PEMBUATAN GAME RACING 2D "SAFETY RIDING" DENGAN PENERAPAN ALGORITMA PERLIN NOISE UNTUK MEMBUAT PETA DENGAN UNITY

Moh.Ibnu Nadhir[1], Ari Kurniawan[2]

Manajemen Informatika, Program Vokasi
Universitas Negeri Surabaya, Surabaya, Indonesia
[1]moh.19020@mhs.unesa.ac.id, [2]arikurniawan@unesa.ac.id

Abstrak— ***Game merupakan sebuah permainan dengan adanya sistem yang melibatkan player dengan konflik buatan yang telah ditentukan atau adanya peraturan untuk memberikan hasil yang terukur dan adanya interaksi antara player dengan sistem didalam permainan tersebut. Aplikasi permainan terbagi menjadi dua yaitu aplikasi permainan 2D (dua dimensi) dan 3D (tiga dimensi). Metode Perlin Noise merupakan teknik yang digunakan untuk mensimulasikan sesuatu yang terlihat berantakan dan natural. Peta merupakan rancangan dua dimensi yang berskala medium dari kumpulan objek untuk membuat suatu jalan, tempat, untuk mengetahui apa saja yang ada disekitar, biasanya digunakan untuk titik tujuan.Map sangat penting dalam permainan petualangan. Genre permainan petualangan adalah genre dengan setting tempat yang luas untuk menjelajahi suatu tempat atau wilayah. Unity merupakan software yang dapat digunakan untuk pengembangan game yang digunakan diberbagai platform..***

Kata kunci— **Game, Perlin Noise, Petualangan, Unity, Map**

I. PENDAHULUAN

Menurut [1]sebanyak 1,24 juta orang meninggal setiap tahun di seluruh dunia akibat kecelakaan lalu lintas, sementara 20–50 juta orang mengalami luka-luka. Kecelakaan lalu lintas juga menjadi penyebab utama kematian anak di seluruh dunia, dengan rata-rata 1000 anak dan remaja meninggal setiap harinya dalam rentang usia 10–24 tahun. Di Indonesia, kecelakaan lalu lintas merupakan penyebab kematian terbesar ketiga setelah penyakit jantung koroner dan tuberkulosis, menurut penilaian WHO.[2].

Perkembangan teknologi di industri game menghasilkan variasi game yang semakin beragam. Saat ini, topik yang banyak diteliti oleh peneliti dalam pembuatan konten game adalah AI, terutama menggunakan metode *Procedural Content Generation (PCG*). Metode ini sukses diterapkan dalam berbagai genre game seperti *RPG, Petualangan, Platformer, Racing, SandBox, dan Simulation.*

Dalam pembuatan game racing, elemen pentingnya adalah pembuatan map secara otomatis tanpa batasan tertentu. Penggunaan metode PCG dengan algoritma Perlin Noise menjadi solusi untuk menghasilkan map yang terlihat natural karena mengikuti bentuk gradien yang halus daripada pembuatan acak. Penelitian sebelumnya telah berhasil menciptakan kota layak menggunakan Perlin Noise.

Perlin Noise dapat mempresentasikan bentuk daratan yang lebih terlihat natural dibandingakan dengan membuatnya secara acak karena dapat menghasilkan angka acak terstruktur mengikuti bentuk gradien yang halus. Pada penelitian yang dilakukan oleh Olsson dan Frank pada 2017, mereka berhasil menghasilkan sebuah kota yang layak menggunakan Perlin Noise.

Tujuan penelitian ini adalah membangun game racing dengan pembuatan map atau daratan menggunakan algoritma Perlin Noise. Game ini bernama "Safety Riding" dan juga berfungsi untuk mengedukasi masyarakat dan pemain game agar lebih berhati-hati dan patuh terhadap peraturan berkendara. Pembuatan game ini menggunakan PCG untuk menghasilkan konten secara otomatis dalam bentuk 2D.

II. METODOLOGI PENELITIAN

A. Pendekatan Penelitian

Pendekatan ini menggunakan metode pengembangan waterfall. Metode pengembangan dari pendekatan yang digunakan dalam penelitian deskriptif-kualitatif. Metode Waterfall ini merupakan metode pengembangan pada perangkat lunak yang diterapkan secara beruntun, yang dianggap seperti air terjun dengan melewati fase-fase beruntun mulai dari analisa kebutuhan (Reqruirements), desain, implementasi dan testing.

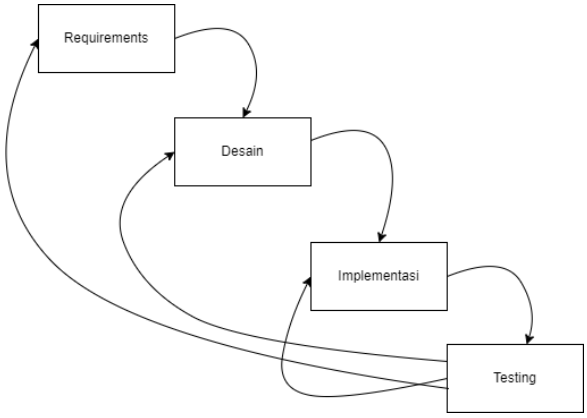
B. Sumber Data Penelitian

Peneliti mengumpulkan data sekunder. Berikut adalah metode yang digunakan dalam mencari dan mengumpulkan data sumber sekunder.

C. Studi Literatur

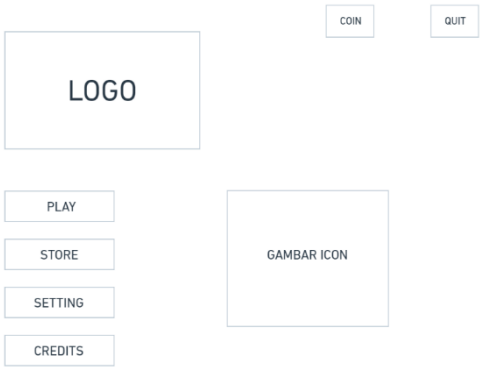
Pada tahap awal penelitian ini, dimulai dengan membuat studi literatur yang digunakan untuk menemukan suatu masalah dan mencari sumber-sumber terkait dengan penelitian yang sedang dilakukukan,beberapa tinjauan pustaka yang digunakan sebagai acuan untuk memulai penelitian ini. Terkait literatur yang dilakukan oleh peneliti menggunakan internet dengan menggunakan beberapa situs jurnal-jurnal seperti *Google Scholar*, *ScientDirect* dan masih banyak lagi dengan menggunakan kata kunci seperti *"perlin noise","game dengan genre racing","pembuatan map dengan perlin noise","kecelakaan lalu lintas di indonesia"* dan lainnya.

D. Alur Penelitian



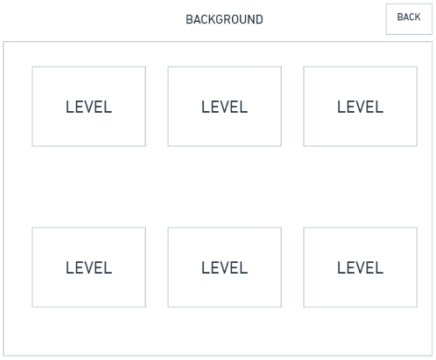
Gambar 1 Alur Penelitian

1. Analisa Kebutuhan (*Requirements*)
   Pada tahap selanjutnya dalam proses penelitian adalah menganalisa kebutuhan yang dibutuhkan untuk membuat sebuah game. Analisa kebutuhan yang pertama yaitu melakukan studi literatur tentang membuat map atau peta dengan perlin noise, peraturan berkendara yang aman dan mencari gagasan ide untuk game yang bergenre racing dua dimensi. Selanjutnya merinci seluruh kebutuhan yang dibutuhkan pada proses perancangan desain asset game dan lainnya.
2. Perancangan Desain (Design)
   Semua kebutuhan yang sudah didapatkan pada proses sebelumnya kemudian diimplementasikan kedalam bentuk desain sistem. Proses perancangan pada desain game dengan membuat sebuah *wireframe* dari game yang akan dibuat, mulai dari tampilan awal mulai, sampai dengan akhir game dapat dilihat pada *wireframe*.



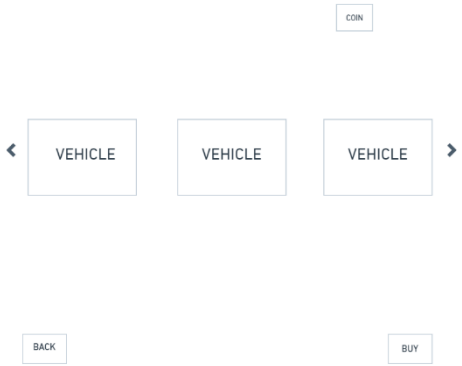
Gambar 2 Wireframe Main Menu

Gambar 2 merupakan tampilan dari main menu setelah mengklik button start pada scene awal, pada scene main menu ini terdapat *button play*, *store*, *setting*, *credits*, *coin* dan *quit*. Untuk tampilan lainnya terdapat logo dari game dan gambar *icon game.*



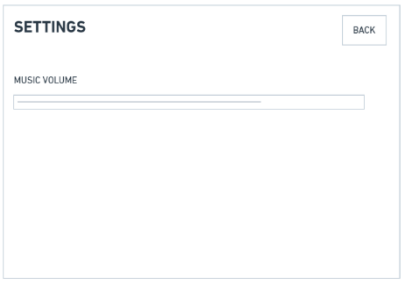
Gambar 3 Wireframe Pilih Level

Pada gambar 3 merupakan tampilan setelak mengklik *button play* dari main menu, terdapat pilihan level stage, *player* dapat memilih level stage untuk dimainkan. Jika belum menyelesaikan lavel sebelumnya, maka level selanjutnya tidak terbuka.



Gambar 4 Wireframe Menu Shop

Gambar 4 merupakan *wireframe* dari *store* yang ada di main menu, yang digunakan untuk membeli kendaraan dengan coin yang sudah didapatkan didalam permainan.

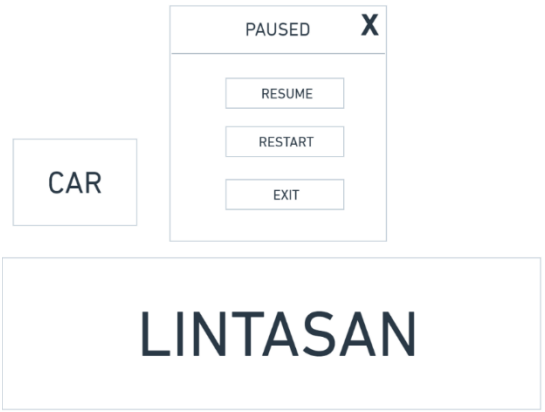


Gambar 5 Wireframe Menu Setting

Pada gambar 5 merupakan *wireframe* dari setting menu, pada tampilan *setting menu*, *player* dapat mengatur *music volume* dari game.

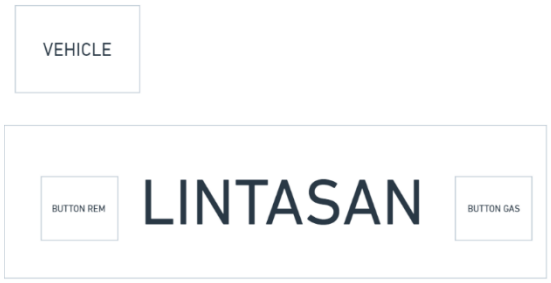
Gambar 6 Wireframe Keluar Game

Pada gambar 3.6 merupakan wireframe confirm dari quit untuk keluar dari game, *player* mengkonfirmasi ulang jika mau keluar dari game.



Gambar 7 Wireframe Pause Menu

Gambar 7 merupakan wireframe dari pause menu didalam game yang terdapat button resume untuk melanjutkan game, restart untuk mengulang permainan dan exit untuk keluar dari permainan.



Gambar 8 Wireframe In Game

Gambar 8 menjelaskan tampilan wireframe dari inti permainan, yang dimana scene setelah memilih kendaraan. Untuk button terdapat button gas dan rem, lalu lintasan atau map yang sudah dibuat dengan perlin noise.

3. Implementasi

   Implementasi sistem adalah tahapan implementasi dari materi hasil pada tahapan sebelumnya.Pertama implementasi dilakukan dengan membuat tahap pembangunan sejumlah resource *game* yang dibutuhkan, membuat tilesmap menggunakan algoritma perlin noise. Selanjutnya adalah melakukan proses pembuatan *game* yang menggunakan software unity dengan bahasa pemrograman C#.

4. Testing

   Pengujian game merupakan tahap akhir dari penelitian ini. Pengujian game dilakukan dengan menguji hasil rancangan sesuai dengan analisis kebutuhan awal. Pengujian ini bertujuan untuk mengetahui hasil dari pembentukan dunia *game* yang sudah dibuat,apakah map untuk berkendara yang dihasilkan sudah terstruktur rapi dan terlihat natural.

   Bentuk dari rancangan *game* yang pertama adalah menentukan variabel jangkauan *noise* yang akan membentuk sebagai pondasi *perlin noise*. Hasil dari *perlin noise* yang diperoleh akan diimplementasikan kedalam sebuah bentuk tile map untuk memperoleh bentuk dunia.

III. HASIL DAN PEMBAHASAN

A. Hasil Penelitian

Berdasarkan alur penelitian tahap analisa kebutuhan (requirement), tahap perancangan desain (design). Tahap implementasi, dan tahap pengujian (testing), langkah selanjutnya adalah implementasi Game Racing “Safety Riding” di android. Game dibuat berdasarkan tahap perancangan desain.

1. Implementasi Menu Utama

Pada tampilan menu utama terdapat tombol Play, tombol Shop, tombol Setting, dan tombol Keluar. Pengguna dapat memilih tombol play jika ingin langsung memainkannya, untuk menuju menu memilih new game dan continue. Tombol Setting sebelah kanan pengguna digunakan untuk mengatur besar kecilnya volume backsound atau musik yang ada didalam game. Tombol shop sebelah kiri pengguna digunakan untuk memilih kendaraan untuk bermain. Pada bagian atas terdapat logo dari game yaitu Safety Riding. Pada bagian kiri atas terdapat tulisan yang menandakan coin yang diperoleh oleh pengguna dalam bermain dan digunakan untuk membeli kendaraan pada menu shop, pada bagian pojok kanan bawah adalah menu untuk melihat peraturan dan tombol yang digunakan didalam game, dan pada bagian kanan atas terdapat tombol keluar digunakan untuk keluar dari game.

Gambar 9 Tampilan Menu Utama

2. Implementasi Menu Shop

Tampilan *shop* muncul ketika *player* mengklik tombol shop dimenu utama. *Shop* berfungsi untuk membeli kendaraan yang digunakan sebagai karakter didalam game. Pada bagian tengah tampilan shop terdapat tombol kanan dan kiri berfungsi untuk memilih kendaraan yang ingin dibeli. Dibagian tengah ada tombol *price* beserta harga untuk kendaraan berfungsi sebagai tombol untuk pembelian kendaraan ketika coin sudah mencukupi. Pada bagian bawah terdapat tombol *back* atau kembali yang berfungsi untuk kembali ke menu utama.

Gambar 10 Tampilan Menu Shop

3. Implementasi Menu Settinh

Tampilan *setting* muncul ketika *player* mengklik tombol setting dimenu utama. *Setting* berfungsi untuk mengatur besar kecilnya *volume bakcsound* yang ada didalam game dengan cara menggeser kiri atau kekanan slider pada bagian tengah yang berwarna *orange*. Pada bagian bawah ada tombol X untuk keluar dari menu setting ke menu utama.

Gambar 11 Tampilan Menu Setting

4. Implementasi Menu Tutorial

Tampilan *Tutorial* muncul ketika *player* mengklik tombol *tutorial* pada bagian pojok kanan bawah dimenu utama. *Tutorial* berfungsi untuk melihat peraturan yang ada didalam game dan tombol yang digunakan. Pada bagian bawah ada tombol X untuk keluar dari menu setting ke menu utama.

Gambar 12 Tampilan Menu Tutorial

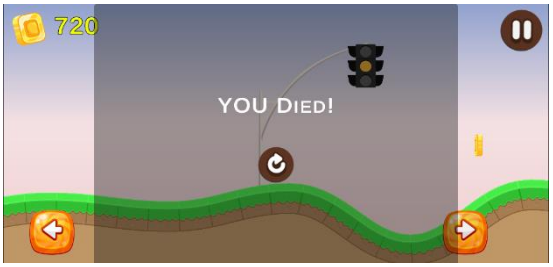
5. Implementasi Menu Keluar Game

Tampilan keluar game muncul ketika *player* mengklik tombol X pada pojok kanan atas dimenu utama. Tombol X berfungsi untuk keluar dari game. Terdapat dua tombol *yes* dan *no* jika *player* mengklik tombol yes akan keluar dari game. Jika *player* mengklik *no* maka akan kembali didalam game dan ke menu utama.

Gambar 13 Tampilan Menu Keluar Game

6. Implementasi Menu Game Over

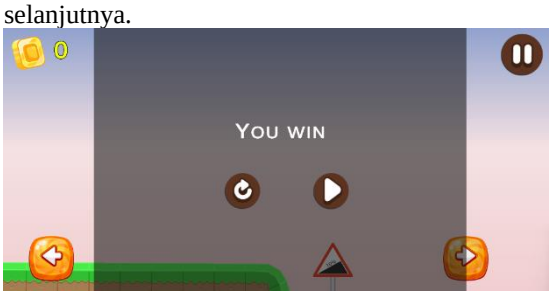
Tampilan *game over* muncul ketika *player* gagal melewati rintangan yang ada didalam game contohnya jika menerobos lampu merah, terjatuh ketika terlalu cepat melaju, dan terkena pohon yang tumbang. Pada bagian tengah terdapat tombol untuk mengulang game dari awal.



Gambar 14 Tampilan Menu Game Over

7. Implementasi Menu Winner

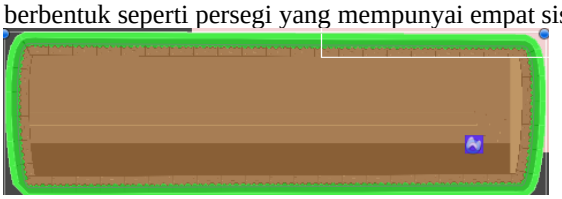
   Tampilan *winner* muncul ketika *player* berhasil menuju *finish* dan melewati semua rintangan yang ada. Pada menu *winner* terdapat tombol untuk mengulangi permainan dan melanjutkan permainan ke level selanjutnya.



Gambar 15 Tampilan Menu Winner

8. Implementasi Sebelum diterapkan Perlin Noise di Map atau Jalan

   Tampilan sebelum diterapkan perlin noise, map berbentuk seperti persegi yang mempunyai empat sisi.



Gambar 16 Tampilan Map Sebelum diterapkan Perlin Noise

9. Implementasi Sesudah diterapkan Perlin Noise di Map atau Jalan

   Tampilan sesudah diterapkan perlin noise, map atau jalan memanjang ke samping pada sisi sebelah kanan atas dan bawah dan membentuk gelombang yang berbeda-beda setiap kali game berlangsung.

Gambar 17 Tampilan Map Sesudah diterapkan Perlin Noise

Kode Program 1 Penerapan Perlin Noise ke dalam Map

```
1.  using System.Collections;
2.  using System.Collections.Generic;
3.  using UnityEngine;
4.  using UnityEngine.U2D;
5.
6.  public class TerrainCreator : MonoBehaviour
7.  {
8.      public SpriteShapeController shape;
9.      public int scale = 50;
10.     public int numofPoints = 100;
11.
12.     private void Start()
13.     {
14.         shape = GetComponent<SpriteShapeController>();
15.         float distanceBwtnpoints = (float)scale / (float)numofPoints;
16.         shape.spline.SetPosition(2, shape.spline.GetPosition(2) + Vector3.right * scale);
17.         shape.spline.SetPosition(3, shape.spline.GetPosition(3) + Vector3.right * scale);
18.
19.         for (int i = 0; i < 100; i++)
20.         {
21.             float xPos = shape.spline.GetPosition(i + 1).x + distanceBwtnpoints;
22.             shape.spline.InsertPointAt(i + 2, new Vector3(xPos, 5 * Mathf.PerlinNoise(i * Random.Range(5.0f, 15.0f), 0)));
23.         }
24.
25.         for (int i = 2; i < 102; i++)
26.         {
27.             shape.spline.SetTangentMode(i, ShapeTangentMode.Continuous);
28.             shape.spline.SetLeftTangent(i, new Vector3(-2, 0, 0));
29.             shape.spline.SetRightTangent(i, new Vector3(2, 0, 0));
30.         }
31.
32.
33.     }
34. }
```

Pada kode program 4.1 merupakan sebuah skrip dalam bahasa pemrograman C# yang digunakan dalam Unity Engine untuk membuat terrain menggunakan SpriteShapeController. Berikut adalah penjelasan kode tersebut:

Metode Start () akan dipanggil saat permainan dimulai. Di dalam metode ini, kita menginisialisasi variable shape dengan mengambil komponen SpriteShapeController dari game object ini sendiri. Kemudian kita menghitung jarak antara setiap titik dengan membagi scale dengan numofPoint dan menyimpannya dalam variabel distanceBwtnpoints. Selanjutnya, kita memindahkan posisi titik ke-2 dan ke-3 sejauh scale ke arah kanan.

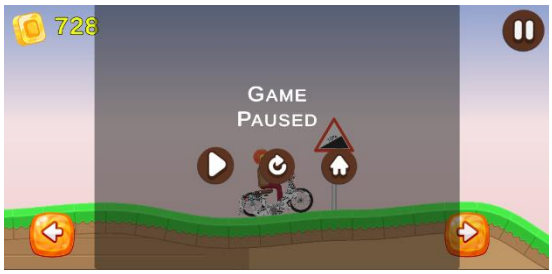
Selanjutnya melakukan pengulangan sebanyak 100 kali untuk menambahkan titik-titik baru ke dalam spline. Didalam pengulangan, kita menghitung posisi horizontal xPos untuk setiap titik baru dengan menambahkan jarak antar titik (distanceBwtnpoints) ke

posisi x titik sebelumnya. Kemudian, kita menggunakan Mathf.PerlinNoise untuk menghasilkan nilai vertical y berdasarkan perhitungan perlin noise dengan input berdasarkan nilai i * Random.Range(5.0f,15.0f).

Hasil perlin noise ini kemudian dikalikan dengan 5 agar mendapatkan variasi yang lebih besar pada tinggi terrain. Titik baru dengan posisi(xPos,y) ditambahkan ke spline menggunakan shape.spline.InsertPointAt().

10. Implementasi Menu Game Pause

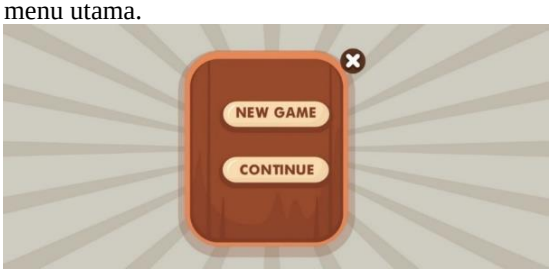
Tampilan Game Paused muncul ketika *player* mengklik tombol pause dibagian pojok kanan atas didalam game. Pada bagian game pause terdapat tiga tombol yang pertama adalah tombol play sebelah kiri untuk melanjutkan permainan. Kedua pada bagian tengah adalah tombol *restart* untuk mengulang permainan. Ketiga adalah tombol *home* sebelah kanan untuk kembali ke menu utama.



Gambar 18 Tampilan Menu Game Pause

11. Tampilan Menu Play

Pada tampilan sesudah mengklik tombol *play*, terdapat menu *new game* dan *continue*. Pada tombol *new game* digunakan untuk memulai game baru dengan memilih level permainan. Pada bagian bawah tombol new game, terdapat tombol *continue* yang digunakan untuk meneruskan atau melanjut permainan dengan memilih level yang sudah dimainkan. Tombol keluar pada bagian kanan atas digunakan untuk kembali ke menu utama.



Gambar 19 Tampilan Menu Play

12. Tampilan Menu Pilih Level Satu

Tampilan pilih level muncul setelah pengguna mengklik tombol new game, selanjutnya akan muncul *scene* memilih level satu sampai enam, level yang terbuka pertama kali adalah level satu, pengguna harus menyelesaikan level pertama untuk membuka level kedua dan seterusnya. Pada bagian kanan atas terdapat tombol keluar digunakan untuk kembali ke menu *new game* dan *continue game.*

Gambar 20 Tampilan Menu Pilih Level Satu

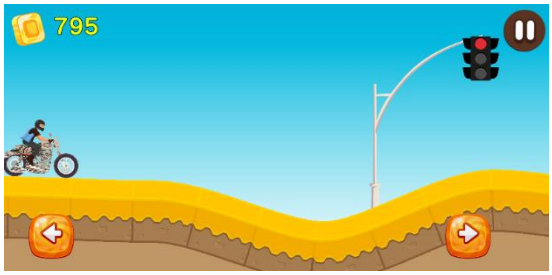
13. Implementasi Menu Stage Game

Pada menu stage game terdapat beberapa tampilan tombol-tombol untuk menjalankan karakter yang berupa kendaraan atau *vehicle.* Pada bagian kanan bawah adalah tombol gas yang digunakan menggerakan karakter atau kendaraan melaju kedepan. Pada bagian kiri bawah adalah tombol brake yang digunakan untuk menggerakan karakter atau kendaraan mundur ke belakang. *Player* diharuskan mencapai garis *finish* untuk lanjut kelevel dua. Pada bagian kiri atas adalah *icon coin* yang didapat oleh pengguna didalam game dan digunakan untuk membeli karakter di menu *shop*. Bagian kanan atas adalah tombol pause untuk berhenti sejenak. *Background stage* pada level satu bewarna biru langit cerah dan jalannya berwarna kuning setiap *stage* jalan atau map berbeda-beda. Terdapat rintangan didalam game *player* harus melewati rintangan tidak boleh tertabrak jika menabrak maka *game* berakhir

Gambar 21 Tampilan Stage Game

14. Implementasi Rintangan Lampu Lalu Lintas

Pada menu stage game level satu terdapat rintangan yaitu lampu lalu lintas, jika lampu lalu lintas berwarna merah maka *player* harus berhenti dan menunggu lampu berwarna hijau, jika *player* melanggar maka *game* akan berakhir.



Gambar 22 Tampilan Lampu Lalu Lintas

Proses untuk menerapkan perlin noise kedalam map menggunakan kode program yang sudah dibuat dengan Bahasa C# sebagai berikut:

```
1.  using System.Collections;
2.  using System.Collections.Generic;
3.  using UnityEngine;
4.
5.  public class TraficLightController : MonoBehaviour
6.  {
7.      public GameObject losserObject;
8.      private GameObject myGameObject1, myGameObject2,
    myGameObject3, myGameObject4;
9.      PlayerMovement componentPlayer;
10.     public enum LightColor
11.     {
12.         Red,
13.         Yellow,
14.         Green
15.     }
16.     public LightColor CurrentLight { get; private
    set; }
17.     public Sprite greenSprite;
18.     public Sprite yellowSprite;
19.     public Sprite redSprite;
20.
21.     private SpriteRenderer spriteRenderer;
22.
23.     private void Awake()
24.     {
25.         spriteRenderer =
    GetComponent<SpriteRenderer>();
26.     }
27.
28.     private void Start()
29.     {
30.         // Mulai animasi dengan memanggil fungsi
    ChangeLight setelah jeda waktu tertentu
31.         InvokeRepeating("ChangeLight", 0f, 3f);
32.         componentPlayer =
    FindObjectOfType<PlayerMovement>();
33.         myGameObject1 =
    GameObject.Find("Player01(Clone)");
34.         myGameObject2 =
    GameObject.Find("Player02(Clone)");
35.         myGameObject3 =
    GameObject.Find("Player03(Clone)");
36.         myGameObject4 =
    GameObject.Find("Player04(Clone)");
37.     }
38.
39.
40.     private void ChangeLight()
41.     {
42.         // Ganti Sprite berdasarkan urutan lampu
    (merah - kuning - hijau)
43.         if (spriteRenderer.sprite == redSprite)
44.         {
45.             spriteRenderer.sprite = yellowSprite;
46.         }
47.         else if (spriteRenderer.sprite ==
    greenSprite)
48.         {
49.             spriteRenderer.sprite = redSprite;
50.         }
51.         else if (spriteRenderer.sprite ==
    yellowSprite)
52.         {
53.             spriteRenderer.sprite = greenSprite;
54.         }
55.     }
56.
57.     void OnTriggerEnter2D(Collider2D other)
58.     {
59.         if (other.transform.tag == "Player")
60.         {
61.
62.             if (spriteRenderer.sprite == redSprite)
63.             {
64.                 if (myGameObject1 != null)
65.                 {
66.                     myGameObject1.SetActive(false);
67.                     losserObject.SetActive(true);
68.                     componentPlayer.traficRed =
    true;
69.                     AudioManager.instance.Play("Game
    Over");
70.                 }
71.
72.                 if (myGameObject2 != null)
73.                 {
74.                     myGameObject2.SetActive(false);
75.                     losserObject.SetActive(true);
76.                     componentPlayer.traficRed =
    true;
77.                     AudioManager.instance.Play("Game
    Over");
78.                 }
79.
80.                 if (myGameObject3 != null)
81.                 {
82.                     myGameObject3.SetActive(false);
83.                     losserObject.SetActive(true);
84.                     componentPlayer.traficRed =
    true;
85.                     AudioManager.instance.Play("Game
    Over");
86.                 }
87.
88.                 if (myGameObject4 != null)
89.                 {
90.                     myGameObject4.SetActive(false);
91.                     losserObject.SetActive(true);
92.                     componentPlayer.traficRed =
    true;
93.                     AudioManager.instance.Play("Game
    Over");
94.                 }
95.
96.             }
97.
98.         }
99.     }
100. }
```

Kode Program 2 Penerapan Lampu Lalu Lintas

Kode di atas adalah sebuah skrip dalam bahasa pemrograman C# yang digunakan dalam Unity Engine untuk mengontrol lampu lalu lintas dalam permainan. Berikut adalah penjelasan singkat dari kode tersebut:

a) Variabel losserObject adalah objek game yang akan diaktifkan saat *player* bertabrakan dengan lampu merah.
b) Variabel myG ameObject1, myGameObject2, myGameObject3 dan myGameObject4 adalah objek game *player*.
c) Variabel componentPlayer adalah komponen PlayerMovement yang digunakan untuk mengakses fungsi-fungsi pada skrip PlayerMovement.
d) enum LightColor mendefinisikan kemungkinan warna lampu lalu lintas: Merah (*Red*), Kuning (*Yellow*) dan Hijau (*Green*).
e) Properti CurrentLight merupakan properti pembaca yang memberikan nilai warna lampu saat ini.
f) greenSprite, yellowSprite, dan redSprite adalah sprite untuk masing-masing warna lampu lalu lintas.
g) spriteRenderer adalah komponen SpriteRenderer yang digunakan untuk mengubah sprite pada objek ini.

Dalam metode Start(), InvokeRepeating() digunakan untuk memanggil fungsi ChangeLight() secara berulang setiap 3 detik, yang mengubah sprite lampu lalu lintas secara berurutan.

Selain itu, dalam metode Start(), variabel componentPlayer diinisialisasi dengan menemukan komponen PlayerMovement pada game object yang ada di dalam scene.

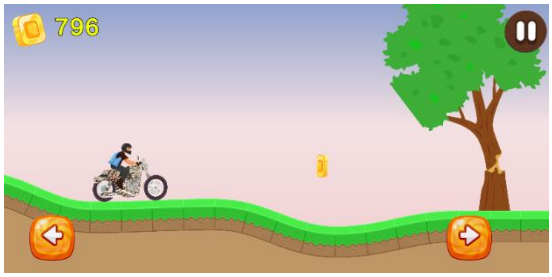
Selain itu, variabel myGameObject1, myGameObject2, myGameObject3, dan myGameObject4 diinisialisasi dengan mencari game object dengan nama yang sesuai.

Dalam metode ChangeLight(), sprite lampu lalu lintas berubah sesuai dengan urutan (merah - kuning - hijau).

Dalam metode OnTriggerEnter2D(Collider2D other), jika objek yang bersentuhan memiliki tag "Player" dan sprite lampu lalu lintas adalah merah, maka objek *player* akan dinonaktifkan, losserObject akan diaktifkan, dan suara "GameOver" akan diputar menggunakan AudioManager. Hal ini menunjukkan bahwa *player* bertabrakan dengan lampu merah dan kalah dalam permainan.

15. Implementasi Pohon Tumbang

Rintangan pada level dua sama halnya seperti pada level satu. Hal yang berbeda adalah jumlah rintangan pohon tumbang bertambah banyak.



Gambar 23 Tampilan Pohon Tumbang

```
1.  using System.Collections;
2.  using System.Collections.Generic;
3.  using UnityEngine;
4.
5.  public class tertimpa : MonoBehaviour
6.  {
7.      private GameObject myGameObject1, myGameObject2,
    myGameObject3, myGameObject4;
8.
9.      PlayerManager componentPlayer;
10.     void Start()
11.     {
12.         componentPlayer =
    FindObjectOfType<PlayerManager>();
13.         myGameObject1 =
    GameObject.Find("Player01(Clone)");
14.         myGameObject2 =
    GameObject.Find("Player02(Clone)");
15.         myGameObject3 =
    GameObject.Find("Player03(Clone)");
16.         myGameObject4 =
    GameObject.Find("Player04(Clone)");
17.     }
18.
19.     // Update is called once per frame
20.
21.     void OnTriggerEnter2D(Collider2D other)
22.     {
23.         if (other.transform.tag == "Player")
24.         {
25.             if (componentPlayer.tertimpa == false)
26.             {
27.                 if (myGameObject1 != null)
28.                 {
29.                     myGameObject1.SetActive(false);
30.                     componentPlayer.tertimpa = true;
31.                 }
32.
33.                 if (myGameObject2 != null)
34.                 {
35.                     myGameObject2.SetActive(false);
36.                     componentPlayer.tertimpa = true;
37.                 }
38.
39.                 if (myGameObject3 != null)
40.                 {
41.                     myGameObject3.SetActive(false);
42.                     componentPlayer.tertimpa = true;
43.                 }
44.
45.                 if (myGameObject4 != null)
```

```
                {
                    myGameObject4.SetActive(false);
                    componentPlayer.tertimpa = true;
                }
            }

            AudioManager.instance.Play("GameOver");

        }
    }
}
```

Kode Program 3 Pohon Tumbang

Kode di atas adalah sebuah skrip dalam bahasa pemrograman C# yang digunakan dalam Unity Engine untuk menangani tabrakan antara *player* (dengan tag "Player") dan objek tertentu. Berikut adalah penjelasan singkat dari kode tersebut:

Variabel myGameObject1, myGameObject2, myGameObject3, dan myGameObject4 adalah objek game *player*.

Variabel componentPlayer adalah komponen PlayerManager yang digunakan untuk mengakses variabel dan fungsi pada skrip PlayerManager.

Dalam metode Start(), variabel componentPlayer diinisialisasi dengan mencari komponen PlayerManager pada game object yang ada didalam scene.

Selain itu, variabel myGameObject1, myGameObject2, myGameObject3, dan myGameObject4 diinisialisasi dengan mencari game object dengan nama yang sesuai.

Dalam metode OnTriggerEnter2D (Collider2D other), jika objek yang bersentuhan memiliki tag "Player" dan componentPlayer.tertimpa adalah false, maka objek player akan dinonaktifkan dan componentPlayer.tertimpa diubah menjadi true. Hal ini menunjukkan bahwa player tertimpa oleh objek tertentu dalam permainan.

Setelah itu, AudioManager.instance.Play("GameOver") dipanggil untuk memutar suara "GameOver".

Dengan menggunakan kode diatas, kita dapat menangani tabrakan antara player dan objek tertentu dalam permainan, menonaktifkan player dan mengubah status tertimpa pada PlayerManager

B. Pembahasan

1. Pengujian

Tahap pengujian (testing) adalah tahap untuk menguji beberapa fungsi utama yang ada pada game, apakah fungsi tersebut berfungsi sesuai yang diharapkan atau tidak. Pengujian yang dilakukan pada game “Safety Riding” adalah dengan pengujian alpha dan pengujian beta.

a) Pengujian Alpha

Dalam pengujian ini penulis menguji langsung game dengan menjalankan semua fungsi game dan mecatatnya dalam tabel. Berikut tabel hasil pengujian alpha.

Table 1

Pengujian Alpha Menu Utama

Kasus Uji	Langkah Pengujian	Hasil yang diterapkan	Hasil uji
Tombol Play	Tekan tombol icon play	Menampilkan cerita yang menandakan game dimulai	Berhasil
Tombol Setting	Tekan tombol icon setting	Menampilkan setting volume	Berhasil
Tombol Shop	Tekan tombol icon shop	Menampilkan menu pembelian kendaraan	Berhasil
Tombol Keluar	Tekan tombol icon keluar	Menampilkan konfirmasi keluar ya atau tidak	Berhasil

Pada table 1 pengujian berupa tombol Play,Setting,Shop,Keluar yang ada dimenu utama game.Tombol tersebut telah diuji dan berhasil semua.

Table 2

Pengujian Menu Play

Kasus Uji	Langkah Pengujian	Hasil yang diterapkan	Hasil uji
Tombol New Game	Tekan tombol icon new game	Memulai game baru dan menampilkan menu pilih level	Berhasil
Tombol Continue	Tekan tombol icon continue	Melanjutkan game yang sudah dimainkan dan menampilkan menu pilih level	Berhasil

Pengujian berupa tombol New Game dan Continue yang ada dimenu *play game*. Tombol tersebut telah diuji fungsinya dan berhasil semua.

Table 3

Pengujian Alpha Menu Pilih Level

Kasus Uji	Langkah Pengujian	Hasil yang diterapkan	Hasil uji

Tombol Level	Tekan tombol icon level	Memulai game sesuai level yang dipilih	Berhasil
Tombol Kunci Level	Tekan tombol icon kunci level	Mengunci level yang belum diselesaikan	Berhasil
Tombol Kembali	Tekan tombol icon kembali	Kembali ke Menu play	Berhasil

Pada table 3 pengujian berupa tombol Level, Kunci Level dan Kembali yang ada dimenu Pilih Level .Tombol tersebut telah diuji fungsinya dan berhasil semua.

Table 4

Pengujian Alpha Menu Game Pause

Kasus Uji	Langkah Pengujian	Hasil yang diterapkan	Hasil uji
Tombol Play	Tekan tombol icon play	Melanjutkan game yang diapause	Berhasil
Tombol Restart	Tekan tombol icon restart	Mengulang game dari awal	Berhasil
Tombol Home	Tekan tombol icon home	Kembali ke menu utama	Berhasil

Pada table 4 pengujian berupa tombol Play, Restart, dan Home yang ada dimenu Game Paused. Tombol tersebut telah diuji berhasil semua dan sesuai dengan fungsinya.

Table 5

Pengujian Alpha Menu Winner

Kasus Uji	Langkah Pengujian	Hasil yang diterapkan	Hasil uji
Tombol Next	Tekan tombol icon play	Melanjutkan ke level selanjutnya dan menuji ke menu pilih level	Berhasil
Tombol Restart	Tekan tombol icon restart	Mengulang game dari awal	Berhasil

Pada table 5 pengujian berupa tombol Next, dan Restart yang ada dimenu Winner Tombol tersebut telah diuji berhasil semua dan sesuai dengan fungsinya.

Table 6

Pengujian Alpha Game Over

Kasus Uji	Langkah Pengujian	Hasil yang diterapkan	Hasil uji
Tombol Restart	Tekan tombol icon restart	Mengulang game dari awal	Berhasil

Pada table 6 pengujian berupa tombol Restart yang ada dimenu Game Over Tombol tersebut telah diuji berhasil semua dan sesuai dengan fungsinya.

Table 7

Pengujian Alpha Menu Setting

Kasus Uji	Langkah Pengujian	Hasil yang diterapkan	Hasil uji
Slider	Geser slider untuk mengatur besar kecilnya backsound	Mengatur volume music game	Berhasil
Tombol Kembali	Tekan tombol icon kembali	Kembali ke menu utama	Berhasil

Pada table 7 pengujian berupa slider dan tombol Kembali yang ada dimenu Setting Tombol tersebut telah diuji berhasil semua dan sesuai dengan fungsinya.

Table 8

Pengujian Alpha Menu Shop

Kasus Uji	Langkah Pengujian	Hasil yang diterapkan	Hasil uji
Tombol Arah Kanan	Tekan tombol iicon arah kanan	Bergerak maju ke kanan	Berhasil
Tombol Arah Kiri	Tekan tombol	Bergerak mundur ke kiri	Berhasil

	icon arah kiri		
Tombol Price	Tekan tombol icon price	Membeli Kendaraan	Berhasil
Tombol Back	Tekan tombol icon back	Kembali ke menu utama	Berhasil

Pada table 8 pengujian berupa tombol arah kanan, arah kiri, Price dan Back yang ada dimenu shop tombol tersebut telah diuji berhasil semua dan sesuai dengan fungsinya.

b) Pengujian Beta

Pengujian beta merupakan pengujian langsung kepada pengguna untuk mencoba game dan mengisi kuesioner mengenai game Safety Riding. Pengujian ini dilakukan secara online dengan membagikan *link google form* kepada 30 orang. Kuisioner terdiri dari 25 pertanyaan.Data yang didapat diuji secara statistik menggunakan spss. dari hasil uji tersebut dapat dilihat kesimpulan mengenai *game Safety Riding* yang telah dibuat. Berikut adalah daftar pertanyaan di kuisioner tersebut, dapat dilihat pada table 9

Table 9

Pertanyaan

No	Pertanyaan
1	Seberapa puaskan anda terhadap tampilan game racing safety riding ini?
2	Seberapa puaskan anda terhadap kecocokan tema game dan tampilannya?
3	Sebarapa puaskah anda terhadap pengalaman visual yang diberikan oleh tampilan game ini?
4	Seberapa puaskah anda terhadap kemudahan untuk mengontrol kendaraan di game racing 2D ini?
5	Seberapa puaskah anda terhadap aturan dan tujuan dalam game racing 2D ini?
6	Seberapa puaskan anda terhadap tantangan yang diberikan saat memainkan game racing 2D ini?
7	Seberapa puaskah anda terhadap fitur yang dimiliki oleh game racing 2D ini?
8	Seberapa puaskah anda terhadap efek visual di game racing 2D ini?
9	Seberapa puaskah anda terhadap efek visual di game racing 2D ini untuk memberikan pengalaman bermain yang lebih menyenangkan?
10	Seberapa puaskan anda terhadap fitur yang dimiliki game racing 2D ini untuk mempertahankan minat Anda dalam bermain?
11	Seberapa puaskah anda terhadap fitur kebebasan dalam memilih kendaraan di game racing 2D ini?
12	Seberapa puaskah anda terhadap fitur stage atau level yang berbeda-beda di dalam game racing 2D ini?
13	Bagaimana Anda menilai kualitas grafis dalam game racing 2D ini?
14	Seberapa puaskah anda terhadap kualitas suara dan efek suara dalam game racing 2D ini?
15	Seberapa puaskah anda terhadap fitur mendapatkan coin dari bermain dan digunakan untuk membeli kendaraan?
16	Seberapa puaskah anda terhadap jalan atau map dalam game racing yang tidak menentu dan selalu berubah?
17	Seberapa puaskah anda terhadap tingkat kesulitan game safety riding?
18	Seberapa puaskan anda terhadap durasi bermain yang diberikan di game racing 2D ini?
19	Seberapa puaskah anda terhadap alur cerita game safety riding?
20	Seberapa puaskah anda terhadap rambu lalu lintas untuk memperingati banyaknya tanjakan jalan?
21	Seberapa puaskah anda terhadap rintangan pohon tumbang yang ada didalam game safety riding?
22	Seberapa puaskah anda terhadap asset karakter atau kendaraan dalam game racing 2D ini?
23	Seberapa puaskah anda terhadap game racing 2D ini yang hanya dimainkan secara offline?
24	Seberapa puaskah anda terhadap Lampu lalu lintas pada game safety riding?
25	Seberapa puaskah anda terhadap keseluruhan game safety riding, mulai dari gameplay, tampilan dan alur cerita?

Pertanyaan kuisioner mempunyai indikator penilaian sebagai berikut:

Nilai	Keterangan
1	Tidak Memuaskan
2	Kurang Memuaskan
3	Cukup Memuaskan
4	Memuaskan
5	Sangat Memuaskan

Selanjutnya hasil pengisian kuisioner tersebut dihitung validitas dan reliabilitasnya menggunakan SPSS. Berikut adalah hasil dari perhitungan menggunakan SPSS:

Hasil nilai korelasi antara skor item dengan skor konstruk (*loading factor*) dari indikator-indikator yang mengukur konstruk tersebut bernilai di atas 0.5 (> 0.5). Dengan demikian dapat disimpulkan bahwa seluruh item telah memenuhi syarat validitas. Hasil nilai composite *reliability* keenam variabel penelitian bernilai di atas 0.7. Dengan demikian dapat disimpulkan bahwa instrumen telah memenuhi persyaratan validitas dan reliabilitas.[3]

Scale: ALL VARIABLES

Case Processing Summary

		N	%
Cases	Valid	30	100.0
	Excluded[a]	0	.0
	Total	30	100.0

a. Listwise deletion based on all variables in the procedure.

Reliability Statistics

Cronbach's Alpha	N of Items
.979	25

Gambar 24 Reliabillity

Item-Total Statistics

	Scale Mean if Item Deleted	Scale Variance if Item Deleted	Corrected Item-Total Correlation	Cronbach's Alpha if Item Deleted
Butir_1	91.03	252.654	.828	.978
Butir_2	91.03	254.378	.761	.979
Butir_3	91.00	254.690	.757	.979
Butir_4	90.97	255.413	.783	.979
Butir_5	90.77	253.151	.889	.978
Butir_6	91.10	253.610	.820	.978
Butir_7	91.03	254.171	.812	.978
Butir_8	90.90	253.128	.858	.978
Butir_9	90.93	253.651	.822	.978
Butir_10	90.90	252.921	.867	.978
Butir_11	90.93	253.030	.847	.978
Butir_12	90.90	254.783	.841	.978
Butir_13	91.07	251.306	.794	.979
Butir_14	90.87	252.602	.899	.978
Butir_15	91.00	257.448	.562	.980
Butir_16	91.00	253.586	.801	.979
Butir_17	90.87	259.016	.631	.980
Butir_18	90.87	256.464	.737	.979
Butir_19	90.97	255.137	.794	.979
Butir_20	91.03	255.826	.792	.979
Butir_21	90.83	254.557	.836	.978
Butir_22	90.83	251.109	.880	.978
Butir_23	90.83	251.454	.866	.978
Butir_24	90.97	252.861	.798	.979
Butir_25	90.97	253.137	.830	.978

Gambar 25 Validitas

Nilai dari *corrected item correlation* semuanya nilai diatas 0,5.

Table 10

Frekuensi hasil kuisioner

Butir	Jawaban										Total	
	Tidak Memuaskan		Kurang Memuaskan		Cukup Memuaskan		Memuaskan		Sangat Memuaskan			
	n	%	n	%	n	%	n	%	n	%	n	%
1	-	-	-	-	14	46.7	9	30	7	23.3	30	100
2	-	-	-	-	14	46.7	8	26.7	8	26.7	30	100
3	-	-	-	-	12	40	10	33.3	8	26.7	30	100
4	-	-	-	-	12	40	10	33.3	8	26.7	30	100
5	1	3.3	1	3.3	6	20	14	46.7	8	26.7	30	100
6	-	-	1	3.3	14	46.7	9	30	6	20	30	100
7	1	3.3	-	-	13	43.3	9	30	7	23.3	30	100
8	1	3.3	-	-	10	33.3	11	36.7	8	26.7	30	100
9	-	-	1	3.3	11	36.7	11	36.7	7	23.3	30	100
10	-	-	-	-	9	30	12	40	9	30	30	100
11	-	-	-	-	11	36.7	10	33.3	9	30	30	100
12	-	-	-	-	8	26.7	15	50	7	23.3	30	100
13	1	3.3	-	-	11	36.7	11	36.7	7	23.3	30	100
14	-	-	-	-	9	30	12	40	9	30	30	100
15	1	3.3	2	6.7	9	30	10	33.3	8	26.7	30	100
16	1	3.3	1	3.3	13	43.3	8	26.7	7	23.7	30	100
17	1	3.3	1	3.3	8	26.7	13	43.3	7	23.3	30	100
18	1	3.3	1	3.3	8	26.7	12	40	8	26.7	30	100
19	-	-	1	3.3	10	33.3	12	40	7	23.3	30	100
20	-	-	-	-	11	36.7	13	43.3	6	20	30	100
21	-	-	-	-	8	26.7	12	40	10	33.3	30	100
22	-	-	-	-	9	30	12	40	9	30	30	100
23	-	-	1	3.3	10	33.3	9	30	10	33.3	30	100
24	-	-	-	-	13	43.3	8	26.7	9	30	30	100
25	-	-	-	-	11	36.7	10	33.3	9	30	30	100

Dari hasil pengisian kuisioner manfaatnya adalah rintangan yang ada digame safety riding menerapkan aturan umum untuk berkendara, mulai dari tidak melanggar lampu lalu lintas, menaati rambu lalu lintas, tampilan game safety riding yang menarik, map atau jalan selalu berubah setiap kali bermain. Kekurangannya ada minimnya fitur yang dimiliki game safety riding.

IV. PENUTUP

A. Kesimpulan

1. Penerapan perlin noise pada source code dan sisi map yang *digenerate* secara otomatis sehingga map mempunyai variasi ketinggian yang berbeda-beda.

2. Penerapan alogaritma perlin noise kedalam map disetiap level menghasilkan map yang berbeda-beda setiap akan memulai game.
3. Penerapan unsur Safety Riding ke dalam game racing dengan menambahkan rambu tanjakan agar pemain berhati-hati, menaati aturan lampu lintas, menggunakan helm, jaket, dan sepatu saat berkendara

B. Saran

Karena keterbatasan game yang dibuat masih 2 dimensi sehingga menyebabkan penerapan unsur dari safety riding kurang, dan map pada game selalu berubah menyebabkan beberapa asset game didalamnya tidak tepat dalam peletakannya.

REFERENSI

[1] World Health Organization., *Global status report on road safety : time for action.* World Health Organization, 2013.

[2] Badan Intelijen Negara Republik Indonesia, "Kecelakaan Lalu Lintas menjadi Pembunuh Terbesar Ketiga," 2014. <http://www.bin.go.id/awas/detil/197/4/21/03/2013/kecelakaan-lalulintas-menjadi-pembunuh-terbesar-ketiga> (accessed Jan. 10, 2023).

[3] R. A. T. Kala'tiku, Arifuddin, and Syamsuddin, "BRAINSTORMING SEBAGAI PEMODERASIPENGARUHPENGALAMAN, PELATIHAN,SKEPTISISME PROFESIONAL DAN INTEGRITAS TERHADAP KEMAMPUAN MENDETEKSIKECURANGAN," vol. 7, pp. 81–90, 2018.