

## BAB 3

### METODOLOGI PENELITIAN

#### 3.1 Metodologi Penelitian

Tahapan-tahapan penelitian untuk merancang *game genre* balap adalah sebagai berikut.

1. Studi Literatur

Pada tahapan studi literatur dilakukan pendalaman terhadap teori-teori yang berhubungan dengan permainan video *genre* balap yang dirancang dan dibangun, yaitu PCG, *Voronoi diagram*, GDD, *linearity*, dan *leniency*.

2. Perancangan *Game*

Pada perancangan desain terhadap *game* dibuat. Pada tahap ini dibuat GDD dan *flowchart* untuk *game* yang dibangun.

3. Pembangunan *Game*

*Game* dibuat pada tahap ini, *game* dikembangkan pada Unity C# dengan sistem operasi Windows berdasarkan tahap perancangan dan mengimplementasi metode PCG yang digunakan *Voronoi diagram*

4. Pengujian *Game*

Tahap ini dilakukan untuk menguji jika *game* dapat dimainkan dengan lancar dan pembangkit lintasan tidak mempunyai *bug* dengan *playtesting* dan *debugging*. *Playtesting* dimulai dengan membuka atau menjalankan *game* dari awal hingga memainkan sebuah lintasan sampai selesai untuk meyakinkan *game* dapat dimainkan dan lintasan yang terbuat dengan

*Voronoi diagram* dapat diselesaikan tanpa *bug*. Pengujian dilakukan menggunakan laptop dengan sistem operasi Windows 10.

## 5. Evaluasi

Evaluasi dilakukan dengan mengukur nilai *linearity* dan *leniency* pada sebuah lintasan berdasarkan banyaknya jalan lurus dan tikungan yang ada. Lintasan dibuat berkali-kali untuk diambil nilai *linearity* dan *leniency* untuk mencari tahu apakah lintasan-lintasan yang dibuat berbeda-beda. Untuk mempermudah mengambil angka tersebut, sebuah fitur pada *game* dibuat untuk mendata informasi tersebut.

## 3.2 Perancangan Aplikasi

Perancangan dari aplikasi *game* yang dibangun dijelaskan pada bagian-bagian berikut.

### 3.2.1 Struktur Permainan

Berdasarkan struktur *game* yang didefinisikan oleh Fullerton (2018), *game* dibangun berdasarkan *formal elements* dan *dramatic elements* yang dijelaskan pada studi literatur. Untuk *formal elements* terdiri dari berikut.

#### 1. *Players*

Pemain bermain sendiri (*single player*).

#### 2. *Objectives*

Melintasi garis *finish* secepat mungkin.

3. *Procedures*

- a. Saat *game* dibuka, *scene main menu* ditampilkan, untuk memulai permainan, pemain memilih opsi *Race* yang membawa pemain ke *scene* pemilihan lintasan.
- b. Pada *scene* pemilihan lintasan, pemain dapat memilih sebuah lintasan yang sudah dibangkitkan oleh *Voronoi diagram*, untuk mulai bermain, pemain memilih opsi *Choose Track* yang menampilkan *scene game* balap tersebut.
- c. Pada permainan, pemain mengontrol sebuah mobil, pemain ditugaskan untuk melakukan beberapa putaran pada lintasan untuk menyelesaikan permainan secepat mungkin.
- d. Saat pemain menyelesaikan beberapa putaran tersebut dan menyentuh garis *finish*, permainan selesai dan pemain kembali ke *main menu*.

4. *Rules*

- a. *Game* dimainkan menggunakan *keyboard*
- b. Pemain harus menyelesaikan lintasan sebanyak beberapa putaran yang ditentukan, permainan dinyatakan selesai apabila jumlah putaran sesuai dengan yang diminta oleh permainan.
- c. Sebuah putaran dinyatakan terpenuhi apabila pemain telah menelusuri lintasan dari awal hingga menyentuh garis *finish*.

5. *Resources*

- a. Lintasan-lintasan hasil *Voronoi diagram*
- b. Kecepatan dari kendaraan
- c. Waktu yang menghitung berapa lama lintasan tersebut diselesaikan.

6. *Conflict*

Menyelesaikan lintasan yang berbeda-beda secepat mungkin selagi mengikuti *checkpoint* yang mengikuti lintasan.

7. *Boundaries*

- a. Bentuk lintasan yang dipilih oleh pemain berdasarkan hasil lintasan-lintasan dari *Voronoi diagram*
- b. Bagian luar lintasan yang dapat dijalani oleh pemain tetapi dibatasi oleh penghalang.

8. *Outcome*

Pemain menyelesaikan lintasan setelah menyentuh garis *finish* dan menyelesaikan beberapa putaran.

Selanjutnya adalah *dramatic elements* yang ada dalam *game*.

1. *Challenge*

Lintasan yang bervariasi yang dapat dipilih dengan tingkat kesulitan yang bervariasi sesuai keinginan pemain.

2. *Play*

*Rule-based play* dimana permainan memiliki beberapa limitasi dan aturan yang perlu diikuti untuk mencapai *objectives*.

3. *Characters*

Mobil yang dikendalikan oleh pemain.

4. *Premise*

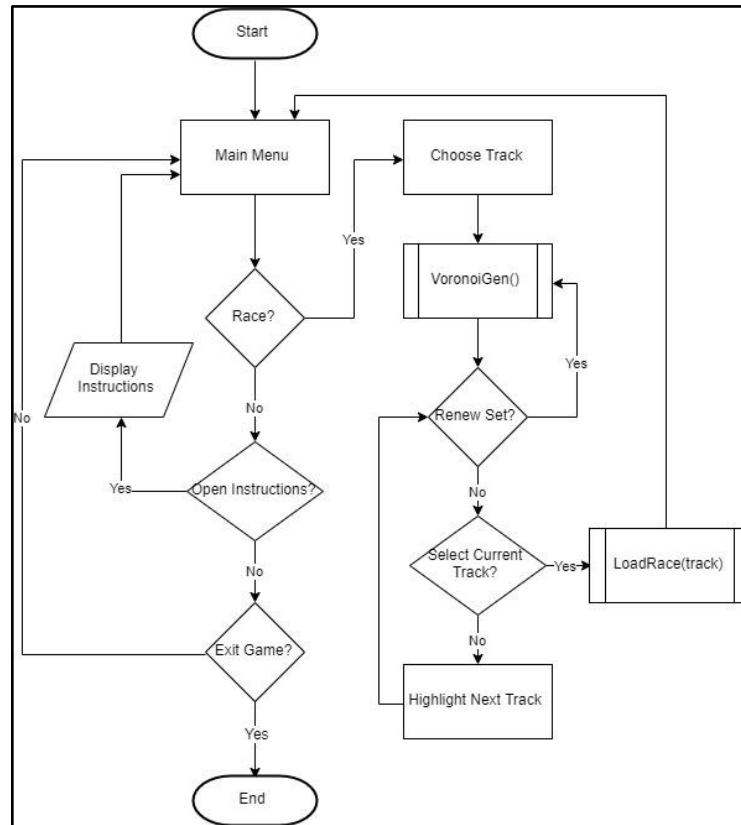
Simulasi balapan di dunia digital.

5. *Story*

Pemain mengendalikan mobil yang diuji pada dunia simulasi.

### 3.2.2 Flowchart

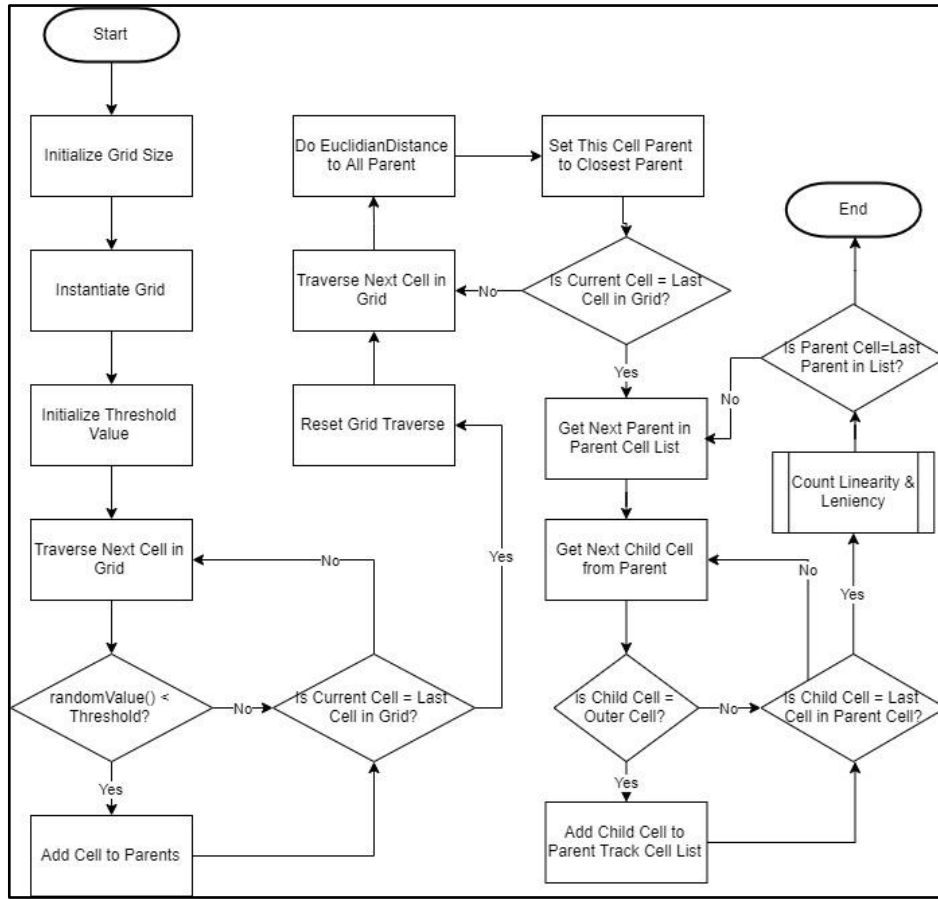
Berikut adalah *flowchart* yang digunakan sebagai panduan untuk merancang aplikasi yang dimulai dengan Gambar 3.1.



Gambar 3.1 Flowchart Utama

Pada Gambar 3.1, alur dari aplikasi *game* yang dibangun dijelaskan

1. Saat aplikasi *game* dimulai, *scene main menu* ditampilkan pertama kali dan mempunyai pilihan untuk mulai bermain atau instruksi bermain. Jika instruksi bermain dipilih, *game* menampilkan kontrol dan cara bermain.
2. Jika opsi “*Race*” dipilih, *game* lalu membangkitkan lintasan-lintasan dengan fungsi *Voronoi diagram* yang dijelaskan selanjutnya. Pemain dapat memilih salah satu lintasan tersebut untuk dimainkan.

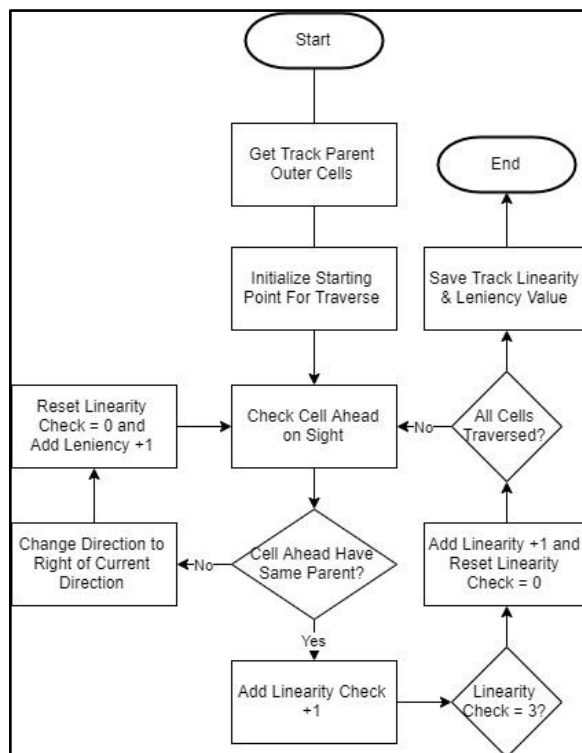


Gambar 3.2 Flowchart Voronoi Diagram Generator

Gambar 3.2 adalah alur dari bagaimana *Voronoi diagram* bekerja untuk membuat lintasan.

1. Ketika fungsi *Voronoi diagram* dipanggil, pertama *grid* dua dimensi dibuat oleh pembangkit, lalu, untuk setiap sel yang ada pada *grid* dua dimensi ini, pembangkit memberikan titik-titik awal pada sel tertentu sebagai *parent* dari sel-sel lain dari *Voronoi diagram* nantinya berdasarkan angka acak yang diberikan.
2. Ketika semua titik *parent* telah dibuat, lakukan perhitungan *Euclidean distance* untuk setiap sel pada *grid* yang bukan *parent*. Lalu jadikan *child* dari *parent* dengan jarak terdekat.

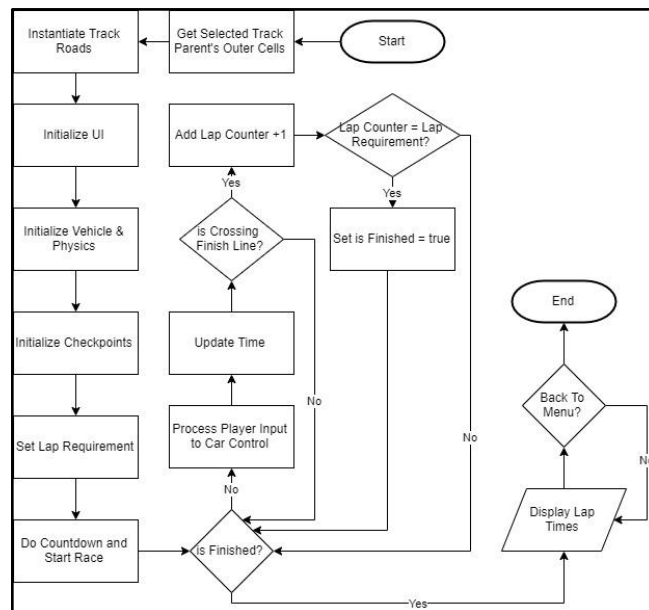
3. Untuk setiap sel *parent* yang ada, cek apabila setiap sel *child* pada *parent* adalah sel terluar dari *region* hasil *Voronoi diagram*, sel terluar digunakan untuk membuat lintasan tersebut. Jika sel merupakan sel terluar, simpan sel untuk digunakan oleh pembangkit sebagai lintasan pada tahap selanjutnya.
4. Setelah seluruh sel telah diperiksa, lakukan perhitungan untuk *linearity* dan *leniency* untuk analisis dan evaluasi *Voronoi diagram* sebagai metode PCG yang dijelaskan selanjutnya.



Gambar 3.3 Flowchart Perhitungan Linearity dan Leniency

1. Perhitungan *linearity* dan *leniency* dimulai dengan mengambil daftar dari sel-sel terluar dari sebuah *parent* atau daerah. Setelah memilih tempat awal untuk melintasi seluruh sel tersebut, maju ke sebuah arah, dan terus maju selama sel tersebut masih merupakan sel dari *parent* atau area yang sama.

2. Setiap kali langkah maju dilakukan, lakukan penjumlahan terhadap variabel *linearity check*. Jika *linearity check* mencapai tiga, maka lakukan penjumlahan sebanyak satu terhadap *linearity* dari lintasan ini dan ubah *linearity check* menjadi nol kembali.
3. Jika sel didepan dari sel yang sedang dilintasi bukan merupakan bagian dari daerah *parent*, maka ubah arah menuju kanan dari arah sekarang. Lakukan hingga sel yang belum dilintasi dan merupakan bagian dari daerah *parent* ditemukan, setelah itu tambahkan satu terhadap *leniency* dan ubah *linearity check* menjadi nol kembali.
4. Setelah semua dari bagian lintasan diperiksa, simpan angka *linearity* dan *leniency* tersebut.



Gambar 3.4 Flowchart Load Scene Race

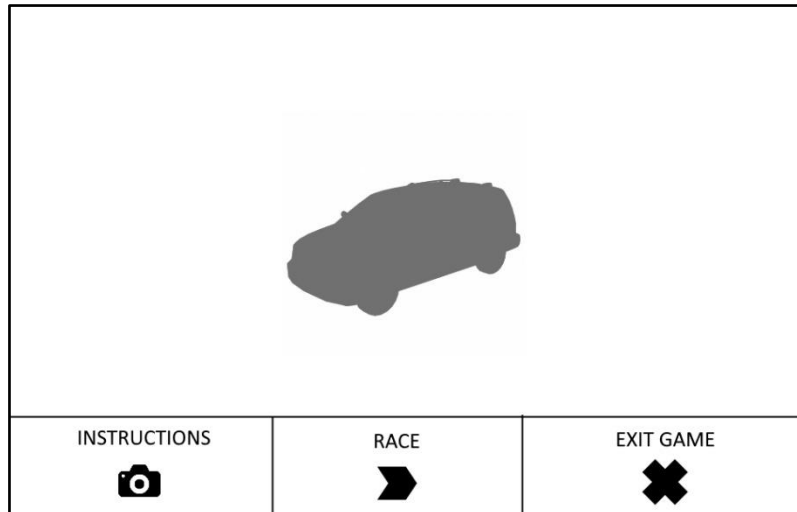


Pada Gambar 3.4, alur dari *scene* permainan balapan dijelaskan.

1. Saat *scene* dimulai, ambil bagian lintasan terluar yang sudah ditentukan sebelumnya. Lalu buat objek jalan lintasan berdasarkan masing-masing titik sel terluar yang menghubungkan masing-masing titik yang bersebelahan.
2. Inisialisasi antarmuka dan hal-hal yang berhubungan dengannya seperti perhitungan waktu, lalu inisialisasi mobil yang dimainkan oleh pemain, dan *checkpoint* untuk meyakinkan pemain tidak memotong jalan untuk mencapai garis *finish*. Banyak putaran yang harus dilalui juga diinisialisasi sebelum permainan dimulai.
3. Saat permainan dimulai, periksa apabila pemain sudah *finish*, jika belum, proses input terhadap kontrol mobil dan terus hitung waktu putaran. Jika melintasi garis *finish*, tambahkan angka putaran yang dilalui, dan apabila putaran yang dilalui sudah mencapai putaran yang harus dilalui, maka permainan selesai, tampilkan waktu dari putaran pemain.
4. Kembali ke *scene main menu* setelah pemain memilihnya.

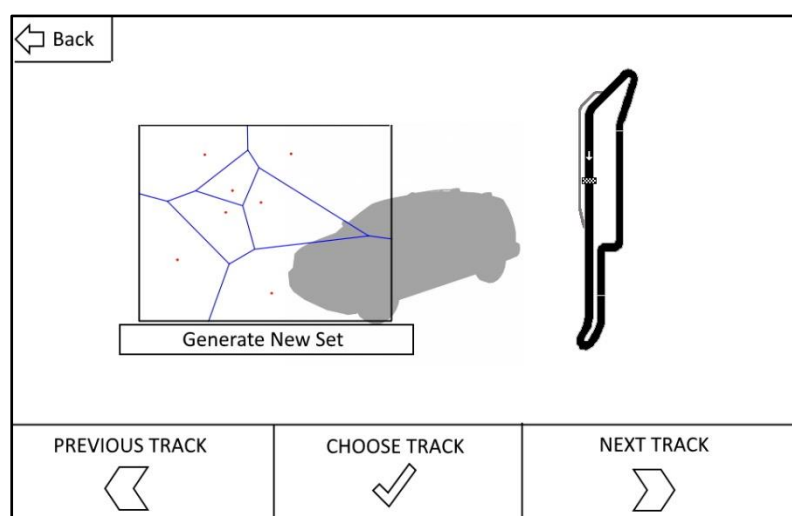
### **3.2.3 Rancangan Antarmuka**

Rancangan antarmuka dari *game* yang dibuat terdiri atas menu utama, pemilihan lintasan, dan tampilan dari permainan pada *scene gameplay*. Pada menu utama yang berada pada Gambar 3.5, terdiri dari tombol instruksi, tombol keluar, dan tombol balap untuk bermain.



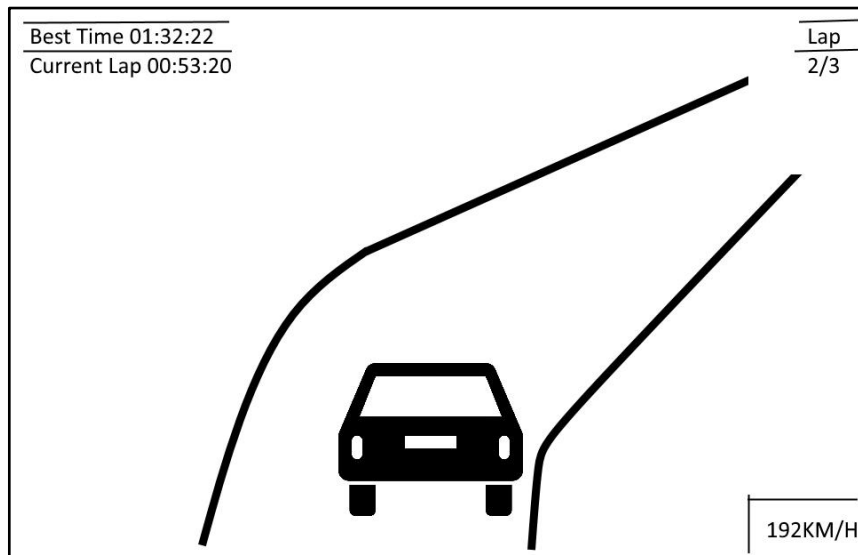
Gambar 3.5 Tampilan Menu Utama

Ketika tombol balap ditekan, *game* menampilkan kumpulan lintasan-lintasan berdasarkan hasil *Voronoi diagram*, dan di sebelah adalah lintasan yang sedang dipilih. Terdapat tombol untuk memilih lintasan sebelum atau setelahnya, juga tombol untuk membuat kumpulan lintasan yang baru dengan *Voronoi diagram*. Terakhir adalah tombol untuk memainkan lintasan yang dipilih, rancangan antarmuka untuk bagian ini terdapat pada Gambar 3.6.



Gambar 3.6 Tampilan Pemilihan Lintasan


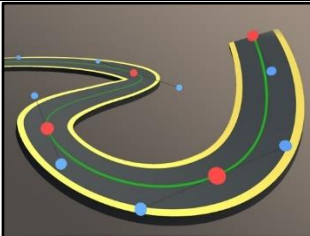
Ketika *scene gameplay* dimainkan, pemain ditampilkan model mobil pemain, dan beberapa antarmuka seperti waktu putaran, kecepatan dari kendaraan, dan banyaknya putaran yang sudah dilalui pemain seperti pada Gambar 3.7.








Gambar 3.7 Tampilan Permainan

### 3.2.4 Daftar Aset

Tabel 3.1 Daftar Aset

Gambar	Deskripsi	Sumber
	Aset mobil yang digunakan dalam <i>game</i> sebagai mobil player.	Pixtim, 2016 (Unity Asset Store)
	Bezier Path Creator untuk membuat jalan lintasan dari titik <i>Voronoi diagram</i> .	Lague, 2019 (Unity Asset Store)

Tabel 3.1 Daftar Aset (Lanjutan)

Gambar	Deskripsi	Sumber
	Aset skybox untuk mengubah langit.	Weaver, 2018 (Unity Asset Store)
	Efek suara ban.	Koenig, 2018 (Soundbible)
	Efek suara <i>boost</i> .	Insanity2Fevr, 2015 (Youtube)
	Musik menu utama.	TeknoAXE, 2018 (teknoaxe.com)
	Musik <i>gameplay</i> .	TeknoAXE, 2016 (teknoaxe.com)