporan pada menu *Reports*	Masuk ke antarmuka menu *Reports*.	Muncul daftar laporan-laporan dari simulasi-simulasi yang pernah dilakukan (bila ada).	Berfungsi
2	Daftar laporan pada menu *Reports*	Klik kiri *mouse* pada salah satu laporan.	Muncul isi laporan pada antarmuka bagian kanan.	Berfungsi
3	Sistem *scroll* pada daftar laporan.	Memutar *mouse wheel* saat kursor berada di atas daftar laporan.	Melakukan *scroll* ke bawah atau ke atas pada daftar laporan sesuai dengan gerakan *mouse wheel*.	Berfungsi
4	Sistem *scroll* pada isi laporan.	Memutar *mouse wheel* saat kursor berada di atas isi laporan.	Melakukan *scroll* ke bawah atau ke atas pada isi laporan sesuai dengan gerakan *mouse wheel*.	Berfungsi
5	Tombol *Back*	Klik kiri *mouse* pada tombol.	Antarmuka berpindah dari menu *Reports* ke *Main Menu*.	Berfungsi

IV. KESIMPULAN & PENGEMBANGAN RISET SELANJUTNYA

Simulator mobil dengan sistem deteksi pelanggaran lalu lintas telah berhasil dibuat dan memenuhi tujuan penelitian dan mencapai seluruh fungsionalitas sistem yang telah disebutkan. Simulator mobil dibuat menggunakan Unity *Game Engine*. Dari hasil percobaan, simulator mobil yang dibuat telah berhasil pada seluruh uji coba fungsionalitas yang ada. Simulator dibuat dengan sistem deteksi pelanggaran lalu lintas untuk:

1) Deteksi tabrakan dengan mobil, pejalan kaki, dan pembatas jalan
2) Deteksi pelanggaran melawan arus lalu lintas
3) Deteksi pelanggaran lampu lalu lintas pada persimpangan
4) Deteksi pelanggaran masuk ke area trotoar

Pengembangan lebih lanjut dapat dilakukan untuk menyempurnakan simulator mengemudi ini. Pelengkapan fitur-fitur pada mobil seperti fungsi fisika dapat ditambahkan untuk memperkuat kesan realitas, misalnya adanya kerusakan kendaraan setelah menabrak sesuatu dan juga efek cedera pada pengendaranya. Beberapa hal seperti lingkungan berkendara, jenis mobil, dan sistem deteksi pelanggaran yang lain pun dapat ditambahkan untuk menyempurnakan simulator ini. Terutama pelanggaraan-pelanggaran terkait rambu-rambu lalu lintas dan pelanggaran lainnya yang tercantum dalam UU lalu-lintas Negara Republik Indonesia untuk memperkaya realitas simulator mengemudi ini.

V. DAFTAR PUSTAKA

[1] Allen, R. et al., (2003). Novice driver training results and experience. PROCEEDINGS of the Second International Driving Symposium on Human Factors in Driver Assessment, Training and Vehicle Design, pp. 165-170.
[2] Duchowski, A. T. (2002). A breadth-first survey of eye-tracking applications. *Behavior Research Methods, Instruments, & Computers, 34*(4), 455-470. https://doi.org/10.3758/BF03195475
[3] Mohammad Iqbal, Karmila Sari, Kemal Ade Sekarwati and Dian Kemala Putri, 2020, Developing PC-Based Driving Simulator System for Driver Behavior Analysis Research, Journal of Physics: Conference Series, Volume 1566, 4th International Conference on Computing and Applied Informatics 2019 (ICCAI 2019) 26-27 November 2019, Medan, Indonesia.
[4] Underwood, G., (2007). Visual attention and the transition from novice to advanced driver. Ergonomics, 50(8), pp. 1235-1249.
[5] _____, (2020a). Unity manual: colliders. Unity Technologies. [Online] Available at: https://docs.unity3d.com/2019.2/Documentation/Manual/CollidersOverview.html [Diakses 19 Maret 2020].
[6] _____, (2020b). Unity manual: prefabs. Unity Technologies [Online] Available at: https://docs.unity3d.com/2019.2/Documentation/Manual/Prefabs.html [Diakses 19 Maret 2020].
[7] _____, (2020c). Visual Studio C# integration. Unity Technologies [Online] Available at: https://docs.unity3d.com/Manual/VisualStudioIntegration.html [Diakses 20 Juli 2020].

UCAPAN TERIMA KASIH

Proyek penelitian ini didukung oleh Kementerian Riset, Teknologi dan Pendidikan Tinggi Republik Indonesia dalam skema Penelitian “Penelitian Unggulan Perguruan Tinggi” tahun 2020 dan oleh Universitas Gunadarma. Ucapan terima kasih kepada para dosen yang terlibat Karmila Sari, Dian Kemalaputri dan Kemal Ade, serta kepada mahasiswa kami yang telah menyelesaikan tugasnya dengan sangat baik dan memberikan artikel yang berkualitas untuk tugas akhir, yaitu Dea Chintia P, Iwang Naufal, Rio Setyo Nugroho dan Fathiyah Hilyah A.