UNIVERSITAS GUNADARMA

FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI

IMPLEMENTASI ALGORITMA *COLLISION* PADA *GAME* "*RUSH HOURS*" BERBASIS *WEBSITE* MENGGUNAKAN JAVASCRIPT DAN HTML

Disusun Oleh:

Nama : Ahmad Bayhaki

NPM : 16119960

Jurusan : Sistem Informasi

Pembimbing : Dr. Fenni Agustina, SKom., MMSI., M.I.Kom.

Diajukan Guna Melengkapi Sebagian Syarat

Dalam Mencapai Gelar Sarjana Strata Satu (S1)

JAKARTA
2023

PERNYATAAN ORIGINALITAS DAN PUBLIKASI

Saya yang bertanda tangan dibawah ini,

Nama : Ahmad Bayhaki

NPM : 16119960

Judul Skripsi : IMPLEMENTASI ALGORITMA *COLLISION* PADA *GAME* "*RUSH HOURS*" BERBASIS *WEBSITE* MENGGUNAKAN JAVASCRIPT DAN HTML

Tanggal Sidang :

Tinggal Lulus :

Menyatakan bahwa tulisan ini adalah merupakan hasil karya saya sendiri dan dapat dipublikasikan sepenuhnya oleh Universitas Gunadarma, Segala kutipan dalam bentuk apapun telah mengikuti kaidah, yang berlaku. Mengenai isi dan tulisan adalah merupakan tanggung jawab Penulis, bukan Universitas Gunadarma.

Demikian pernyataan ini dibuat dengan sebenar-benarnya dan dengan penuh kesadaran.

Bekasi, 25 Agustus 2023.

Ahmad Bayhaki

LEMBAR PENGESAHAN

KOMISI PEMBIMBING

NO	NAMA	KEDUDUKAN
1.	Dr. Fenni Agustina, SKom., MMSI., M.I.Kom.	Ketua
2.	Nama Penguji 1	Anggota
3.	Nama Penguji 2	Anggota

Tanggal Sidang

PANITIA UJIAN

NO	NAMA	KEDUDUKAN
1.	Dr. Ravi Ahmad Salim	Ketua
2.	Prof. Dr. Wahyudi Priyono	Sekretaris
3.	Dr. Fenni Agustina, SKom., MMSI., M.I.Kom.	Anggota
4.	Nama Penguji 1	Anggota
5.	Nama Penguji 2	Anggota

Tanggal Lulus :

Mengetahui,

Pembimbing **Kepala Bagian Sidang Ujian**

(Dr. Fenni Agustina, SKom., MMSI., M.I.Kom.) **(Dr. Edi Sukirman, SSi, MM., M.I.Kom.)**

ABSTRAK

Ahmad Bayhaki. 16119960.

IMPLEMENTASI ALGORITMA COLLISION PADA *GAME* "*RUSH HOURS*" BERBASIS *WEBSITE* MENGGUNAKAN JAVASCRIPT DAN HTML

Skripsi. Jurusan Sistem Informasi, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Gunadarma, 2023.

Kata Kunci: *Game, HTML, JavaScript, Website*

(xv + 68 + Lampiran)

Bermain game telah menjadi salah satu cara yang populer untuk mengatasi kebosanan dan kepenatan. Namun, banyak dari *game* yang ada saat ini hanya menawarkan hiburan semata dan kurang memberikan nilai lebih. Untuk mengatasi hal ini, penulis telah merancang sebuah *game* inovatif yang tidak hanya menghibur, tetapi juga memberikan manfaat tambahan berupa pelatihan fokus kepada pemainnya. Dalam mengembangkan *game* berbasis *web* ini, penulis menggunakan bahasa pemrograman JavaScript dan HTML sebagai kerangka utama, dengan dukungan penuh dari perangkat lunak Visual Studio Code.

Tujuan utama dari penelitian ini adalah merancang dan mengembangkan sebuah *game* berbasis *web* yang tidak hanya menghibur, tetapi juga memiliki potensi untuk melatih konsentrasi pemainnya. Metode penelitian yang diadopsi terdiri dari beberapa langkah, termasuk pengumpulan data terkait kebutuhan pengguna dalam sebuah *game*, perancangan keseluruhan konsep *game*, pembuatan *game* itu sendiri, serta tahap uji coba untuk mengevaluasi kinerja dan efektivitasnya.

Hasil dari uji coba yang dilakukan menunjukkan bahwa *game* yang telah dikembangkan berjalan dengan baik dan sesuai dengan harapan. Melalui tautan https://rushhours.000webhostapp.com/, *game* dengan nama *"Rush Hours"* dapat diakses dengan mudah.

Pada akhirnya, penelitian ini menegaskan bahwa *game* berbasis web memiliki potensi yang besar. Dalam hal ini, penggunaan HTML dan JavaScript sebagai fondasi pengembangan *game* telah terbukti efektif. Sebagai kesimpulan, *"Rush Hours"* adalah contoh nyata bagaimana pengembangan *game* dapat menghasilkan produk yang berhasil dibuat dengan bahasa pemrograman tersebut.

Daftar Pustaka (2014 - 2023).

ABSTRACT

Ahmad Bayhaki. 16119960.

IMPLEMENTATION OF COLLISION ALGORITHM IN THE "RUSH HOURS" GAME BASED ON A WEBSITE USING JAVASCRIPT AND HTML

Thesis. Department of Information Systems, Faculty of Computer Science and Information Technology, Universitas Gunadarma, 2023.

Keywords: Game, HTML, JavaScript, Website

(xv + 68 + Appendixes)

Playing games has become a popular way to alleviate boredom and fatigue. However, many of the existing games nowadays only offer entertainment without providing much additional value. To address this issue, the author has designed an innovative game that not only entertains but also provides an added benefit of training players' focus. In developing this web-based game, the author utilized JavaScript and HTML programming languages as the main framework, with full support from Visual Studio Code software.

The primary objective of this research is to design and develop a web-based game that not only entertains but also has the potential to enhance players' concentration. The adopted research methodology consists of several steps, including collecting data related to user needs in a game, designing the overall game concept, creating the game itself, and conducting testing to evaluate its performance and effectiveness.

The results of the conducted tests indicate that the developed game performs well and aligns with expectations. Through the link https://rushhours.000webhostapp.com/, the game named "Rush Hours" can be easily accessed.

In the end, this research reaffirms the substantial potential of web-based games. In this context, the use of HTML and JavaScript as the foundation for game development has proven to be effective. In conclusion, "Rush Hours" stands as a tangible example of how game development can yield successful products using these programming languages.

Bibliography (2014 – 2023).

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh.

Segala puji dan syukur ke khadirat Allah SWT yang telah memberikan berkat, anugrah dan karunia yang melimpah, sehingga penulis dapat menyelesaikan Tugas Akhir ini.

Tugas Akhir ini disusun guna melengkapi syarat dalam mencapai gelar Sarjana Strata Satu pada jurusan Sistem Informasi, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Gunadarma. Adapun judul Tugas Akhir ini adalah IMPLEMENTASI ALGORITMA *COLLISION* PADA *GAME "RUSH HOURS"* BERBASIS *WEBSITE* MENGGUNAKAN JAVASCRIPT DAN HTML.

Walaupun banyak kesulitan yang penulis hadapi ketika menyusun Tugas Akhir ini, namun berkat bantuan dan dorongan dari berbagai pihak Tugas Akhir ini dapat diselesaikan dengan baik. Untuk itu penulis mengucapkan terima kasih, kepada:

1. Ibu Prof. E.S. Margianti, S.E, MM., selaku Rektor Universitas Gunadarma.
2. Bapak Prof. Dr. Rer. Nat. A. Benny Mutiara, selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Gunadarma.
3. Bapak Dr. Setia Wirawan, S.Kom., M.M.S.I, selaku Ketua Jurusan Sistem Informasi, Universitas Gunadarma.
4. Bapak Dr. Edi Sukirman, S.Si., MM., M.I.Kom., selaku Kepala Bagian Sidang Ujian Universitas Gunadarma.
5. Ibu Dr. Fenni Agustina, S.Kom., MMSI., M.I.Kom., selaku Dosen Pembimbing yang telah banyak membimbing dan memberikan saran dari awal sampai selesainya Tugas Akhir ini.
6. Kedua Orang tua penulis, Sahabudin dan Ariyanti yang selalu memberikan doa dan semangat sampai selesainya Tugas Akhir ini. Dalam perjalanan panjang ini, tak henti penulis merasa diberkahi oleh kehadiran dan dukungan tak tergantikan dari orang tua tercinta. Dengan penuh cinta dan pengorbanan, telah menjadi tiang penopang dalam setiap langkah perjalanan ini. Tugas Akhir ini bukanlah hanya hasil dari usaha dan dedikasi penulis semata, tetapi juga merupakan buah

dari kasih sayang, doa, dan dorongan yang tanpa henti dari orang tua. Terima kasih atas segala dukungan, bimbingan, dan inspirasi yang telah diberikan.

7. Nadya Alifia Fawza yang selalu memberikan dukungan, doa, dan memberikan semangat dan inspirasi dalam setiap langkah hidup ini. Penulis ingin menyampaikan penghargaan yang tak terhingga. Terima kasih telah memberikan warna istimewa dalam setiap momen, bahkan dalam tantangan yang penuh liku, sehingga penulis dapat menyelesaikan Tugas Akhir ini dengan baik
8. Kakak dan adik penulis, Stevany Purnamasari dan Ghoffar Indra Fata, terima kasih atas segala doa dan dukungan.
9. Shiroyasa, Kuroneko dan Kiko, selaku kucing peliharaan yang selalu menemani dan selalu menjadi penghibur dalam mengerjakan Tugas Akhir ini.
10. Semua pihak yang tidak dapat disebutkan satu persatu, yang telah tulus ikhlas memberikan doa dan motivasi, sehingga dapat terselesaikan Tugas Akhir ini.

Akhir kata, hanya kepada Allah SWT jualah segalanya dikembalikan dan penulis menyadari bahwa Tugas Akhir ini masih jauh dari sempurna, disebabkan berbagai keterbatasan dan pengetahuan yang penulis miliki. Untuk itu penulis mengharapkan keritik dan saran yang bersifat membangun untuk menjadi perbaikan dan meningkatkan hasil yang lebih baik lagi dimasa yang akan datang.

Wassalamu'alaikum Wr. Wb.

Bekasi, 25 Agustus 2023.

Ahmad Bayhaki

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Game merupakan jenis hiburan yang disukai oleh semua orang dari usia anak-anak, dewasa maupun tua. Selain digunakan untuk menghilangkan kepenatan dalam beraktivitas, sebuah *game* juga dapat berfungsi untuk melatih pola pikir seseorang untuk mencari solusi memecahkan suatu permasalahan yang ada di sebuah *game* (Singh, Sharma, & Talwar, 2012). Dahulu *game* dimainkan secara tradisional, seperti permainan kartu, catur, ular tangga, petak umpet, dan lainnya. Seiring dengan berkembangnya teknologi, permainan tersebut dikembangkan ke dalam teknologi yang lebih modern. Sekarang banyak *game* baru yang memanfaatkan teknologi modern dalam pembuatannya dan penggunaannya.

Perkembangan *game* begitu pesat dengan jenis yang beragam, mulai dari *game* strategi, *adventure, arcade, puzzle, dan sport* yang dikemas dalam *playstation game*, *game* konsol, ataupun *mini* saat ini sangat digemari baik oleh anak-anak maupun dewasa muda (15-30 tahun) yang sudah mengikuti sekuel (biasanya terdapat pada permainan konsol) dari sebuah *game* yang pernah dimainkan dari masa kecil.

Perkembangan teknologi dan informasi yang semakin pesat memberikan pengaruh yang kuat pada berbagai bidang kehidupan, salah satunya adalah bidang hiburan, yaitu *game*. Dalam bidang hiburan berupa *game* ini komputer merupakan alat yang sudah tidak asing lagi untuk digunakan dalam sarana mencari hiburan.

Perkembangan ini didukung oleh industri *hardware* (perangkat keras) dan *software* (perangkat lunak) yang tidak lepas dari semakin tingginya kebutuhan masyarakat terhadap pelayanan yang berhubungan dengan internet.

Game berasal dari kata bahasa Inggris yang memiliki arti dasar permainan. Permainan adalah sesuatu yang dapat dimainkan dengan aturan tertentu, sehingga ada pihak yang menang dan ada pihak yang kalah, biasanya *game* dilakukan dengan tidak serius atau dengan tujuan menghibur. Menurut Katie Salen dan Eric

Zimmerman (2003), sebuah permainan adalah sebuah sistem di mana pemain terlibat dalam konflik buatan, ditentukan oleh aturan, yang menghasilkan hasil yang terukur.

Seiring dengan berjalannya waktu, muncul beberapa jenis permainan yang membutuhkan kecepatan, akurasi, dan reflek pengguna agar bisa bereaksi lebih cepat akan sesuatu. Sebagai contohnya permainan "*Subway Surf*" yang terkenal karena melatih kecepatan dan akurasi pengguna, "*Point Blank*" yang membutuhkan konsentrasi penuh dalam setiap permainan, dan masih banyak lagi *game* lainnya.

Game *"Rush Hours"* dipilih sebagai alat pelatihan kognitif, karena meningkatnya kebutuhan untuk melatih kemampuan kognitif di era teknologi yang berkembang pesat dan tuntutan kehidupan yang kompleks. Dalam *game* ini, pemain dihadapkan pada pemecahan masalah yang menarik, di mana mereka harus menggunakan pemikiran strategis untuk mencapai solusi.

Game *"Rush Hours"* menyajikan pemecahan masalah dalam konteks permainan yang menyenangkan, *game* ini memberikan pengalaman menarik dan mendalam bagi para pemainnya. Selain itu, *"Rush Hours"* juga menempatkan fokus pada waktu dan kecepatan, di mana pemain harus berpikir dan bertindak dengan cepat untuk mencapai tujuan dalam situasi yang ditantang. Melatih reflek pengguna dan meningkatkan kecepatan pemrosesan informasi menjadi aspek penting dalam pengembangan *game* ini. Dengan demikian, *game "Rush Hours"* menggabungkan elemen-elemen tersebut untuk memberikan pengalaman pelatihan kognitif yang menarik dan efektif bagi para pemainnya.

1.2 Rumusan Masalah

Adapun rumusan masalahnya adalah sebagai berikut:

1. Bagaimana penggunaan Algoritma *Collision* dapat meningkatkan pengalaman bermain *game*?
2. Bagaimana *game "Rush Hours"* dapat menjadi *game* yang kompetitif antar sesama pemain?

1.3 Ruang Lingkup

Ruang lingkup mencakup batasan-batasan yang dibahas di dalam Tugas Akhir ini, yaitu sebagai berikut:

1. Pembuatan *game* berbasis *web*.
2. *Game* ini dibuat dengan menggunakan *Visual Studio Code,* bahasa pemrograman *HTML, JavaScript dan CSS*.
3. Pengimplementasian *algoritma collision* pada sebuah *game*.

1.4 Tujuan Penelitian

Adapun tujuan yang diinginkan dalam penelitian dan penyusunan penulisan ini adalah:

1. Pembuatan aplikasi *game* yang mudah dimengerti pengguna (*user friendly)*, sehingga diminati oleh pengguna.
2. Menjadikan *game* ini dapat dimainkan oleh *single player* ataupun *multiplayer* dengan metode *highscore.*

1.5 Metode Penelitian

Metode penelitian yang digunakan dalam pembuatan *game* ini adalah *Prototype,* yaitu sebagai berikut:

1. **Identifikasi Masalah**
   Penelitian dengan penerapan atau pengimplementasian aplikasi permainan *Rush Hours* dengan bahasa pemrogaman *JavaScript* dan melakukan perbandingan *game* sejenis pada mesin *arcade* lawas yang telah ada sebelumnya.
2. **Analisis Data**
   Penggunaan metode ini adalah menganalisa akan *game* sejenis yang sudah pernah ada sebelumnya untuk mengetahui proses jalannya *game* tersebut,

lalu mempelajari algoritma dasar, sehingga *game* yang dibuat ini akan berjalan lancar dan minim *bug.*

3. **Perancangan Aplikasi**
   Penulis disini mulai merancang aplikasi mulai dari tampilan *game*, perancangan sistem menggunakan struktur navigasi dan *UML diagram*, yaitu *Use Case Diagram* dan *Activity Diagram.*
4. **Uji Coba**
   Pada tahap ini bertujuan untuk memastikan bahwa tidak ada kesalahan pada *game* yang telah dibuat. Pada tahap ini akan dilakukan dengan tiga tahap, yaitu uji coba terhadap *browser* untuk melihat apakah *game* sudah berjalan sesuai yang diharapkan. Uji coba *game* menggunakan tiga *browser* yang terdapat pada laptop dengan perangkat Lenovo Ideapad Gaming 3.

1.6 Sistematika Penulisan

Sistematika penulisan Tugas Akhir ini dibagi menjadi 5 pokok bahasan, yaitu sebagai berikut:

BAB 1. Pendahuluan

Bagian Pendahuluan berisi latar belakang masalah, ruang lingkup, tujuan penelitian, metode penulisan dan sistematika penulisan.

BAB 2. Tinjauan Pustaka

Berisi tentang tinjauan pustaka yang terkait, seperti struktur navigasi, UML,CSS, dan HTML.

BAB 3. Analisis dan Perancangan

Bagian Analisis dan Perancangan akan menguraikan mengenai proses pembuatan *game* mulai dari perancangan hingga upload *game*. Perancangan meliputi perancangan struktur navigasi, rancangan tampilan, pembuatan *game*, uji coba *game* dan proses *hosting game*.

BAB 4. Hasil dan Pembahasan

Bab ini berisi hasil dan pembahasan, yang berisi gambaran mengenai pelaksanaan penulisan, hasil penulisan, serta pembahasan dari hasil yang diperoleh secara teoritis.

BAB 5. Penutup

Bagian berisi kesimpulan dan saran sebagai masukan agar *game* ini dapat dikembangkan lagi.

BAB 2
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Jurnal penelitian terkait merupakan penelitian yang telah dilakukan peneliti sebelumnya yang dijadikan acuan oleh penulis dalam melakukan penelitian, sehingga penulis dapat memperkaya teori yang digunakan dalam mengkaji penelitian yang akan dilakukan, seperti pada Tabel 2.1.

Tabel 2.1 Tabel Penelitian Terdahulu

No.	Nama Peneliti	Judul	Kelebihan	Kekurangan
1	Lui Haekal Fasha ,Fauziah, M.Gufroni,(2018, ISSN : 2549 - 2837)	Implementasi Algoritma Collision Detection Pada Game Simulator Driving Car	Pada *game* ini, semua sudah tertata dengan baik, adanya sistem *game* yang baik, sehingga dapat menjadi solusi untuk metode pembelajaran pemecahan masalah yang baik.	Kurang jelasnya alur dari permainan tersebut menurunkan minat para pemain untuk memainkan permainan tersebut.

No.	Nama Peneliti	Judul	Kelebihan	Kekurangan
2	Ilham Nurdiansyah, Eka Wahyu Hidayat , Andi Nur Rachman, (2018, ISSN : 2621-1416)	Penerapan Metode Collision Detection pada Game Platformer Mr. Hoax	Pada *game* ini, halaman *user interface* sudah sangat baik. Pemilihan warna dan tema dapat meningkatkan minat para pemain.	Proses deteksi tabrakan untuk banyak objek dalam lingkungan *game* yang padat bisa memakan banyak sumber daya komputasi, terutama pada perangkat dengan keterbatasan daya komputasi, seperti perangkat *mobile.*
3	Sulistyanto Laili R ,Dodik Arwin Dermawan, (2018, ISSN : 2686-2220)	Implementasi Algoritma Collision Detection dan Markov Chain untuk Menentukan Behaviour NPC dan Karakter Player pada Game Higeia	*Assets* yang digunakan cukup bagus dan dapat menarik minat pemainnya terutama anak-anak. Sesuai dengan tujuan *game* ini untuk mengedukasi anak-anak tentang kebersihan.	Terdapat "tunneling" di mana objek melewati satu sama lain dengan kecepatan tinggi tanpa terdeteksi tabrakannya, menyebabkan permainan terlihat tidak realistis.
4	Fauziah,Eka Mutia Putri, (2021 ISSN : 2614-5448)	Rancang Bangun Game Getuk Shooter Menggunakan Algoritma Collision Detection Berbasis Android	*User Interface* yang cukup sederhana memudahkan para pemain untuk memainkannya terutama anak-anak.	Kurangnya animasi pada karakter yang dimainkan oleh pemain, seolah karakter terlihat sangat kaku.

No.	Nama Peneliti	Judul	Kelebihan	Kekurangan
5	Wiwit Mararizki, (2023, ISSN : 2808-5027)	Perancangan dan Pembuatan Game “JUMP CHICKEN” Berbasis Android	*Collision* bekerja sangat baik dan cukup presisi untuk mendeteksi tabrakan di rintangan bagian atas maupun bawah.	Kurangnya kualitas grafis, variasi rintangan, dan menciptakan pengalaman bermain yang lebih memukau dan menarik.

Penulisan mengenai penggunaan Algoritma *Collision* dalam pembuatan *game* telah diteliti melalui lima jurnal terkait. Jurnal pertama membahas implementasi Algoritma *Collision Detection* pada *game* simulator mengemudi mobil, dengan fokus pada sistem *game* yang baik sebagai solusi untuk pembelajaran pemecahan masalah. Namun, kurangnya kejelasan alur permainan dapat menurunkan minat para pemain.

Jurnal kedua mengeksplorasi penerapan metode *collision detection* pada *game platformer* "Mr. Hoax", dengan antarmuka pengguna yang menarik minat pemain. Namun, deteksi tabrakan untuk banyak objek dalam lingkungan *game* yang padat memerlukan sumber daya komputasi yang cukup besar.

Jurnal ketiga menyajikan implementasi *Algoritma Collision Detection* dan Markov Chain untuk mengatur perilaku NPC dan karakter pemain dalam *game* "Higeia". Meskipun assets yang digunakan menarik minat anak-anak, adanya masalah "tunneling" mengurangi realisme permainan.

Jurnal keempat mendokumentasikan rancang bangun game "Getuk Shooter" menggunakan *Algoritma Collision Detection* berbasis Android. Antarmuka pengguna sederhana, tetapi kurangnya animasi pada karakter mengurangi tingkat kedinamisan permainan.

Terakhir, jurnal kelima memperkenalkan permainan "Jump Chicken" berbasis Android dengan *Algoritma Collision* yang bekerja baik dan presisi. Namun, permainan tersebut kurang menarik, karena kurangnya kualitas grafis dan variasi rintangan. Melalui tinjauan ini, penggunaan *Algoritma Collision* dalam

pembuatan *game* menawarkan potensi dan tantangan yang harus diperhatikan untuk menciptakan pengalaman bermain yang lebih baik.

Berdasarkan beberapa batasan tersebut peneliti akan membuat *game* yang menggunakan algoritma sederhana dan *Collision Avoidance System* untuk memunculkan lawan secara acak dan mendeteksi tabrakan menggunakan algoritma tersebut.

2.2 Pengertian *Game*

Game berasal dari bahasa Inggris. Dalam kamus bahasa Indonesia istilah *Game* berarti permainan. Menurut Zamroni, Suryawan, dan Jalaluddin (2013), *game* adalah sebuah sistem dimana pemain terlibat dalam konflik buatan. Pemain berinteraksi dengan sistem dan konflik dalam *game*. Dalam permainan terdapat peraturan yang bertujuan untuk membatasi perilaku pemain dan menentukan permainan.

Menurut Singkoh, Lumenta, dan Talenan (2016) *game* yang pertama di dunia diciptakan pada tahun 1963 oleh Steve Russel seorang ahli komputer yang berasal dari Amerika. *Game* tersebut adalah *Spacewar* yang kemudian dikembangkan oleh sebuah tim yang terdiri dari Martin Graetz, Pete Simson, dan Dan Edwards. Hal ini mengubah persepsi masyarakat pada saat itu yang menganggap komputer hanya untuk melakukan pekerjaan yang serius.

Masih menurut Singkoh, Lumenta, dan Tulenan(2016) konsol *game* yang pertama di dunia dibuat oleh Ralph H. Baer, lahir 8 Maret 1922. Seorang Jerman yang telah tinggal di Amerika sejak kecil. Ralph mengembangkan *game* untuk televisi yang dikerjakan di sebuah perusahaan bernama Sanders sekitar tahun 1966. Penemuan ini dikembangkan lebih lanjut menjadi *prototype* konsol *game* pertama bernama *Brown Box* dan dipatenkan pada tahun 1986. Ralph juga menemukan *control* pistol 8 untuk video *game* yang bisa dimainkan di televisi, juga yang pertama di dunia.

Menurut Wafda (2015) jika dilihat dari grafis yang digunakan, *game* dapat digolongkan menjadi dua jenis, yaitu 2D dan 3D. Sementara jika dilihat dari cara

memainkannya *game* memiliki beberapa genre di antaranya: *First-Person Shooter, Role-Playing Game, Arcade, Simulation, Racing,* dan sebagainya.

2.2.1 *First Person Shooter (FPS)*

Menurut Santoso (2020) sesuai nama genrenya yang mengandung kata '*first person'* yang berarti 'orang pertama' dan '*shooter*' yang berarti 'penembak', *game FPS* berfokus pada aksi tembak-menembak dengan beragam senapan yang dimainkan dari pandangan orang pertama atau langsung dari penglihatan mata si karakter yang dimainkan. Jadi *game* jenis ini melatih keahlian pemain untuk membidik musuh. Genre *FPS* adalah salah satu genre yang sangat terkenal dan sangat sering dipakai di *game-game* jaman sekarang. Contoh *game FPS* yang terkenal adalah seri *Battlefield*, seri *Half-life*, seri *Left 4 Dead*, dan seri *Call of Duty*.

2.2.2 *Role Playing Game (RPG)*

RPG adalah genre *game* yang menempatkan pemain untuk berperan sebagai salah satu karakter di dalam *game* tersebut. Karakter yang dimaksud dapat berupa karakter yang telah disediakan oleh *game* atau dibuat sendiri oleh pemain, jika *game* yang dimainkan memiliki fitur tersebut. Kata *role playing* mempunyai arti memerankan sebuah peran dalam sebuah *game.*

RPG sering dimaksudkan sebagai sebuah *game* dengan cerita, memiliki sistem level di mana pemain dapat meningkatkan level-nya untuk menjadi semakin kuat, memiliki sistem status di mana pemain dapat mengalokasikan poin-poin yang didapat ketika berhasil menaikkan level-nya untuk meningkatkan misalnya kekuatan fisik, pertahanan, kecepatan, dan sebagainya, memiliki sistem *skill* atau kemampuan-kemampuan khusus yang dapat digunakan pemain dalam menghadapi musuh, memiliki sistem *equipment* untuk mengganti perlengkapan yang dipakai pemain, dan seringkali memperbolehkan pemain untuk bergerak bebas di dunia game tersebut (Taurusta, 2017). Contoh *RPG* yang terkenal adalah seri *Neptunia*, seri *Final Fantasy*, seri *Dragon Age*, seri *Elder Scrolls*, dan seri *Fallout*.

2.2.3 *Arcade*

Menurut Roberto et.al (2020) *game arcade* adalah *game* singkat yang dimainkan untuk kesenangan dan menghabiskan waktu. Tidak seperti dua genre di atas, *game arcade* biasanya tidak memiliki cerita dan sistem yang mendalam namun *game arcade* memiliki cara main yang lebih bervariasi tergantung *game*nya. Contoh *game arcade* yang terkenal adalah *tetris*, *pac-man*, *tic-tac-toe*, alien *invasion*, dan *snake*.

2.2.4 *Simulation*

Sesuai namanya, 'simulation' berarti 'simulasi'. *Game simulation* adalah game yang mengsimulasikan sesuatu sedekat mungkin dengan dunia nyata, walaupun beberapa diantaranya sengaja mensimulasikannya dengan salah untuk tujuan kesenangan (Kamal et al., 2018). Simulasi yang dimaksud beragam, bisa berupa simulasi kehidupan, 10 simulasi bertani, simulasi mengemudi, simulasi

roket luar angkasa, dan sebagainya. Contoh *game simulation* yang terknal adalah seri *The Sims*, *Kerbal Space Program*, dan *Truck Simulator*.

2.2.5 *Racing*

Racing dalam Bahasa Indonesia artinya balap. Jadi, *game racing* adalah *game* yang menantang pemain untuk beradu kecepatan dan kegesitan dengan komputer atau pemain lain menggunakan beragam kendaraan, seperti mobil, motor, dan sebagainya tergantung *game racing* apa yang dimainkan (Abiyit, 2019). Contoh *game racing* yang terkenal adalah seri *Need For Speed* dan seri *Test Drive*.

2.3 Sejarah *Racing Game*

Menurut Abiyit (2019) *Game* balapan telah menjadi salah satu genre *game* paling populer di seluruh dunia, dengan jutaan pemain di seluruh dunia. Dalam sejarahnya, *game* balapan telah mengalami banyak evolusi dan perkembangan. *Game* balapan pertama kali muncul pada tahun 1970-an dalam bentuk *game arcade* sederhana. Salah satu *game* balapan *arcade* paling awal yang populer adalah "*Gran Trak 10*" yang dirilis pada tahun 1974. *Game* ini menggunakan teknologi grafis vektor untuk menampilkan mobil dan jalanan. Pada tahun 1982, Atari merilis *game* balap mobil populer pertama untuk konsol rumahan, yaitu "*Pole Position*". *Game* ini menampilkan grafis 3D dan merupakan salah satu *game* paling populer pada masanya. Sejak saat itu, *game* mobil balap semakin populer di kalangan pemain *game* di seluruh dunia.

Selama tahun 1990-an dan awal 2000-an, *game* mobil balap menjadi semakin populer dan kompleks. Beberapa *game* mobil balap terkenal pada masa itu adalah "*Need for Speed*", "*Gran Turismo*", dan "*Burnout*". *Game-game* ini menawarkan lebih banyak pilihan mobil, lebih banyak jenis balapan, dan lebih banyak mode permainan. Selain itu, teknologi grafis terus berkembang, membuat *game* mobil-mobilan semakin realistis dan memukau. Dengan perkembangan teknologi *smartphone*, *game* mobil balap kini telah menjadi salah satu jenis *game* paling populer di perangkat *mobile*. Ada banyak *game* mobil balap yang tersedia di toko aplikasi, seperti "*Asphalt*", "*Real Racing*", dan "*CSR Racing*". Dengan

tampilan yang menakjubkan dan kontrol yang mudah, *game* mobil balap di *smartphone* telah memperluas basis penggemar *game* balapan ke seluruh dunia.

Selain itu, *game* balapan *online* juga semakin populer. *Game* ini memungkinkan pemain untuk berkompetisi dengan pemain lain di seluruh dunia secara *real-time*, memperluas pengalaman permainan dan memperkuat komunitas *game*. Dalam beberapa tahun terakhir, *game* balapan juga mulai menggunakan teknologi *virtual reality* dan *augmented reality*, memberikan pengalaman permainan yang lebih mendalam dan mengesankan. Dari *game arcade* sederhana pada tahun 1970-an hingga *game* balapan yang memukau dan realistis pada masa kini, *game* balapan telah memperluas dan memperkaya pengalaman permainan *game* di seluruh dunia.

2.4 Algoritma *Collision Avoidance System*

Algoritma *Collision Avoidance System* (CAS) adalah suatu metode yang digunakan dalam pengembangan permainan atau simulasi untuk menghindari tabrakan antara objek yang bergerak. Tujuan utama CAS adalah memastikan bahwa objek-objek dalam permainan dapat bergerak secara aman tanpa saling bertabrakan.

CAS beroperasi dengan memperhitungkan posisi, kecepatan, arah, dan ukuran objek yang bergerak, serta memprediksi kemungkinan tabrakan di masa depan. Algoritma ini menganalisis informasi tersebut dan menghasilkan keputusan untuk menghindari tabrakan dengan mengubah jalur atau kecepatan objek yang bergerak. (Gugah Alwan Hamanako, 2023.)

Ada beberapa jenis algoritma CAS yang umum digunakan, antara lain:

1. *Rule-based Systems*: Algoritma CAS berbasis aturan menggunakan himpunan aturan yang telah ditentukan sebelumnya. Aturan ini memetakan situasi tertentu dengan tindakan yang harus diambil untuk menghindari tabrakan. Contohnya, jika objek A mendekati objek B dengan kecepatan tinggi, maka objek A harus mengubah arah atau kecepatannya untuk menghindari tabrakan.

2. *Potential Field Systems*: Algoritma CAS berbasis potensial menggunakan konsep medan potensial untuk menghindari tabrakan. Objek-objek diberikan muatan positif atau negatif, dan medan potensial dihasilkan di sekitar mereka. Objek yang bergerak akan merasakan

medan potensial dan mencoba menghindari objek dengan muatan yang sama atau mendekati objek dengan muatan yang berlawanan.

3. *Steering Behaviors*: Algoritma CAS berbasis perilaku menggabungkan berbagai perilaku atau strategi yang dijalankan oleh objek untuk menghindari tabrakan. Setiap perilaku memiliki tujuan yang berbeda, seperti menghindari objek lain, mengikuti jalur yang ditentukan, atau menjaga jarak. Objek menggabungkan perilaku ini dengan cara tertentu untuk menghasilkan perilaku yang adaptif dan menghindari tabrakan.

Implementasi CAS dalam permainan sering melibatkan kombinasi dari beberapa algoritma di atas, tergantung pada kompleksitas permainan dan tujuan yang ingin dicapai. Dengan menggunakan algoritma CAS yang tepat, objek dalam permainan dapat bergerak secara aman dan realistis, meningkatkan pengalaman bermain dan mencegah terjadinya tabrakan yang tidak diinginkan. Berikut adalah rumus CAS:

Jarak = sqrt((x2 - x1)^2 + (y2 - y1)^2)

Rumus di atas adalah salah satu contoh penggunaan algoritma CAS yang juga digunakan pada *game* ini, Rumus tersebut adalah rumus jarak *Euclidean* atau jarak lurus antara dua titik dalam bidang dua dimensi. Berikut adalah penjelasan dari setiap komponennya:

1. **Jarak (*Distance*)**: Ini adalah jarak sebenarnya antara dua titik. Dalam kasus pendeteksian tumbukan, kita menggunakan rumus ini untuk menghitung jarak antara dua objek, misalnya dua mobil dalam permainan balap.
2. **sqrt() (*Square Root*)**: Ini adalah fungsi akar kuadrat, yang digunakan untuk menghitung akar kuadrat dari nilai di dalam tanda kurung. Fungsi ini diterapkan pada hasil perhitungan di dalam tanda kurung untuk mendapatkan jarak akhir.
3. **x1, y1 (Koordinat Titik Pertama)**: Ini adalah koordinat titik pertama dalam sistem koordinat dua dimensi. Ini bisa menjadi posisi atau letak objek lain yang ingin kita hitung jaraknya dari objek kedua.
4. **x2, y2 (Koordinat Titik Kedua):** Ini adalah koordinat titik kedua dalam sistem koordinat dua dimensi. Pada kasus mobil balap, ini bisa menjadi posisi atau letak satu objek.

Rumus ini bekerja dengan menghitung selisih antara koordinat x dan y dari kedua titik, memangkatkannya dengan dua (untuk mengkuadratkan), kemudian menjumlahkan hasil kuadrat dari selisih koordinat x dan y. Akhirnya, hasil penjumlahan ini diakarkan untuk mendapatkan jarak akhir antara dua titik.

Dalam konteks pendeteksian tumbukan, kita dapat menggunakan rumus ini untuk menghitung jarak antara dua objek, misalnya dua mobil. Jika jarak antara objek-objek ini lebih kecil dari atau sama dengan jarak minimum yang ditentukan (misalnya radius mobil), maka kita dapat menyimpulkan bahwa terjadi tumbukan atau tumpang tindih antara objek-objek tersebut yang bergerak.

2.5 Konsep Dasar *Web*

Pada saat ini teknologi berkembang sangat pesat, hal ini disebabkan oleh banyak faktor diantaranya perkembangan pola fikir masyarakat yang cukup pesat, untuk memenuhi kebutuhan masyarakat dalam hal informasi dan ilmu pengetahuan serta mekanis dunia kerja, maka dibutuhkan para pengembang aplikasi *web* supaya dapat terus beraktifitas dan berinovasi. *Web* suatu jaringan yang bisa mempermudah, serta mempercepat penyampaian informasi secara luas, dan dapat diakses dengan mudah dan cepat oleh siapapun yang mendapatkan akses internet.

Menurut Sibero (2013) *web* adalah suatu sistem yang berkaitan dengan dokumen digunakan sebagai media untuk menampilkan teks, gambar, multimedia, dan lainnya pada jaringan internet. Sedangkan menurut Kustiyahningsih dan Devie (2011) web merupakan salah satu layanan yang didapat oleh pemakai komputer yang terhubung dengan fasilitas *hypertex*t untuk menampilkan data berupa teks, gambar, suara, animasi, dan multimedia lainnya. Berdasarkan dari teori tersebut, penulis menarik kesimpulan *web* adalah fasilitas *hypertext* untuk menampilkan data dan berisikan dokumen-dokumen 9 multimedia yang berupa teks, gambar, suara, animasi dan lainnya dengan mengunakan *browser* sebagai perangkat lunak untuk mengaksesnya.

2.5.1 *Website*

Dalam dunia teknologi yang pesat ini diperlukan suatu jaringan yang bisa mempermudahkan serta mempercepat penyampaian informasi secara luas, dan dapat dengan mudah dan cepat oleh siapapun yang mendapatkan akses *internet*.

Menurut Bekti (2015) menyimpulkan bahwa *website* merupakan kumpulan halaman-halaman yang digunakan untuk menampilkan informasi teks, gambar diam atau gerak, animasi, suara,dan atau gabungan dari semuanya, baik yang bersifat statis maupun dinamis yang membentuk satu rangkaian bangunan yang saling terkait, yang masing-masing dihubungkan dengan jaringan-jaringan halaman.

Menurut Rahmadi (2013) *website* (lebih dikenal dengan sebutan situs) adalah sejumlah halaman *web* yang memiliki topik saling terkait, terkadang disertai pula dengan berkas-berkas gambar, video atau jenis-jenis berkas lainnya. Sedangkan menurut Ippho Santoso dalam Rahmadi (2013) "membagi *website* menjadi golongan kanan dan golongan kiri. Dalam *website* dikenal dengan sebutan *website* dinamis dan *website* statis.

Website statis adalah *website* yang mempunyai halaman konten yang tidak berubah-ubah. *Website* dinamis merupakan *website* yang secara struktur ditujukan untuk *update* sesering mungkin. Dari uraian teori di atas penulis menarik kesimpulan *website* adalah kumpulan halaman-halaman yang dapat menampilkan teks, gambar, animasi, video, suara yang masing-masing dihubungkan dengan jaringan-jaringan halaman. *Website* dibagi menjadi dua golongan, yaitu *website* statis dan *website* dinamis.

2.5.2 *Internet*

Internet sebagai jaringan terbesar sebagai sumber informasi yang telah menjadi kebutuhan banyak orang. *Internet* menyimpan berbagai jenis informasi yang tidak terbatas. *Internet* berperan sebagai sarana komunikasi, publikasi, serta sarana untuk mendapatkan berbagai informasi yang dibutuhkan. Menurut Supriyanto (2007) *internet* adalah sebuah jaringan komputer global, yang terdiri dari jutaan komputer yang saling berhubungan dengan menggunakan protokol yang sama untuk berbagi informasi secara bersama".

Sedangkan menurut Sibero (2013) *internet (Interconnected Network)* adalah "jaringan komputer yang menghubungkan antar jaringan secara global, *internet*, dapat juga disebut jaringan dalam suatu jaringan yang luas". Berdasarkan pendapat para ahli, penulis dapat menyimpulkan *internet* adalah kumpulan dari jutaan komputer yang terhubung melalui jaringan global untuk membagi informasi secara bersama dengan mencangkup suatu jaringan yang sangat luas.

2.5.3 *Web Server*

Pada umumnya *web server* berperan sebagai *server* yang memberikan layanan kepada komponen yang meminta informasi berkaitan dengan *web*, dalam *web* yang telah dirancang dalam *internet.* Menurut Sibero (2013) "*web server* adalah sebuah komputer yang terdiri dari perangkat keras dan perangkat lunak". Sedangkan menurut Kustiyahningsih dan Devie (2011) "*web server* adalah komputer yang digunakan untuk menyimpan dokumen-dokumen *web*, komputer ini melayani permintaan dokumen *web* dari kliennya".

Dari penjelasan teori di atas, penulis menyimpulkan *web server* adalah komputer yang digunakan untuk menyimpan dokumen dengan mengakses dan menampilkan halaman *web* tersebut dari komputer *client.*

2.5.4 *Web browser*

Peralatan elektronik saat ini dilengkapi oleh *web browser*, mulai dari komputer, *handphone* ataupun *gadget* telah dilengkapi *web browser* yang biasa digunakan untuk menjelajah *internet*. *Web browser* dapat diartikan sebagai *tools* atau aplikasi yang digunakan untuk mencari informasi, membuka atau menjelajah halaman *internet* melalui *web.* Menurut Kustiyahningsih dan Devie (2011) *web browser* adalah *Software* yang digunakan untuk menampilkan informasi dari *server web*.

Sedangkan menurut Sibero (2013) *web browser* adalah aplikasi perangkat lunak yang digunakan untuk mengambil dan menyajikan sumber informasi *web*. Sejalan dengan teori di atas, penulis menyimpulkan *web browser* adalah sebuah

aplikasi atau *software* yang digunakan untuk menampilkan sumber informasi yang disajikan dari *web server*.

2.6 Bahasa Pemograman

Bahasa pemograman suatu perangkat lunak yang menggunakan bahasa-bahasa pemograman yang digunakan untuk merancang atau membuat program sesuai keinginan dan kegunaan (Naufal, 2018). Diantaranya adalah:

2.6.1 *HTML*

Pada umumnya *HTML* suatu bahasa yang digunakan untuk membuat halaman *web. HTML* juga dikenal sebagai aplikasi yang memiliki kemampuan browser. Menurut Sutarman (2007) *HTML Hypertext Markup Language)* adalah suatu bahasa yang digunakan untuk menulis halaman *web*.

Sedangkan menurut Larry (2012) *Hypertext Markup Language* merupakan suatu metode untuk mengimplementasikan konsep *hypertext* dalam suatu naskah atau dokumen. Jadi, dapat disimpulkan bahwa *HTML* adalah bahasa pemograman yang digunakan untuk menulis halaman *web* dengan metode untuk mengimplementasikan konsep *hypertext* dalam suatu naskah atau dokumen.

2.6.2 *CSS*

Menurut Wahyudi (2017) *CSS* biasanya digunakan untuk mengatur tampilan elemen yang tertulis dalam bahasa *markup*, seperti *HTML*. *CSS* berfungsi untuk memisahkan konten dari tampilan visualnya di situs. *CSS* dibuat dan dikembangkan oleh ***W3C*** (*World Wide Web Consortium*) pada tahun 1996 untuk alasan yang sederhana. Dulu *HTML* tidak dilengkapi dengan *tags* yang berfungsi untuk memformat halaman. Anda hanya perlu menulis *markup* untuk situs.

Tags, seperti ***<font>***, diperkenalkan di *HTML* versi 3.2, dan ketika itu menyebabkan banyak masalah bagi *developer.* Karena *website* memiliki berbagai *font*, warna *background*, dan *style*, maka untuk menulis kembali (*rewrite*) kode

memerlukan proses yang sangat panjang dan sulit. Oleh sebab itu, *W3C* membuat *CSS* untuk menyelesaikan masalah ini.

HTML dan *CSS* memiliki keterikatan yang erat. Karena *HTML* adalah bahasa *markup* (fondasi situs) dan *CSS* memperbaiki *style* (untuk semua aspek yang terkait dengan tampilan *website*), maka kedua bahasa pemrograman ini harus berjalan beriringan.

2.6.3 *JavaScript*

Menurut Siahaan & Rismon (2020) *JavaScript* adalah bahasa pemrograman yang digunakan untuk pengembangan *website* agar lebih dinamis. Ibarat kata, *JavaScript* memberikan "kehidupan" dalam *website*, sehingga terciptanya interaksi antara pengunjung dengan situs tersebut. *Website* dinamis yang dimaksud berarti konten di dalamnya dapat bergerak atau mengubah apapun yang tampak di layar tanpa harus dimuat ulang secara manual. Misalnya seperti konten gambar animasi, *maps*, *slideshow*, *polling*, dan sebagainya.

Elemen-elemen tersebut tentunya membuat *website* menjadi lebih menarik, sehingga pengunjung jadi betah mengeksplorasi isi di dalamnya. Awalnya, *JavaScript* hanya bekerja pada sisi *client/frontend* saja. Dengan begitu, proses pengolahan kode-kodenya hanya berjalan di sisi *browser*. Namun, seiring perkembangannya, *JavaScript* juga bisa digunakan di sisi *server.*

Penggunaan *JavaScript* dalam pengembangan *website* sering dikaitkan dengan *HTML* dan *CSS.* Hal ini karena dalam pembuatan *website*, ketiga elemen tersebut berperan penting dan saling berkaitan satu sama lain. Ilustrasi berikut menggambarkan fungsi *HTML, CSS,* dan *JavaScript* ketika membangun sebuah *website*.

2.7 Multimedia

Multimedia adalah penggunaan komputer untuk menyajikan dan menggabungkan teks, suara, gambar, animasi dan video dengan alat bantu (*tool*) dan koneksi (*link*), sehingga pengguna dapat bernavigasi, berinteraksi, berkarya dan berkomunikasi (Hofstetter 2001). Multimedia sering digunakan dalam dunia

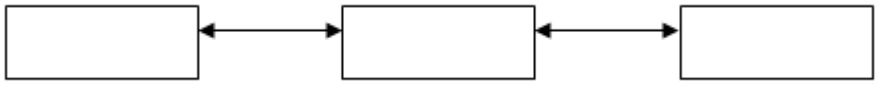
hiburan. Selain dari dunia hiburan, Multimedia juga diadopsi oleh dunia *Game*. Multimedia juga dapat diartikan sebagai penggunaan beberapa media yang berbeda dalam menyampaikan informasi berbentuk *text*, audio, grafik, animasi, dan video.

2.8 Struktur Navigasi

Struktur atau alur dalam sebuah program yang merupakan rancangan hubungan dan rantai kerja dari beberapa area yang berbeda dan dapat membantu mengorganisasikan seluruh elemen pembuatan *website* (Prihatna, 2005). Dalam membangun suatu *website* struktur navigasi merupakan hal yang sebaiknya dilakukan sebelum membuat suatu *website*. Ada empat bentuk dasar dari struktur navigasi yang biasa digunakan dalam membuat suatu *website*, yaitu:

2.8.1 Struktur Navigasi Linier

Struktur navigasi linier adalah struktur yang memiliki suatu rangkaian alur cerita berurut. Navigasi pada jenis ini terdapat pada satu halaman sebelumnya atau sesudahnya, tidak dapat dua halaman sebelumnya atau sesudahnya, seperti pada Gambar 2.1.



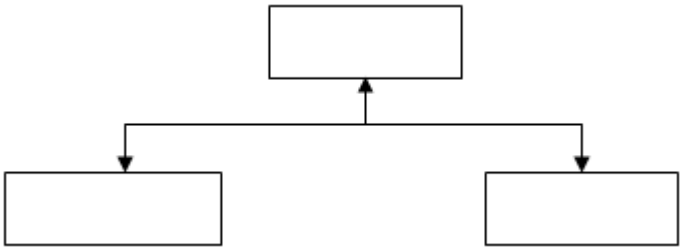
Gambar 2.1 Struktur Navigasi Linier

2.8.2 Struktur Navigasi Hirarki

Menurut Binanto (2010) Struktur dasar ini disebut juga struktur “liniear dengan percabangan”, karena pengguna melakukan navigasi di sepanjang cabang pohon struktur yang terbentuk oleh logika isi.

Struktur Navigasi Hirarki ini biasa disebut dengan struktur bercabang, Struktur ini mengandalkan percabangan untuk menampilkan data berdasarkan kriteria tertentu. Tampilan pada menu satu akan disebut *Master Page* (Halaman utama pertama), halaman utama ini memiliki halaman pendukung yang disebut dengan *Slave Page* (Halaman pendukung), jika salah satu halaman pendukung

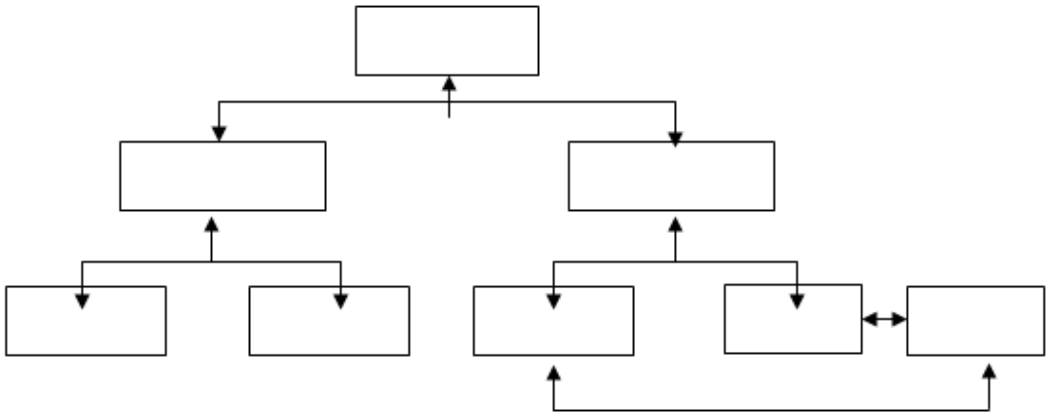
dipilih atau diaktifkan, maka halaman tampilan tersebut akan bernama *Master Page*(Halaman utama kedua), dan seterusnya, seperti pada Gambar 2.2.



Gambar 2.2 Struktur Navigasi Hirarki

2.8.3 Struktur Navigasi Komposit

Struktur Navigasi Komposit (Campuran) disebut juga dengan struktur navigasi bebas yang merupakan gabungan dari ketiga struktur yang ada. Struktur ini biasanya digunakan dalam membuat multimedia, karena dapat memberikan keinteristikan yang lebih tinggi, seperti pada Gambar 2.3.



Gambar 2.3 Struktur Navigasi Komposit

2.9 *Use Case Diagram*

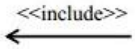
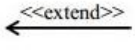
Use Case Diagram Menurut Munawar (2005) *Use Case* adalah deskripsi fungsi dari sebuah sistem dari perspektif pengguna. *Use Case* bekerja dengan cara mendeskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan

sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan *system* disebut *scenario*. Setiap scenario mendeskripsikan urutan kejadian. *Use Case* diagram menampilkan *actor*, *Use Case*, dan hubungan antara *Actor* dan *Use Case*:

1. Aktor mana yang menggunakan *Use Case* mana.
2. *Use Case* mana yang memasukkan *Use Case* lain.

Simbol *Use Case* dapat dilihat seperti pada Tabel 2.2.

Tabel 2.2 Simbol *Use Case*

Simbol	Keterangan
	Aktor : Mewakili peran orang, sistem yang lain, atau alat ketika berkomunikasi dengan *use case*
	*Use case* : Abstraksi dan interaksi antara sistem dan aktor
	Association : Abstraksi dari penghubung antara aktor dengan use case
	*Generalisasi* : Menunjukkan spesialisasi aktor untuk dapat berpartisipasi dengan use case
	Menunjukkan bahwa suatu use case seluruhnya merupakan fungsionalitas dari use case lainnya
	Menunjukkan bahwa suatu use case merupakan tambahan fungsional dari use case lainnya jika suatu kondisi terpenuhi

Sumber: Jagoanhosting.com

2.10 *Activity Diagram*

Activity Diagram Menurut Munawar (2005) *Activity Diagram* adalah teknik untuk mendeskripsikan logika *procedural*, proses bisnis dan aliran kerja dalam banyak kasus. *Activity Diagram* mempunyai peran seperti halnya *flowchart*, akan

tetapi perbedaannya dengan *flowchart* adalah *Activity Diagram* bisa mendukung perilaku paralel sedang *flowchart* tidak bisa.

Activity Diagram mesti digunakan sejajar (*horizontal*) dengan teknik pemodelan lainnya, seperti diagram *Use Case* dan *Diagram State*. Kamu bisa menggunakan *Activity Diagram* agar dapat memodelkan alur kerja sistem dengan baik. *Activity Diagram* berfungsi juga untuk menganalisis diagram *Use Case* dengan cara mendeskripsikan aktor, tindakan yang perlu dilakukan, dan kapan harus terjadi. Simbol *Activity Diagram* dapat dilihat seperti pada Tabel 2.3.

Tabel 2.3 Simbol *Activity Diagram*

Simbol	Nama	Keterangan
	Status awal	Sebuah diagram aktivitas memiliki sebuah status awal.
	Aktivitas	Aktivitas yang dilakukan sistem,aktivitas biasanya diawali dengan kata kerja.
	Percabangan / Decision	Percabangan dimana ada pilihan aktivitas yang lebih dari satu.
	Penggabungan / Join	Penggabungan dimana yang mana lebih dari satu aktivitas lalu digabungkan jadi satu.
	Status Akhir	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir
	Swimlane	Swimlane memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

Sumber : Dicoding.com

2.11 Visual Studio Code

Visual Code Studio adalah sebuah *code editor* gratis yang bisa dijalankan di perangkat *desktop* berbasis Windows, Linux, dan MacOS. *Code editor* ini dikembangkan oleh salah satu raksasa teknologi dunia, Microsoft.

Visual Code adalah *software editor* yang *powerful*, tapi tetap ringan ketika digunakan. Dapat digunakan untuk membuat dan meng-*edit source code* berbagai bahasa pemrograman. Misalnya, seperti *JavaScript*, *TypeScript*, dan *Node.js* (Permana, 2019).

Visual Code Studio juga kompatibel dengan bahasa dan *runtime environment* lain, *seperti PHP*, *Python*, *Java*, dan *.NET*. Hal ini berkat ekosistemnya yang luas dan ketersediaan *extension* yang melimpah.

2.11.1 Komponen Visual Studio Code

Visual Studio Code (VS Code) adalah sebuah lingkungan pengembangan terintegrasi (IDE) yang populer dan kuat, terutama digunakan untuk pengembangan perangkat lunak. VS Code memiliki sejumlah komponen visual yang memfasilitasi pengalaman pengguna yang intuitif dan produktif (Permana, 2019). Antara lain:

2.11.1.1 *Customize*

Digunakan untuk menambahkan ekstensi bahasa pemprograman. konfigurasi dan kustomisasi *template* dengan menambahkan ekstensi bahasa pemprograman maka kita tidak perlu selalu mengingat fungsi Bahasa.

2.11.1.2 *Command Palette*

Command palette menyediakan banyak akses perintah, kita bisa memberikan perintah *editor* membuka *file*, mencari *file* dan sebagainya dengan cepat dan mudah untuk membuka *commad palette* bisa dengan tekan Ctrl+Shift+p.

2.11.1.3 *Integreted Terminal*

Integrital Terminal digunakan untuk mengeksekusi skrip di *editor*. Kita bisa mengeksekusi skrip *editor* secara langsung di terminal tanpa harus membuka terminal tambahan, ini adalah salah satu kelebihan dari visual studio code.

2.11.1.4 *Extention*

Extention adalah fungsi tambahan dalam ynag berfungsi untuk memperluas kemampuan dari *editor* yang dapat membantu *developer* dalam melakukan *programming*.

2.11.1.5 *Search*

Fitur *search* Visual Studio Code juga sangat cepat dan kemudahan yang diberikan selain kecepatan *query* pencarian data juga akan mencari sampai ke level kontennya.

2.11.1.6 *Grid Editor Layout*

Visual Studio Code memudahkan dalam manajemen *layout*, kita dapat dengan mudah mengatur grup *editor* dalam tata letak apapun baik secara vertikal maupun horizontal.

2.11.1.7 *Color Themes*

Color Themes digunakan untuk memodifikasi warna dalam antarmuka Visual Studio Code agar sesuai dengan selera yang diinginkan, caranya dengan pilih *File >Prefences>Color Theme* , lalu geser cursor keatas dan kebawah untuk memilih tema yang diinginkan.

2.11.1.8 *Cloud Enviroment*

Kita juga bisa melakukan sesuatu di lingkungan *cloud* melalui Visual Studio Code, seperti membuat *database*, melakukan perintah *insert, update , delete* , dan sebagainya di *cloud*.

2.12 *Black Box Testing*

Black box testing atau dapat disebut juga *Behavioral Testing* adalah pengujian yang dilakukan untuk mengamati hasil *input* dan *output* dari perangkat lunak tanpa mengetahui struktur kode dari perangkat lunak (Jaya, 2018). Pengujian ini dilakukan di akhir pembuatan perangkat lunak untuk mengetahui apakah perangkat lunak dapat berfungsi dengan baik. Untuk melakukan pengujian, penguji

tidak harus memiliki kemampuan menulis kode program. Pengujian ini dapat dilakukan oleh siapa saja.

Dalam metode ini, pengujian dilakukan dengan melihat perangkat lunak sebagai "kotak hitam" di mana input diberikan dan output di observasi, tanpa pengetahuan tentang bagaimana perangkat lunak mengolah input tersebut. Tujuannya adalah untuk mengidentifikasi kesalahan, kesalahan logika, atau masalah lain dalam perilaku perangkat lunak yang mungkin muncul saat digunakan oleh pengguna.

Para pengujian dapat merancang kasus uji berdasarkan spesifikasi, kebutuhan, atau pemahaman umum tentang cara perangkat lunak seharusnya beroperasi. *Blackbox testing* memungkinkan penguji untuk melihat perangkat lunak dari perspektif pengguna akhir dan membantu menjamin bahwa perangkat lunak berfungsi dengan benar, independen dari detail teknis internalnya.

BAB 3
ANALISIS DAN PERANCANGAN

3.1 Gambaran Umum

Perkembangan industri *game* sekarang ini sangat pesat terutama dari segi grafis dan *Artificial Intelligent* (AI)-nya, ditandai dengan para pengelola industri *game* berlomba-lomba untuk menciptakan *game* yang lebih mendekati nyata atau riil dan tentunya menarik bagi para pemainnya. Sehingga *game* bukan hanya sekedar hobi untuk mengisi waktu luang, melainkan sebuah cara untuk meningkatkan kreativitas dan tingkat intelektual penggunanya.

3.2 Analisis Kebutuhan *Game*

Tujuan dari menganalisis *game* adalah untuk memahami dengan sebenar-benarnya kebutuhan dari *game* tersebut. Kebutuhan *game* dapat diartikan sebagai pernyataan apa yang harus dikerjakan oleh *game* itu sendiri. Untuk mempermudah dalam menentukan kebutuhan secara keseluruhan, maka analisis kebutuhan dibagi menjadi dua jenis, kebutuhan fungsional dan kebutuhan non-fungsional.

3.2.1 Kebutuhan Fungsional

Ditujukan untuk menentukan proses-proses apa saja yang dapat dilakukan oleh *game* atau informasi yang dapat diproses oleh pengguna, sebagai berikut:

a. Pemain dapat memulai permainan dan memilih permainan untuk menggunakan suara dalam permainannya ataupun tidak.
b. Pemain dapat melihat *Credit* yang menunjukan laman *Instagram* pembuat.
c. Pemain dapat melihat *score* pemain sebelumnya untuk bertanding.

3.2.2 Kebutuhan Non-Fungsional

Dalam pembuatan *game* ini membutuhkan serangkaian peralatan untuk mendukung kelancaran dalam pembuatan dan pada saat pengujian. Ada dua aspek yang digunakan pada pembuatan *game*. Berikut aspek-aspek yang diunakan:

a. Aspek *Hardware*

Saat membuat *Rush Hours hardware* yang digunakan, yaitu:

1. Ryzen 7 5800H @4.4 Ghz
2. Memori 6GB
3. SSD 1TB
4. VGA Nvidia GeForce RTX 3050 4GB

Kebutuhan minimum untuk menjalankan *game* yang dibuat ini adalah:

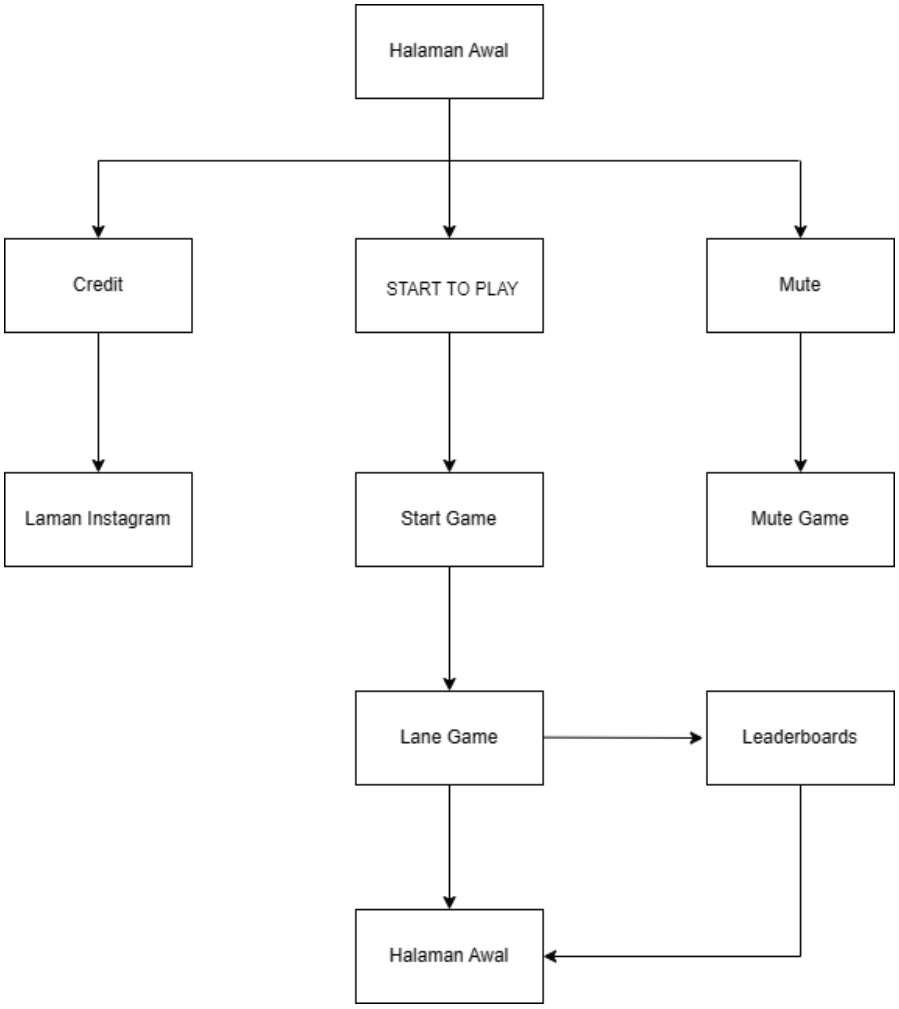
1. Intel Pentium IV atau keatas
2. Memori 512MB
3. Free space HD 2GB
4. VGA 256MB

b. Aspek *Software*

Dalam proses pembuatan game ini perangkat lunak yang digunakan adalah sistem operasi Windows 11 dan Visual Studio Code.

3.3 Perancangan Struktur Navigasi

Struktur navigasi merupakan suatu urutan alur dari suatu program yang merupakan rancangan hubungan antar area yang berbeda. Pada *game Rush Hours*, terdapat struktur navigasi, yaitu struktur navigasi *user*, seperti pada Gambar 3.1.



Gambar 3.1 Navigasi User

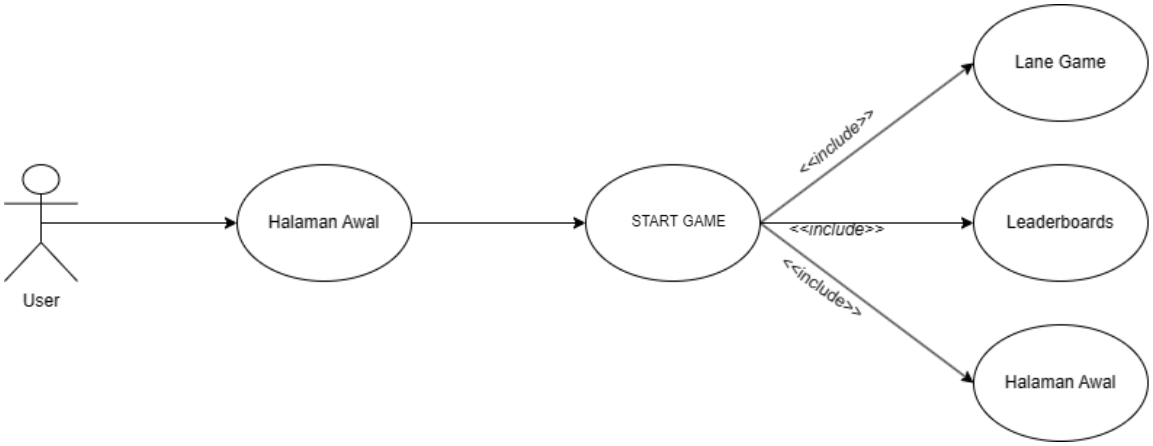
Struktur navigasi *user* menggunakan jenis struktur navigasi linier. Berikut adalah alur struktur navigasi *User*:

a. *User* akan ditampilkan halaman awal yang dimana berisi beberapa pilihan yang dapat dipilih.
b. Didalam halaman utama terdapat pilihan, seperti *Credit*, *Insert Coin*, *Mute* dan panduan untuk bermain.
c. Didalam *Credit*_terdapat informasi pembuat *game* yang mana akan diteruskan ke laman *Instagram* pembuat.
d. Pada pilihan *Insert Coin* digunakan untuk memulai *game*, *Insert Coin* terinspirasi dari halaman awal mesin *Arcade* lawas
e. Apabila *user* memilih opsi *Mute*, maka *game* akan dijalankan dengan tanpa suara.
f. Apabila *user* ingin mengembalikan suara permainan, maka dapat memilih opsi *Mute* maka *game* akan dijalankan dengan suara.

g. Pada bagian *Start Game* terdapat hitung mundur sebagai penanda *game* telah dimulai.
h. Pada *Lane Game* terdapat 3 baris jalur yang dapat *User* gunakan untuk berpindah menghindari mobil lawan.
i. Setelah pertandingan selesai maka akan ditunjukan *Leaderboards* dimana berisi *score score* pemain sebelumnya.
j. Setelah itu *game* akan memunculkan halaman awal *game* untuk memulai dari awal permainan.

3.4 *Use Case Diagram*

Pada Gambar 3.2 adalah *Use Case Diagram* dari *game Rush Hours.*



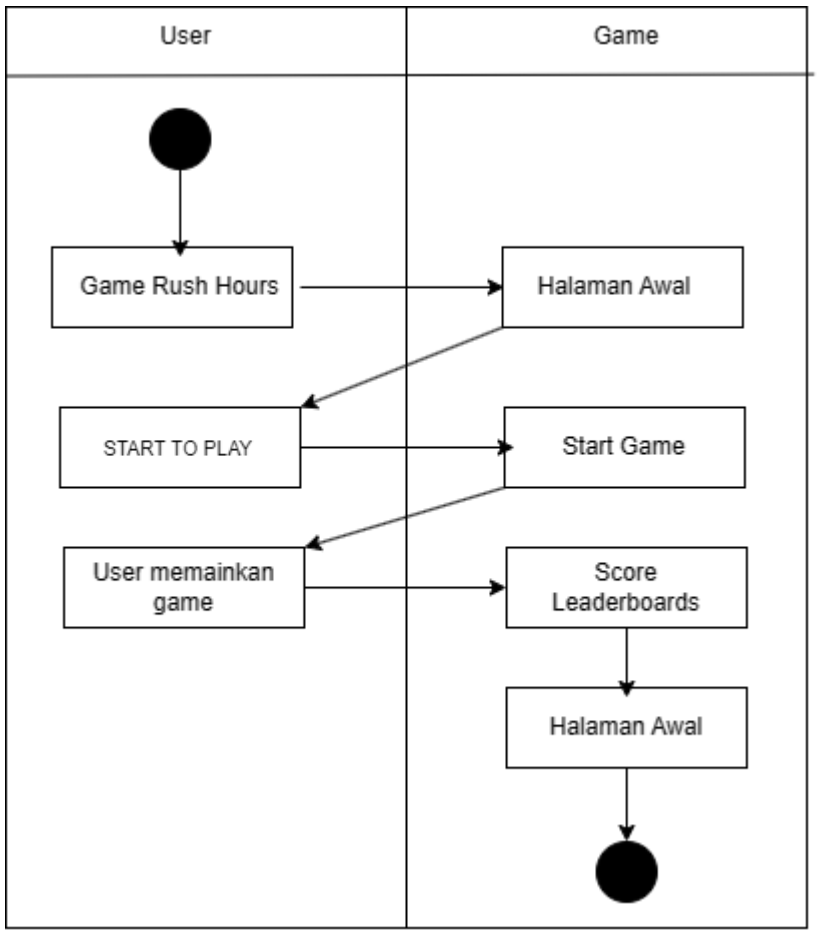
Gambar 3.2 *Use Case Diagram*

Rancangan Use Case Diagram untuk *User* berguna untuk *actor* dari diagram ini.

1. *User* dapat memilih halaman utama:
   User dapat mengakses menu utama yang menampilkan opsi dan fitur yang tersedia dalam permainan.
2. *User* dapat melakukan *Start Game* untuk memulai *game*:
   User dapat menekan tombol *"Start Game"* untuk menginisiasi permainan dan memulai pengalaman bermain.
3. *User* dapat mengakses *Lane Game*:
   Setelah memulai permainan, *User* dapat memilih *lane game* yang diinginkan untuk memulai permainan dan menghindari mobil lawan.
4. *User* dapat mengakses *Leaderboards*:
   Setelah menyelesaikan permainan, *User* dapat mengakses *leaderboards* untuk melihat peringkat mereka dan peringkat pemain lainnya.
5. *User* dapat kembali ke halaman awal:
   Setelah menyelesaikan permainan dan melihat leaderboards, *User* dapat kembali ke halaman awal untuk mengakses menu utama atau fitur lainnya.

3.5 *Activity Diagram*

Berikut adalah tabel *activity diagram* pada *game Rush Hours* dapat dilihat pada Gambar 3.3.



Gambar 3.3 *Activity Diagram*

Saat *user* memasuki *game*, mereka akan dihadapkan dengan halaman utama yang menampilkan berbagai opsi dan fitur yang tersedia dalam permainan. Halaman utama ini menjadi titik awal bagi *user* untuk memulai permainan.

Setelah memilih untuk memulai permainan, user akan langsung dibawa ke halaman *Lane Game*. Halaman ini merupakan tempat utama di mana *user* dapat benar-benar terlibat dalam permainan.

Setelah *user* menyelesaikan permainan, mereka akan disambut dengan munculnya *Leaderboards* bersamaan dengan Halaman Awal permainan. *Leaderboards* memberikan informasi tentang pencapaian skor *user*, dan pencapaian

dari pemain lain dalam permainan. *User* dapat melihat prestasi mereka sendiri dan membandingkannya dengan pemain lain untuk meningkatkan motivasi dan semangat kompetisi. Sementara itu, Halaman Awal permainan memungkinkan *user* untuk kembali ke menu utama dan memulai permainan baru.

Secara keseluruhan, halaman utama, *lane game, leaderboards*, dan halaman awal merupakan bagian penting dalam pengalaman permainan *user*. Mereka membentuk alur yang mulus dan memberikan *user* kesempatan untuk memulai, bermain, dan berkompetisi dalam permainan.

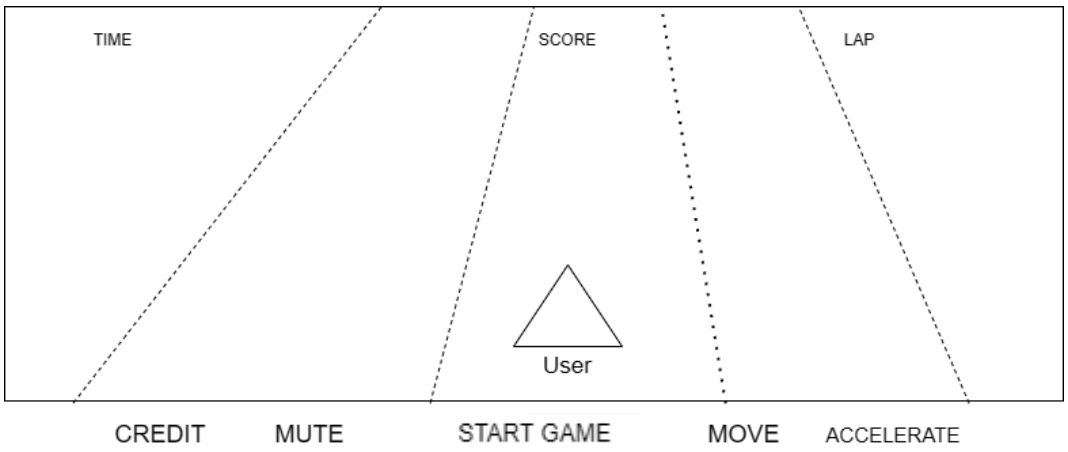
3.6 Perancangan Tampilan Halaman Utama

Rancangan awal pada *game* ini di awali dengan halaman utama yang terdiri dari Judul *Game*, *Credit, Mute, Start Game,* dan Panduan tombol, seperti pada Gambar 3.4.

Gambar 3.4 Rancangan Halaman Utama

3.6.1 Perancangan Tampilan *Lane Game*

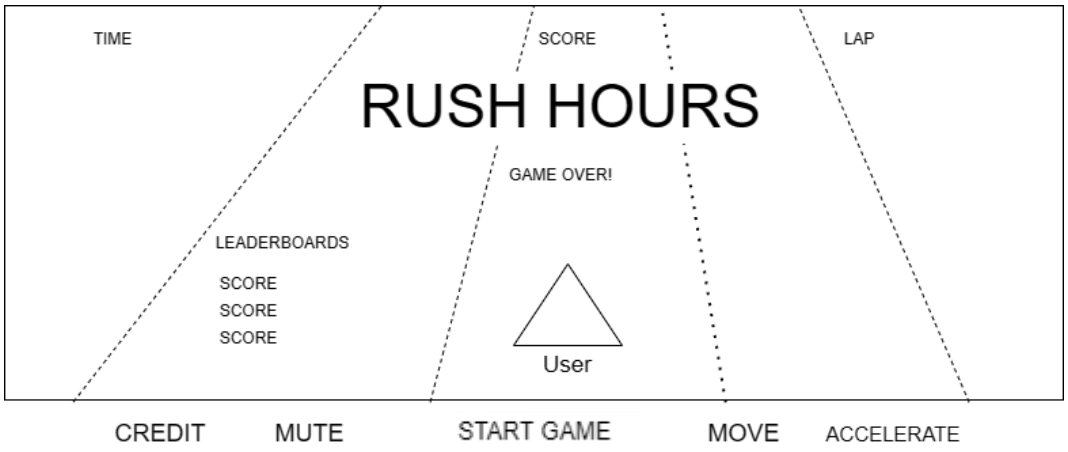
Rancangan awal tampilan *lane game* terdapat 3 jalur yang dapat digunakan user untuk berpindah-pindah untuk menghindari mobil lawan, seperti pada Gambar 3.5.



Gambar 3.5 Rancangan Halaman *Lane Game*

3.6.2 Perancangan Tampilan *Highscore (Leaderboards)*

Rancangan awal tampilan *leaderboards* berisi hasil dari permainan yang telah dimainkan *user*. Pada bagian bawah hasil pertandingan terdapat tombol *Start Game* yang dapat digunakan dan akan mengirim *user* kembali ke halaman awal dan memulai kembali permainan, seperti pada Gambar 3.6.



Gambar 3.6 Rancangan *Highscore*

3.7 *Asset*

Pada *game* ini digunakan beberapa asset untuk menghidupkan dan memberi suasana pada *game* sekaligus membuat tampilan game menjadi lebih menarik.

3.7.1 Mobil *User*

Asset ini akan menjadi asset yang digunakan oleh *user* untuk memainkan permainan. Pada Gambar 3.7 adalah gambar mobil yang dibuat dengan 3 sudut yang berbeda.

Gambar 3.7 Mobil *User*

Asset dibuat dari 3 sudut yang berbeda agar dapat memberikan efek animasi bergerak saat akan berpindah jalur.

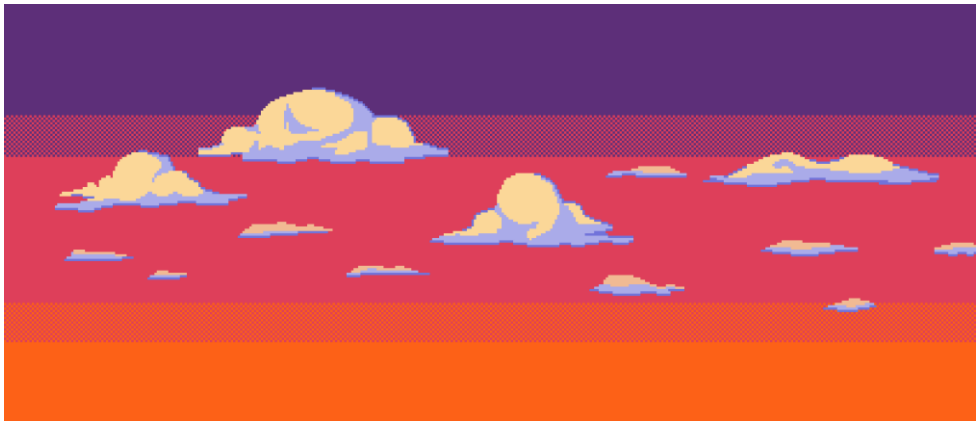
3.7.2 Mobil Lawan

Asset mobil lawan hanya dibuat dalam bentuk 1 sudut saja, karena mobil lawan tidak dapat berpindah jalur, seperti pada Gambar 3.8.

Gambar 3.8 Mobil Lawan

3.7.3 *Background*

Asset *background* digunakan untuk menjadi latar dan menggambarkan suasana permainan, seperti pada Gambar 3.8.



Gambar 3.9 *Background*

3.7.4 Pohon

Asset pohon digunakan sebagai dekorasi dan sekaligus “pembatas” keluar jalur, seperti pada Gambar 3.10.

Gambar 3.10 Pohon

3.7.5 Garis *Finish*

Asset garis *finish* digunakan sebagai penanda selesainya permainan dan batas akhir dicatatnya skor pemain, seperti pada Gambar 3.11.

Gambar 3.11 Garis *Finish*

3.8 Kontrol *Game*

Untuk memainkan *game Rush Hours* pemain dapat menggunakan *keyboard*. Berikut tombol yang digunakan seperti pada Tabel 3.1

Tabel 3. 1 Kontrol *Game*

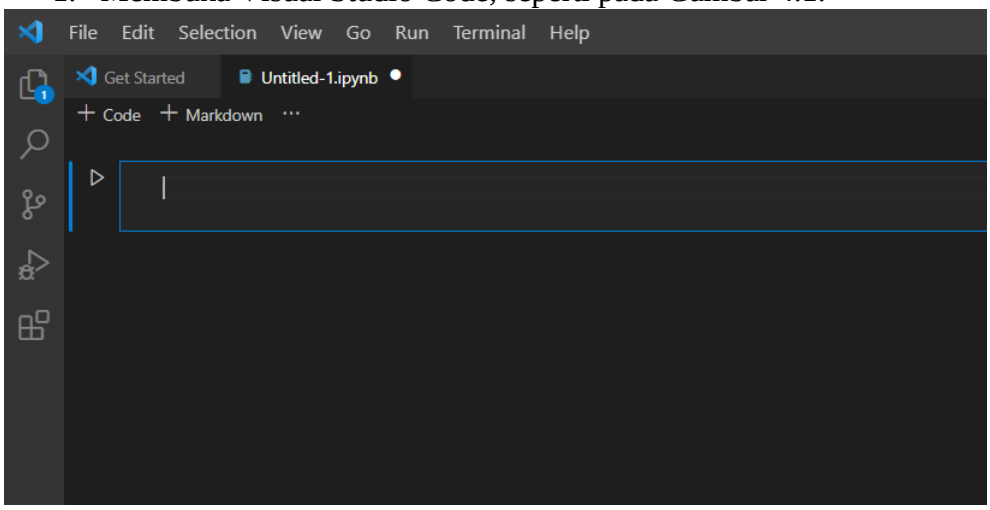
C	: Memuat laman *Instagram*
SPASI	: *Start Game* atau memulai permainan.
M	: Mematikan dan menghidupkan suara dalam *game*.
Panah Kiri dan Kanan	: Pergerakan untuh pindah jalur.
Panah Atas dan Bawah	: Mengatur kecepatan mobil *user*.
Esc	: Tombol untul menjeda permainan

BAB 4
IMPLEMENTASI DAN UJI COBA

4.1 Implementasi

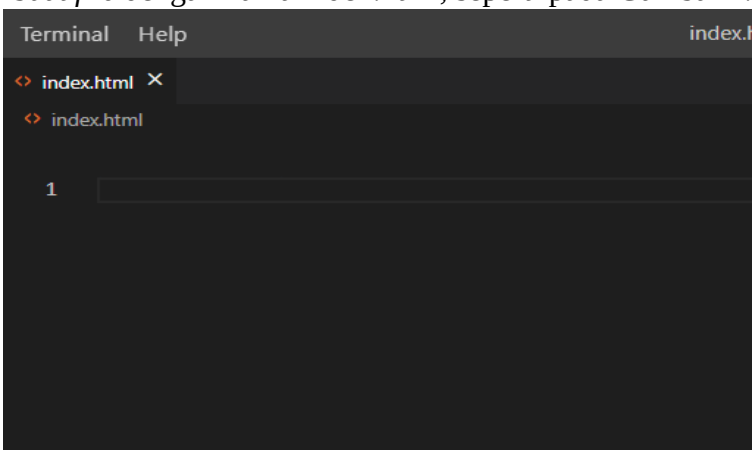
Pada tahap implementasi, Untuk perangkat lunak yang digunakan adalah Visual Studio Code. Berikut adalah pembuatan *game* menggunakan *Visual Studio Code:*

1. Membuka Visual Studio Code, seperti pada Gambar 4.1.



Gambar 4.1 *Editor* Visual Studio Code

2. Membuat *file* dengan nama index.html, seperti pada Gambar 4.2.



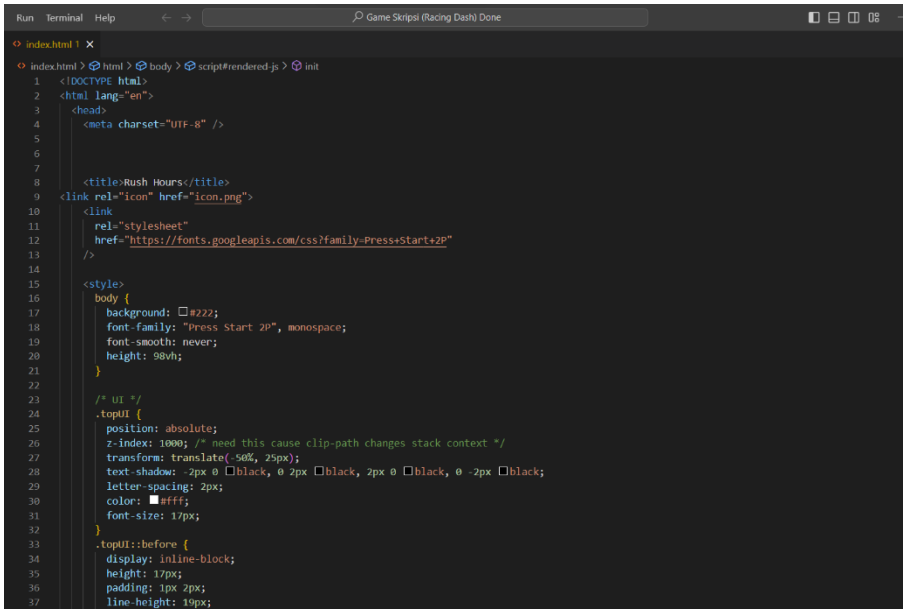
Gambar 4.2 Membuat File *index*

4.2 Pembuatan Halaman *Game*

Pada bagian ini pembuatan halaman *game* dibagi menjadi beberapa tahap.

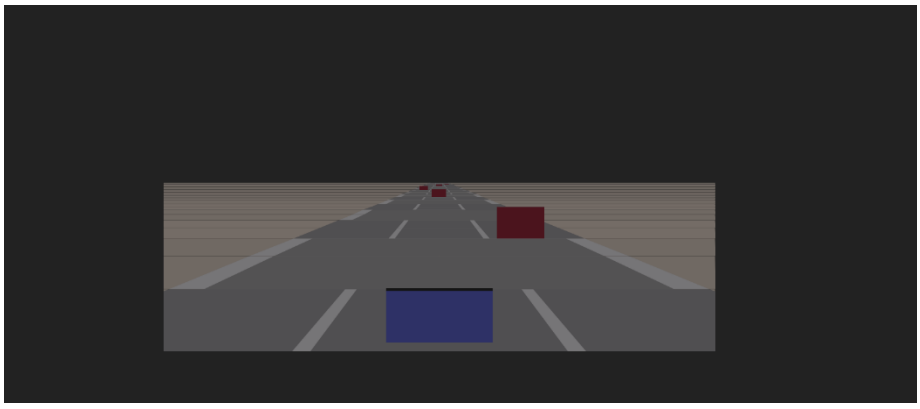
4.2.1 Pembuatan Halaman Utama

Pembuatan halaman utama pada *game* ini menggunakan *file* yang berformat *html.*



Gambar 4.3 Program Halaman Utama

Pada Gambar 4.3 merupakan potongan dari pembuatan halaman utama. *User* akan disuguhi oleh judul *game*, dan berlatar *lane game* tersebut, seperti pada Gambar 4.4.



Gambar 4.4 Halaman Utama

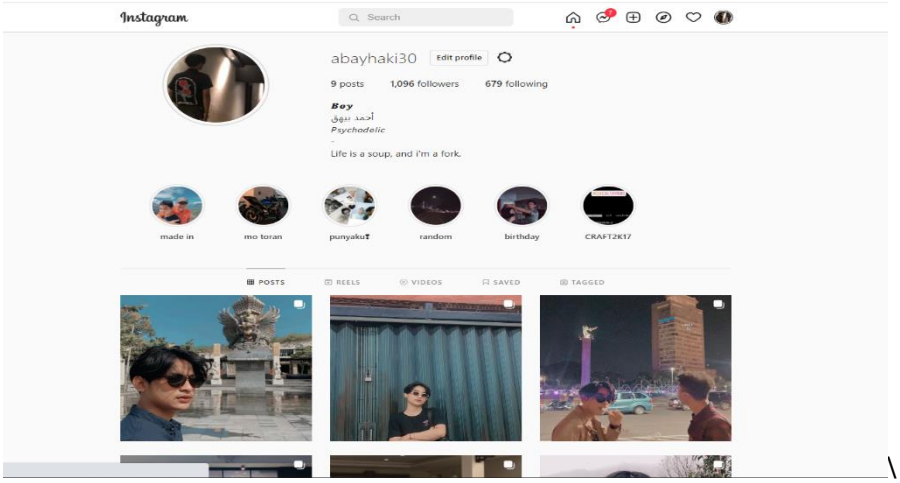
```
<div id="controls">
  <span><span>C</span>Credit</span>
  <span><span>M</span>Mute</span>
  <span><span>Space</span>Insert Coin</span>
  <span><span>&lt;</span><span>&gt;</span>Move</span>
  <span><span>&lt;</span><span>&gt;</span>Accelerate</span>
```

Gambar 4.5 Program Tombol Petunjuk

Pada Gambar 4.5 merupakan Program untuk menambahkan tombol petunjuk untuk memainkan permainan. Tombol tersebut terisi dari tombol, “C”, yaitu untuk menunjukan *credit* yang dimana akan memuat laman *Instagram* pembuat, kemudian ada tombol “M" yang digunakan untuk mengatur suara permainan, kemudian ada “Space” yang digunakan untuk memulai permainan. Setelah itu diberikan tombol petunjuk untuk memainkan permainan, seperti tombol arah “←→” untuk berpindah jalur dan tombol “⇅“ untuk mengatur laju kendaraan *user*, seperti pada Gambar 4.6.

Gambar 4.6 Tombol Petunjuk

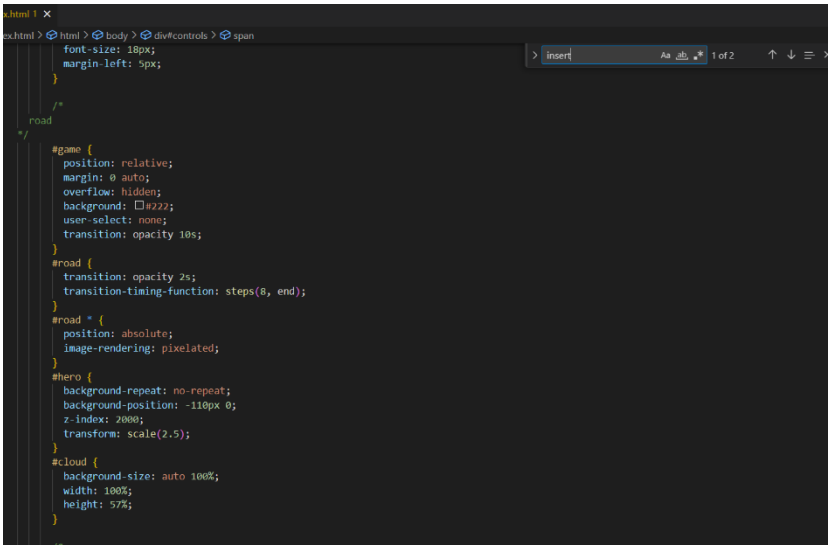
Pada Gambar 4.6 terdapat beberapa tombol termasuk tombol *credit*, apabila tombol ditekan, maka akan terbuka tab baru dan menghasilkan output, seperti pada Gambar 4.7.



Gambar 4.7 Laman *Instagram*

4.2.2 Halaman *Lane Game*

Setelah *user* memulai permainan, maka laman akan dialihkan ke halaman *lane game* yang dibuat dengan campuran bahasa pemrograman *html*, *javascript*, dan *CSS*.



Gambar 4.8 Pembuatan Program *Lane Game*

Pada Gambar 4.8 adalah potongan dari bahasa pemrograman *javascript*, program di atas berfungsi untuk pengalihan dari halaman utama menuju ke halaman *lane game*. Program tersebut adalah kode CSS yang mengatur tata letak dan gaya untuk elemen-elemen dalam sebuah permainan.

Bagian pertama, dengan *selector #game*, mengatur properti-posisi, *margin*, *overflow*, latar belakang, *user-select*, dan transisi-opasitas untuk permainan itu sendiri. Selanjutnya, *selector #road* mengatur transisi-opasitas, transisi-*timing-function*, dan posisi untuk elemen yang mewakili jalan dalam permainan. Dalam hal ini, transisi-opasitas akan berlangsung selama 2 detik dengan 8 langkah, menghasilkan animasi yang tersegmentasi.

Selanjutnya, aturan *#road* berlaku untuk semua elemen yang berada di dalam elemen jalan (*#road*). Aturan ini mengatur posisi absolut dan *rendering* gambar pikselated untuk semua elemen dalam jalan permainan.

Selanjutnya, *selector #hero* digunakan untuk elemen yang mewakili *user* dalam permainan. Aturan ini mengatur pengulangan latar belakang, posisi latar belakang, z-index, dan transformasi skala untuk elemen pahlawan tersebut. Terakhir, aturan

#cloud mengatur ukuran latar belakang, lebar, dan tinggi untuk elemen yang mewakili awan dalam permainan, seperti pada Gambar 4.9.

Gambar 4.9 *Lane Game*

4.2.3 Halaman *Highscore*

Setelah permainan selesai maka *leaderboards* atau papan skor akan muncul dan menampilkan skor pemain dan pemain sebelumnya.

```
function updateHighscore() {
  let hN = Math.min(12, highscores.length);
  for (let i = 0; i < hN; i++) {
    highscore.children[i].innerHTML = `${(i + 1).pad(2, " ")}. ${
      highscores[i]
    }`;
  }
}
```

Gambar 4.10 Pembuatan Program *Highscore*

Pada Gambar 4.10 adalah program untuk menunjukan skor pada akhir permainan dan meng *update* sesuai dengan urutannya. Pada Gambar 4.11 adalah outputnya.

Gambar 4.11 Halaman *Highscore*

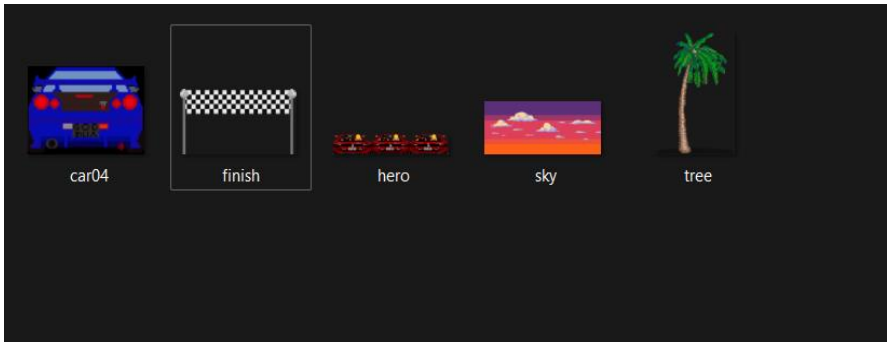
4.3 Penambahan *Assets*

Penambahan *assets* pada *lane game* agar *UI* lebih menarik, *assets* yang digunakan, yaitu mobil *user*, mobil lawan, pohon, garis *finish* dan *background* awan, seperti pada Gambar 4.12.

```
const ASSETS = {
  COLOR: {
    TAR: ["#959298", "#9c9a9d"],
    RUMBLE: ["#959298", "#f5f2f6"],
    GRASS: ["#eedccd", "#e6d4c5"],
  },

  IMAGE: {
    TREE: {
      src: "images/tree.png",
      width: 132,
      height: 192,
    },

    HERO: {
      src: "images/hero.png",
      width: 110,
      height: 56,
    },

    CAR: {
      src: "images/car04.png",
      width: 50,
      height: 36,
    },

    FINISH: {
      src: "images/finish.png",
      width: 339,
      height: 180,
      offset: -0.5,
    },

    SKY: {
      src: "images/sky.png",
    },
```

Gambar 4.12 Program *Assets Game*

Semua gambar disimpan pada *folder* yang sama dan diberi nama *folder images*, seperti pada Gambar 4.13.

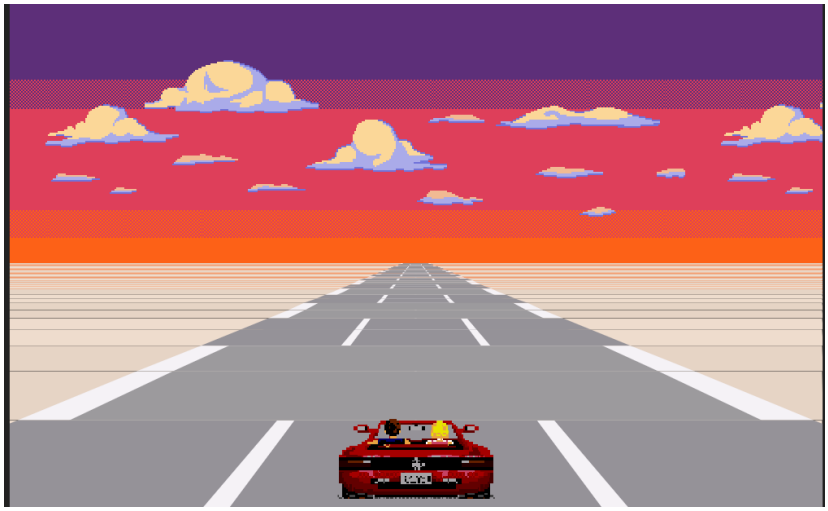


Gambar 4.13 *Folder Assets*

Setelah semua *assets* telah terkoneksi dengan baik maka tampilan *UI* akan berubah dan menampilkan *assets* tersebut, seperti pada Gambar 4.14.

Gambar 4.14 Penambahan *Assets Background*

Assets "sky" berguna untuk menjadi latar dalam *lane game* agar *game* tampak lebih hidup dan tidak kosong.



Gambar 4.15 Penambahan *Assets Hero*

Pada Gambar 4.15 Assets "Hero" disini mewakilkan mobil yang akan digunakan *user* untuk bermain.

Gambar 4.16 Penambahan *Assets Car*

Assets "Car" berfungsi untuk memberikan tampilan pada mobil lawan dan menggunakan jenis mobil yang berbeda agar pemain mudah membedakan antara "Hero" dan mobil lawan.

Gambar 4.17 Penambahan *Assets Tree*

Pada Gambar 4.17 Penambahan *assets* ini bertujuan untuk menjadi dekorasi pada bagian pinggir *lane* dan sekaligus menjadi pembatas jalan.

Gambar 4.18 Penambahan *Assets* Garis *Finish*

Sebagai penanda garis akhir permainan ditambahkan juga *assets* garis *finish*. Tampilan pada Gambar 4.18 adalah gambaran apabila semua *assets* telah terpasang dan berfungsi.

4.4 Penambahan Fitur *Game*

Penambahan fitur-fitur pada *game* bertujuan untuk memberikan pengalaman permainan yang lebih baik kepada para pemain. Dengan demikian, penambahan fitur-fitur ini bertujuan untuk memperkaya pengalaman permainan dan memastikan bahwa pemain terus terlibat dan terhibur saat bermain *game*.

4.4.1 Fitur HUD

Penambahan fitur *HUD* (*Heads-Up Display*) pada *game* bertujuan untuk memberikan informasi penting kepada pemain secara langsung dan mudah terlihat selama permainan. Fitur *HUD* dapat berupa tampilan grafis yang menampilkan elemen-elemen, seperti peta, indikator kesehatan karakter, tingkat amunisi, kekuatan khusus, skor, waktu, dan banyak lagi. Dengan adanya fitur *HUD*, pemain dapat dengan cepat melihat informasi tersebut tanpa harus mengganggu pandangan utama dari permainan.

Selain itu, penambahan fitur *HUD* yang baik juga dapat membantu pemain dalam mengambil keputusan yang lebih cepat dan efektif selama permainan. Misalnya, dengan memiliki indikator waktu dan skor yang jelas, pemain dapat dengan mudah mengetahui seberapa banyak waktu yang tersisa dan melihat skor yang telah dicapai, serta mengambil tindakan yang sesuai. Fitur-fitur *HUD* yang

intuitif dan informatif juga dapat meningkatkan imersi pemain dalam permainan, membantu mereka terhubung dengan dunia *virtual* yang sedang mereka jelajahi. Dengan demikian, penambahan fitur *HUD* pada *game* bertujuan untuk memberikan pemain akses yang lebih baik terhadap informasi penting, meningkatkan kemampuan pengambilan keputusan, dan meningkatkan pengalaman keseluruhan dalam bermain *game*, seperti pada Gambar 4.19 dan 4.20.

```
</head>

<body translate="no">
  <div id="game">
    <div id="road">
      <div id="cloud"></div>
      <div id="hero"></div>
    </div>

    <div id="hud">
      <span id="time" class="topUI">0</span>
      <span id="score" class="topUI">0</span>
      <span id="lap" class="topUI">0'00"000</span>
      <span id="tacho">0</span>
    </div>

    <div id="home">
      <h1>Rush Hours</h1>
      <p id="text"></p>


      <div id="highscore"></div>
    </div>
  </div>
```

Gambar 4.19 Program HUD

Gambar 4.20 HUD

HUD yang dihasilkan masih kurang dapat dimengerti, karena peletakan yang kurang bagus dan ukurannya yang terlalu kecil, maka dari itu ditambahkan program CSS untuk memperbagus tampilan *HUD*, seperti pada Gambar 4.21.

```
}
#time {
  left: 13%;
  color: #f4f430;
}
#time::before {
  content: "TIME";
  color: #f57214;
}
#score {
  left: 45%;
}
#score::before {
  content: "SCORE";
  color: #a61a9d;
}
#lap {
  left: 88%;
  width: 45%;
}
#lap::before {
  content: "LAP";
  color: #0082df;
}
```

Gambar 4.21 Penambahan Program CSS *HUD*

Maka tampilan *HUD* akan berubah menjadi lebih mudah dimengerti dan tata letaknya menjadi lebih teratur, seperti pada Gambar 4.22.

Gambar 4.22 *HUD*

4.4.2 Fitur *Tachometer*

Dalam dunia mobil balap, tak terelakkan bahwa *tachometer* menjadi salah satu komponen penting adalah program untuk menambahkan tacho.

```
#tacho {
  position: absolute;
  text-align: right;
  width: 23%;
  bottom: 5%;
  z-index: 2000;
  color: #e62e13;
  text-shadow: -2px 0 black, 0 2px black, 2px 0 black, 0 -2px black;
  letter-spacing: 2px;
  font-size: 23px;
}
#tacho::after {
  content: "km/h";
  color: #fab453;
  font-size: 18px;
  margin-left: 5px;
}
```

Gambar 4.23 Penambahan *Tachometer* Pada CSS

Pada Gambar 4.23 adalah bagian dari tampilan atau *styling* untuk elemen *tachometer* pada permainan *Rush Hours*. Pada CSS, kode ini mendefinisikan tampilan visual untuk elemen dengan **id="tacho"**, yang merupakan elemen untuk menampilkan nilai kecepatan kendaraan dalam permainan.

Penjelasan kode CSS:

1. **#tacho**: Ini adalah *selector* untuk memilih elemen dengan **id="tacho"**. *Selector* ini digunakan untuk mendefinisikan gaya untuk elemen *tachometer*.
2. **position: absolute;**: Ini mengatur posisi elemen secara absolut, artinya elemen akan ditempatkan berdasarkan posisi yang ditentukan oleh properti **top**, **bottom**, **left**, dan **right**.
3. **text-align: right;**: Properti ini mengatur teks di dalam elemen *tachometer* akan diatur rata kanan.
4. **width: 23%;**: Properti ini mengatur lebar elemen *tachometer* sebesar 23% dari lebar *parent element.*
5. **bottom: 5%;**: Properti ini mengatur jarak elemen *tachometer* dari bagian bawah (*bottom*) sebesar 5% dari tinggi *parent element*.
6. **z-index: 2000;**: Properti ini mengatur tingkat lapisan (z-index) elemen *tachometer*. Semakin tinggi nilai z-index, semakin tinggi elemen akan berada dalam lapisan tampilan.
7. **color: #e62e13;**: Properti ini mengatur warna teks pada elemen *tachometer* menjadi merah (#e62e13).
8. **text-shadow**: Properti ini menambahkan efek bayangan teks untuk elemen *tachometer* dengan menggunakan nilai *offset* dan warna tertentu.
9. **letter-spacing: 2px;**: Properti ini mengatur jarak antar huruf (*letter-spacing*) menjadi 2px.
10. **font-size: 23px;**: Properti ini mengatur ukuran teks pada elemen *tachometer* menjadi 23px.

Selain itu, terdapat juga *pseudo-element* **::after** yang digunakan untuk menambahkan teks "km/h" setelah nilai *tachometer*. *Pseudo-element* ini didefinisikan dengan menggunakan **content**, **color**, **font-size**, dan **margin-left** untuk mengatur tampilan teks "km/h".

Secara keseluruhan, kode CSS di atas bertanggung jawab untuk menampilkan elemen *tachometer* pada permainan *Rush Hours* dengan tampilan visual yang diatur dengan baik, termasuk warna, ukuran, dan posisi tachometer dalam permainan.

```
    inGame = false;
} else {
    time.innerText = (countDown | 0).pad(3);
    score.innerText = (scoreVal | 0).pad(8);
    tacho.innerText = speed | 0;
```

Gambar 4.24 Penambahan Fungsi *Tachometer* Pada JavaScript

Pada Gambar 4.24 terdapat kode **tacho.innerText = speed | 0;** digunakan untuk mengatur nilai tachometer pada permainan. *Tachometer* digunakan untuk menampilkan nilai kecepatan kendaraan kepada pemain, kode di atas menggunakan JavaScript untuk mengupdate nilai *tachometer* sesuai dengan nilai kecepatan yang ada. Variabel **speed** merupakan variabel yang menyimpan nilai kecepatan aktual kendaraan dalam permainan.

Pada sisi kanan dari kode, **| 0** digunakan untuk melakukan operasi bitwise OR dengan 0 pada nilai **speed**. Operasi bitwise OR ini secara efektif akan membulatkan nilai kecepatan menjadi bilangan bulat (integer), sehingga menghilangkan bagian desimal dari angka kecepatan. Hal ini biasanya dilakukan karena *tachometer* pada mobil biasanya hanya menampilkan angka bulat untuk kecepatan.

Setelah operasi bitwise OR dilakukan, nilai hasilnya akan diatur sebagai *inner text* dari elemen *tachometer*. Ini berarti angka kecepatan yang sudah dibulatkan akan ditampilkan dalam elemen tachometer pada permainan, sehingga pemain dapat melihat dan memantau kecepatan kendaraan mereka.

4.4.3 Fitur *Switch Lane*

Fitur pindah jalur pada game diperlukan untuk meningkatkan tingkat keseruannya dan memberikan tantangan lebih kepada pemain. Dengan adanya fitur ini, pemain harus memiliki keterampilan mengemudi yang baik dan reaksi yang cepat untuk dapat menghindari kendaraan lawan dan menjaga posisi di jalur yang tepat. Fitur ini juga memberikan elemen strategi dalam bermain, karena pemain harus memilih jalur yang paling optimal untuk mencapai skor tertinggi dan menghindari tabrakan.

Selain itu, fitur pindah jalur juga memberikan variasi dan dinamika pada permainan. Dengan kemampuan berpindah jalur, pemain tidak hanya terpaku pada satu jalur dan melakukan pergerakan yang monoton. Mereka dapat menghadapi rintangan dan kendaraan lawan dari berbagai arah, sehingga permainan menjadi lebih menantang dan seru.

Dalam keseluruhan, fitur pindah jalur menjadi salah satu aspek penting dalam membangun pengalaman bermain yang menyenangkan dan menantang bagi para pemain. Hal ini membuat game *Rush Hours* menjadi lebih menarik, menguji keterampilan pemain, dan menghadirkan pengalaman bermain yang lebih dinamis.

```
playerX -= (lines[startPos].curve / 5000) * step * speed;

if (KEYS.ArrowRight)
  (hero.style.backgroundPosition = "-220px 0"),
    (playerX += 0.007 * step * speed);
else if (KEYS.ArrowLeft)
  (hero.style.backgroundPosition = "0 0"),
    (playerX -= 0.007 * step * speed);
else hero.style.backgroundPosition = "-110px 0";

playerX = playerX.clamp(-3, 3);
```

Gambar 4.25 Penambahan Fitur *Switch Lane*

Pada Gambar 4.25 adalah bagian dari *script* yang mengatur pergerakan pemain (*hero*) pada *game* “Rush Hours”. Ketika pemain menekan tombol panah kanan (KEYS.ArrowRight), *hero* akan dipindahkan ke kanan dengan kecepatan sebanding dengan nilai “speed”. Hal yang sama berlaku ketika pemain menekan tombol panah kiri (KEYS.ArrowLeft), *hero* akan dipindahkan ke kiri dengan kecepatan yang sama.

Untuk memvisualisasikan pergerakan *hero*, kode tersebut menggunakan *sprite sheet* yang berisi tiga *frame* gambar. Ketika pemain bergerak ke kanan, *frame* pertama dari *sprite sheet* (backgroundPosition: “-220px 0”) digunakan untuk menampilkan animasi gerakan ke kanan. Ketika pemain bergerak ke kiri, *frame* kedua dari *sprite sheet* (backgroundPosition: “0 0”) digunakan untuk menampilkan

animasi gerakan ke kiri. Jika pemain tidak bergerak (tidak menekan tombol panah), *frame* ketiga dari *sprite sheet* (backgroundPosition: "-110px 0") digunakan untuk menampilkan animasi *hero* dalam kondisi diam.

Selain itu, terdapat fungsi "clamp" yang digunakan untuk memastikan pemain tidak keluar dari batas jalur yang ditentukan, yaitu antara -3 hingga 3, dengan demikian, pemain tetap terbatas dalam tiga jalur yang ada dan tidak bisa bergerak terlalu ke kiri atau ke kanan melewati batas tersebut, apabila melewati batas tersebut atau keluar jalur maka kecepatan akan turun secara konstan.

4.4.4 Fitur *Speed*

```
if (inGame && KEYS.ArrowUp) speed = accelerate(speed, accel, step);
else if (KEYS.ArrowDown) speed = accelerate(speed, breaking, step);
else speed = accelerate(speed, decel, step);

if (Math.abs(playerX) > 0.55 && speed >= maxOffSpeed) {
  speed = accelerate(speed, offDecel, step);
}

speed = speed.clamp(0, maxSpeed);
```

Gambar 4.26 Penambahan Fitur *Speed*

Pada Gambar 4.26 adalah bagian dari *script* yang mengatur sistem kecepatan pada *game* "Rush Hours". Dalam *game* tersebut, kecepatan pemain (*speed*) dapat diatur berdasarkan tindakan yang dilakukan oleh pemain melalui tombol panah (ArrowUp, ArrowDown) pada saat permainan berlangsung (inGame).

Pertama-tama, kita lihat bahwa kecepatan pemain akan berubah tergantung pada tombol yang ditekan. Jika permainan sedang berlangsung (inGame = true) dan tombol panah ke atas (KEYS.ArrowUp) ditekan, maka kecepatan akan meningkat dengan memanggil fungsi "accelerate" dengan parameter kecepatan saat ini (speed), percepatan (accel), dan langkah waktu (step). Jika tombol panah ke bawah (KEYS.ArrowDown) ditekan, maka kecepatan akan berkurang dengan memanggil fungsi "accelerate" dengan parameter kecepatan saat ini (speed), pengereman (breaking), dan langkah waktu (step).

Jika tidak ada tombol panah yang ditekan, kecepatan akan berkurang dengan memanggil fungsi “accelerate” dengan parameter kecepatan saat ini (speed), perlambatan (decel), dan langkah waktu (step).

Selanjutnya, kode tersebut mengecek apakah nilai absolut dari posisi pemain (playerX) melebihi 0.55 dan kecepatannya (speed) mencapai atau melebihi batas maksimum offSpeed (maxOffSpeed). Jika kedua kondisi tersebut terpenuhi, maka kecepatan akan diperlambat dengan memanggil fungsi “accelerate” dengan parameter kecepatan saat ini (speed), perlambatan ketika offTrack (offDecel), dan langkah waktu (step). Hal ini menggambarkan bahwa ketika pemain menyimpang terlalu jauh dari jalur yang ditentukan (melebihi 0.55), dan kecepatannya terlalu tinggi (melebihi maxOffSpeed), maka kecepatannya akan dikurangi agar pemain dapat dengan lebih mudah mengendalikan kembali pergerakan hero.

Terakhir, kecepatan pemain dibatasi ke dalam rentang 0 hingga maxSpeed menggunakan fungsi “clamp”. Ini berarti kecepatan pemain tidak akan melebihi batas maksimum (maxSpeed) dan tidak bisa bernilai negatif (dalam keadaan diam atau berhenti).

4.4.5 Fitur Algoritma *Collision*

```
 const isCollide = (x1, w1, x2, w2) => (x1 - x2) ** 2 <= (w2 + w1)
** 2;
// collision
          const offsetRatio = 5;
          if (
            (car.pos | 0) === startPos &&
            isCollide(playerX * offsetRatio + LANE.B, 0.5, car.lane,
0.5)
          ) {
            speed = Math.min(hitSpeed, speed);
            if (inGame) audio.play("honk");
          }
        }
```

Potongan kode di atas yang diberikan mengandung fungsi *isCollide* dan sebagian dari perulangan dalam fungsi pembaruan (*update)* dalam *game loop*. Fungsi *isCollide* digunakan untuk mendeteksi tabrakan antara dua objek berdasarkan posisi

dan lebar masing-masing objek. Fungsi ini menghitung jarak horizontal antara dua objek (dinyatakan oleh **x1** dan **x2**) dan membandingkannya dengan jumlah jarak lebar kedua objek (dinyatakan oleh **w1** dan **w2**). Jika jarak horizontal kuadrat kurang dari atau sama dengan jumlah jarak lebar kuadrat, maka fungsi ini mengembalikan *true*, menandakan terjadinya tabrakan.

Dalam perulangan di dalam fungsi pembaruan permainan, kode ini menerapkan logika deteksi tabrakan antara mobil pemain dan mobil-mobil lainnya. Setiap mobil dalam *array cars* diperiksa untuk tabrakan dengan mobil pemain. Ini dilakukan dengan menghitung posisi horizontal aktual mobil pemain, yang dihitung berdasarkan posisi jalur (*lane*) dan offsetnya (**playerX * offsetRatio + LANE.B**), dan membandingkannya dengan posisi dan lebar mobil lain.

Jika terjadi tabrakan, maka kecepatan mobil pemain akan dikurangi menjadi kecepatan tabrakan (*hitSpeed*) atau kecepatan yang lebih rendah, dan jika permainan sedang berlangsung (*inGame* bernilai *true*), suara klakson akan diputar menggunakan objek *audio*.

Secara keseluruhan, kode ini bertanggung jawab atas deteksi tabrakan antara mobil pemain dan mobil-mobil lainnya dalam permainan. Jika terdeteksi tabrakan, kecepatan mobil pemain akan dikurangi, menciptakan efek tabrakan, dan klakson akan bunyi sebagai umpan balik visual dan auditori kepada pemain.

4.4.6 Fitur Respawn

```
// respawn
if ((car.pos | 0) === endPos) {
  if (speed < 30) car.pos = startPos;
  else car.pos = endPos - 2;
  car.lane = randomProperty(LANE);
}
```

Gambar 4.27 Penambahan Fitur *Respawn*

Pada Gambar 4.27 adalah potongan kode untuk suatu permainan atau simulasi yang melibatkan kendaraan di jalan. Di dalam program tersebut, terdapat dua bagian utama, yaitu "respawn" dan "collision".

Bagian "respawn" berfungsi untuk mengatur kembali kemunculan kendaraan jika sudah mencapai ujung jalanan. Jika kendaraan berjalan dengan kecepatan lambat, maka kendaraan akan muncul kembali di awal jalanan. Namun, jika kendaraan berjalan dengan kecepatan cepat, maka kendaraan akan muncul dua langkah sebelum ujung jalanan. Selain itu, jalur kendaraan juga akan dipilih secara acak.

4.4.7 Fitur *Highscore*

```css
#highscore {
  position: absolute;
  width: 100%;
  height: 20%;
  bottom: 0;
  column-count: 3;
  column-fill: auto;
}

#highscore * {
  color: #9e95a8;
  margin: 0 0 6px 27px;
}
```

Gambar 4.28 Penambahan Fitur *Highscore* pada CSS

Pada Gambar 4.28 terdapat bagian CSS, elemen div dengan ID "highscore" diberikan gaya tampilan dengan posisi absolut di bagian bawah halaman dan ditampilkan dalam tiga kolom. Elemen-elemen di dalam div "highscore" akan memiliki warna teks #9e95a8 dan margin yang memberikan jarak pada sisi kiri dan bawah.

```javascript
function updateHighscore() {
  let hN = Math.min(12, highscores.length);
  for (let i = 0; i < hN; i++) {
    highscore.children[i].innerHTML = `${(i + 1).pad(2, " ")}. ${
      highscores[i]
    }`;
  }
}
```

Gambar 4.29 Penambahan Fitur *Highscore* pada JavaScript

Pada Gambar 4.29 terdapat kode JavaScript berisi fungsi **updateHighscore()** yang berperan dalam memperbarui data *highscore*. Fungsi ini akan mengambil data skor tertinggi dari variabel **highscores** dan mengubah tampilan setiap elemen di dalam

div "highscore" sesuai dengan data yang ada. Data *highscore* akan ditampilkan dengan format nomor urut dan nilai skor.

Dengan implementasi kode tersebut, permainan "Rush Hours" memiliki tampilan yang menarik untuk daftar highscore dan secara dinamis memperbarui data tersebut setiap kali ada perubahan. Hal ini memberikan pengalaman bermain yang lebih menantang karena pemain dapat selalu berusaha untuk mencatatkan skor tertinggi dan bersaing dengan pemain lain untuk mendapatkan peringkat teratas.

4.4.8 Fitur Sound

```
  AUDIO: {
    theme:
      "https://s3-us-west-2.amazonaws.com/s.cdpn.io/155629/theme.mp3",
    engine:
      "https://s3-us-west-2.amazonaws.com/s.cdpn.io/155629/engine.wav",
    honk: "https://s3-us-west-2.amazonaws.com/s.cdpn.io/155629/honk.wav",
    beep: "https://s3-us-west-2.amazonaws.com/s.cdpn.io/155629/beep.wav",
  },
};
```

Gambar 4.30 Penambahan Fitur Sound

Pada Gambar 4.30 terdapat beberapa *audio* yang digunakan untuk menjadi *sound* pada *game*, masing masing *audio* memiliki fungsinya masing masing, seperti *theme* yang berfungsi untuk menjadi lagu pengiring saat *game* dimainkan.

Kemudian *engine* yang digunakan untuk menjadi suara mobil yang digunakan, lalu ada *honk* atau suara klakson yang digunakan sebagai indikator saat terjadi tabrakan, dan yang terakhir adalah *beep* yang digunakan sebagai suara saat hitung mundur dilakukan.

```
addEventListener(`keyup`, function (e) {
  if (e.code === "KeyM") {
    e.preventDefault();

    audio.volume = audio.volume === 0 ? 1 : 0;
    return;
  }
```

Gambar 4.31 Penambahan Fungsi Mute

Kemudian ditambahkan program, seperti pada Gambar 4.31 dengan tujuan untuk memberikan fungsi *mute* pada permainan.

4.4.9 Fitur Help

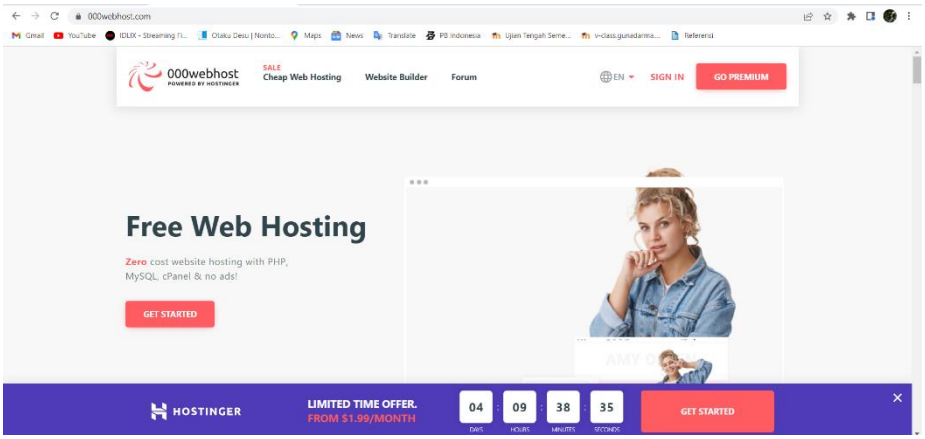
```
addEventListener('keyup', function(e) {
  if (e.code === 'KeyH') {
    e.preventDefault();
    alert(
      "Tutorial Bermain:\n\n" +
      "C: Kunjungi Instagram Developer\n\n"+
      "M: Mute/Unmute Suara\n\n" +
      "Space: Mulai Permainan\n\n" +
      "Panah Kiri: Pindahkan Kendaraan ke Kiri\n\n" +
      "Panah Kanan: Pindahkan Kendaraan ke Kanan\n\n" +
      "Panah Atas: Percepat Kendaraan\n\n" +
      "Panah Bawah: Mengurangi Kecepatan Kendaraan\n\n"


    );
  }
});
```

Gambar 4.32 Penambahan Fitur Help

Pada Gambar 4.32 adalah penambahan program *help* yang berisi tata cara memainkan *game* tersebut.

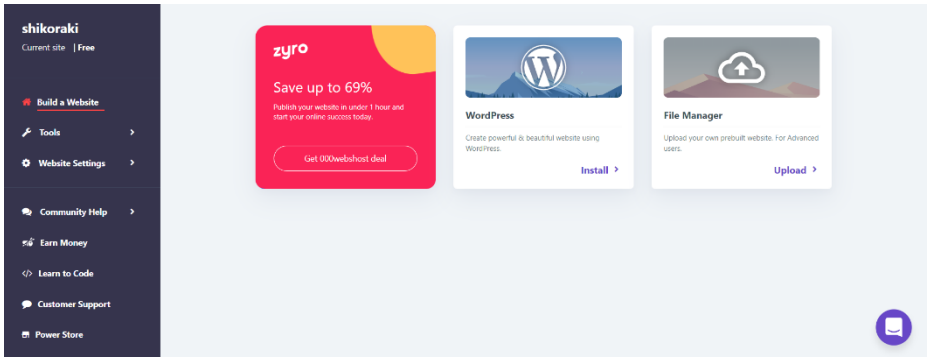
4.5 *Hosting*

Proses *hosting* dilakukan agar *game* yang digunakan dapat terintegrasi dengan *internet*, berikut adalah langkah-langkah hosting *game* berbasis *website*, seperti pada Gambar 4.33.



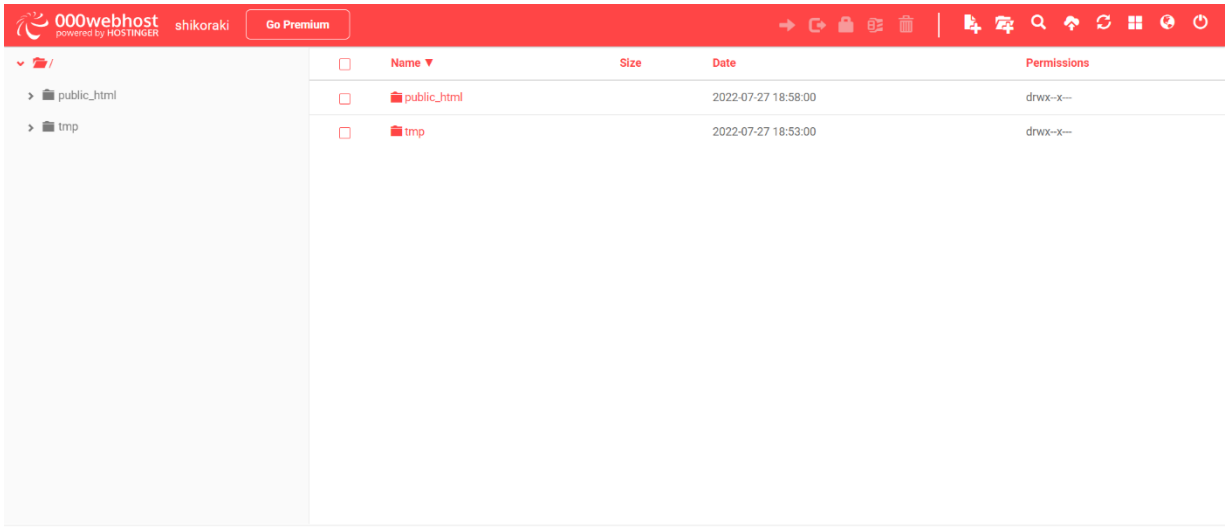
Gambar 4.33 Halaman Awal *Hosting*

Pertama, buka laman https//000webhost.com kemudian klik *sign in*, selesaikan semua proses pendaftaran hingga bagian membuat nama domain, seperti pada Gambar 4.34.



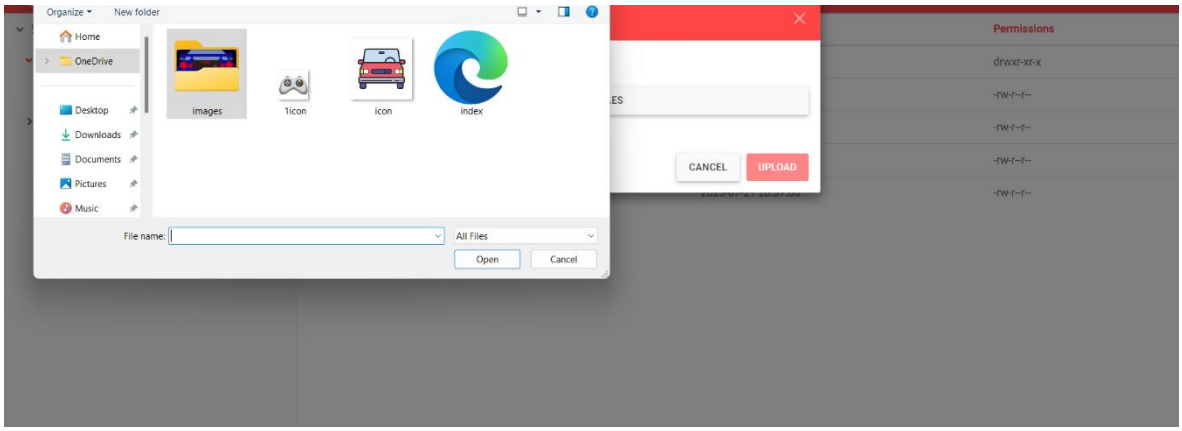
Gambar 4.34 Halaman *Dashboard Hosting*

Setelah proses pendaftaran selesai maka akan dilarikan ke laman *dashboard*, setelah itu klik *file manager*, seperti pada Gambar 4.35.



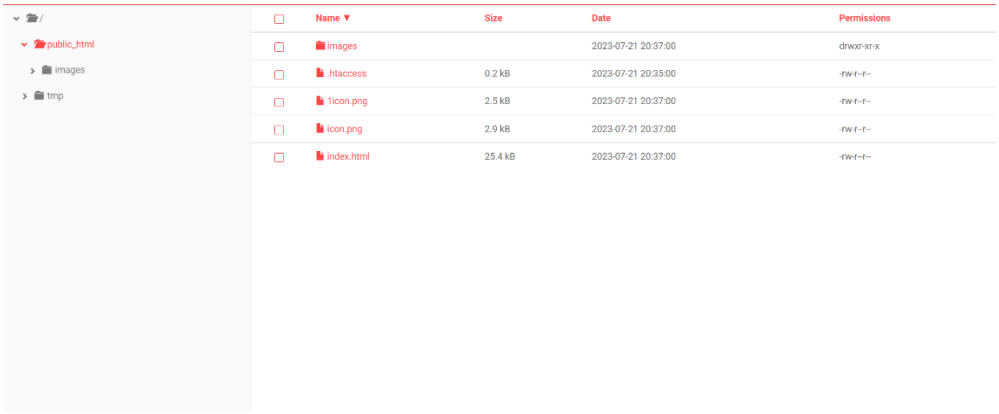
Gambar 4.35 Halaman *File Manager*

Pada laman *file manager* klik folder “public_html” kemudian pilih tombol *upload*, seperti pada Gambar 4.36.



Gambar 4.36 Proses *Upload File*

Pilih *file* yang akan di*upload* kemudian tekan *open*, maka *file* yang dipilih akan ter unggah, seperti pada Gambar 4.37.



Gambar 4.37 Halaman *File Manager*

Pada Gambar 4.37 adalah *file-file* yang telah ter*upload* pada *file manager*, setelah semua selesai, maka *website* siap digunakan.

Gambar 4.38 Halaman *Game* Setelah *Hosting*

Pada Gambar 4.38 adalah halaman *game* setelah *hosting*. Proses *hosting* sudah selesai, *website* berhasil terintegrasi dengan *internet* dengan domain yang bernama:

https://rushhours.000webhostapp.com/

4.6 Implementasi Uji Coba

Tahap uji coba dilakukan agar aplikasi yang dibuat dapat berjalan dengan baik, uji coba yang dilakukan dengan melalukan pengecekan apakah sistem mengeluarkan *output* yang sesuai dengan yang diharapkan.

4.6.1 Uji Coba Halaman Utama

Uji coba pertama dilakukan pada halaman utama, uji coba ini dilakukan untuk melihat apakah output yang dikeluarkan sudah sesuai dengan yang diharapkan, seperti pada Tabel 4.1.

Tabel 4.1 Tabel Uji Coba Halaman Utama

Uji Coba	**Hasil Uji Coba**	**Output**	**Kesimpulan**
Halaman Utama	Menampilkan halaman awal, judul *game*, tutorial dan *credit*	Sistem dapat menampilkan semua isi pada halaman utama *game*	Berhasil
Tutorial	Menampilkan tombol – tombol yang digunakan untuk bermain	Sistem dapat menampilkan tombol yang dapat digunakan untuk bermain	Berhasil
Credit	Menampilkan nama laman Instagram penulis apabila menekan tombol credit	Sistem dapat menampilkan credit pada halaman utama	Berhasil
Mute	Mematikan *sound*	*Sound* pada *game* dapat	Berhasil

	pada *game*	dimatikan oleh sistem.	

4.6.2 Uji Coba Halaman Lane Game

Uji coba kedua dilakukan pada halaman *Lane Game*, uji coba ini dilakukan untuk melihat apakah *output* yang dikeluarkan sudah sesuai dengan yang diharapkan, seperti pada Tabel 4.2.

Tabel 4.2 Tabel Uji Coba Halaman *Lane Game*

Uji Coba	**Hasil Uji Coba**	**Output**	**Kesimpulan**
Lane game	Menampilkan jalur mobil	Sistem dapat menampilkan jalur mobil	Berhasil
Menampilkan Hero (User)	Menampilkan mobil yang digunakan user	Sistem dapat menampilkan Hero	Berhasil
Menampilkan mobil lawan	Menampilkan mobil lawan	Sistem dapat menampilkan mobil lawan	Berhasil
Menampilkan background awan	Menampilkan latar pada langit lane game	Sistem dapat menampilkan background langit pada game	Berhasil

Uji Coba	**Hasil Uji Coba**	**Output**	**Kesimpulan**
Menampilkan asset pohon	Menampilkan pohon sebagai aksesoris pada pinggir lane	Sistem dapat menampilkan asset pohon	Berhasil
Menampilkan asset garis ginish	Menampilkan garis finish	Sistem dapat menampilkan garis finish saat permainan selesai	Berhasil
Melakukan switch lane	Melakukan perpindahan jalur menggunakan tombol arah kiri dan kanan	Sistem dapat menampilkan fungsi switch lane untuk berpindah jalur pada game	Berhasil
Menambah dan mengurangi kecepatan	Dapat menambah dan mengurangi kecepatan sesuai dengan tombolnya	Sistem dapat menampilkan fungsi untuk menambah dan mengurangi kecepatan	Berhasil

Pada Tabel 4.2, hasil uji coba yang dilakukan mengeluarkan *output* yang diharapkan penulis.

4.6.3 Uji Coba *Highscore*

Uji coba terakhir dilakukan pada halaman *highscore*, uji coba ini dilakukan untuk melihat apakah *output* yang dikeluarkan sudah sesuai dengan yang diharapkan

Tabel.4.3 Tabel Uji Coba Highscore

Uji Coba	Hasil Uji Coba	Output	Kesimpulan
Mencatat highscore	Mencatat perolehan waktu, skor, dan lap (selisih waktu)	Sistem dapat mencatat perolehan waktu, skor, dan lap (selisih waktu)	Berhasil
Menyusun highscore	Menyusun perolehan waktu yang telah dicatat berdasarkan yang paling cepat	Sistem dapat menyusun perolehan waktu yang telah dicatat berdasarkan yang paling cepat	Berhasil
Menghapus perolehan waktu	Menghapus perolehan waktu yang telah dicatat	Sistem dapat menghapus perolehan waktu apabila melebih kapasitas list	Berhasil

Pada Tabel 4.3, hasil uji coba yang dilakukan mengeluarkan output yang sesuai, kesimpulan dari uji coba diatas adalah berhasil.

4.6.4 Uji Coba *Browser*

Tahap uji coba *browser* bertujuan untuk menilai apakah aplikasi berfungsi di berbagai *browser*, dan melihat berapa lama *browser* untuk merespon.

Tabel 4.4 Tabel Uji Coba *Browser*

Browser	**Hasil**
Mozilla Firefox	Membutuhkan waktu kurang lebih 0,88 detik untuk akses setiap halaman
Google Chrome	Membutuhkan waktu kurang lebih 1,04 detik untuk akses setiap halaman
Microsoft Edge	Membutuhkan waktu kurang lebih 0,98 detik untuk akses setiap halaman

Tahap uji coba pada Tabel 4.4 menunjukkan bahwa aplikasi bekerja dengan baik di berbagai *browser*, hanya saja menunjukan sedikit perbedaan respon pada beberapa *browser*.

BAB 5
PENUTUP

5.1 Kesimpulan

Berdasarkan uji coba yang telah dilakukan didapat bahwa semua fungsi dapat berfungsi dengan baik, sesuai dengan yang diharapkan. *Game* ini diharapkan dapat menjadi sarana hiburan, karena dapat dimainkan sendiri dan menjadi sarana melatih fokus dan memecahkan masalah.

Penggunaan Algoritma *Collision* dalam permainan dapat meningkatkan pengalaman bermain dengan mendeteksi tabrakan antara objek-objek secara akurat, menciptakan respons visual yang realistis dan interaksi yang sesuai aturan.

Dalam *game "Rush Hours",* potensi kompetitif dapat dicapai melalui sistem penilaian dan peringkat berdasarkan kecepatan dan ketepatan pemain dalam menyelesaikan level, mendorong pemain untuk berkompetisi dalam mencapai peringkat tertinggi dan memperpanjang daya tarik permainan https://rushhours.000webhostapp.com/

5.2 Saran

Dalam pengembangan *game "Rush Hours",* terdapat potensi peningkatan yang dapat meningkatkan pengalaman bermain:

1. **Tingkat Kesulitan**: Penambahan level kesulitan akan memberikan tantangan lebih kompleks, mendorong pemain untuk meningkatkan keterampilan dan strategi.
2. **Peningkatan Visual**: Grafis yang ditingkatkan dan animasi yang lebih halus akan menciptakan daya tarik visual dan imersi yang lebih baik.
3. **Pengayaan Konten**: Penambahan kendaraan, rute, dan fitur khusus memberikan variasi dan kejutan dalam permainan.

Dengan langkah-langkah ini, *game "Rush Hours"* dapat menjadi permainan yang lebih kompetitif dan menghibur.

DAFTAR PUSTAKA

Abiyit, Iruhan Zain, Firhan (2019). "Indonesian Game Racing: Game Edukasi 3D Permainan Tradisional Indonesia Dengan Menggunakan Unity Berbasis Android (Pemrograman)." Tesis Diploma, Politeknik Negeri Jember. URL: https://sipora.polije.ac.id/23380/.

Antonio, Roberto, Jeanny Pragantha, dan Darius Andana Haris. "Pembuatan Game Arcade 2D 'WEAPOWIZE' Berbasis Android." Teknik Informatika, Fakultas Teknologi Informasi, Universitas Tarumanagara, Jakarta 11440, Indonesia. https://doi.org/10.24912/jiksi.v8i1.11488.

Arrahman, M. M. A., & David. (2022). Penerapan Collision Detection Pada Game Platformer "Culture Seeker". Jurnal Sistem Informasi dan Teknologi Informasi, 11(1), 101. Jurusan Teknik Informatika, STMIK Pontianak.

Dicoding. (2019). Contoh Use Case Diagram Lengkap Dengan Penjelasannya. Retrieved July 19, 2022, from https://www.dicoding.com/blog/contoh-use-case-diagram/.

Fauziah, E. M. P. (2021). Rancang Bangun Game Getuk Shooter Menggunakan Algoritma Collision Detection Berbasis Android. ISSN: 2614-5448.

Fasha, L.H., Fauziah, M., & Gufroni, M. (2018). Implementasi Algoritma Collision Detection pada Game Simulator Driving Car. Jurnal String, 3(1), 58. p-ISSN: 2527 - 9661, e-ISSN: 2549 -2837.

Haryono, A. N., Hendro, S., & Setiawan. (2010). penggunaan struktur data Quad Tree dalam Algoritma Collision Detection pada Vertical Shooter Game penggunaan struktur data Quad-Tree dalam Algoritma Collision Detection pada Vertical Shooter Game penggunaan struktur data Quad-Tree dalam Algoritma Collision.

Hamanako, Gugah Alwan, dan Adi Sucipto. "Implementation of Collision Avoidance System Algorithm in NPC Game 3D." Journal of Information Technology and Its Utilization, vol. 6, no. 1, June 2023, hal. 39. EISSN 2654-802X.

Jaya, Tri Sandhika. (2018). "Pengujian Aplikasi dengan Metode Blackbox Testing Boundary Value Analysis (Studi Kasus: Kantor Digital Politeknik

Negeri Lampung)." Jurnal Tri Sandhika Jaya, vol. 45. ISSN: 2477-5126. e-ISSN: 2548-9356.

Kamal, Luthfi, Lina, dan Darius Andana Haris. "Pembuatan Game Simulasi 'SAFETY WAY OUT'." Mahasiswa Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Tarumanagara, Jakarta 11440, Indonesia. https://doi.org/10.24912/jiksi.v6i2.2646.

Naufal, Mohammad Farid (2018). "Analisa Teknik Pembelajaran dan Pengajaran Pemrograman pada Universitas dan Industri." Jurnal Informatika & Multimedia, vol. 10, no. 2, Tahun 2018, hal. 1. pISSN: 2252–486X, eISSN: 2548–4710. URL: http://repository.ubaya.ac.id/34132/1/%5B1%5D%20Jurnal%20Nasional.pdf.

Nurdiansyah, Ilham, Eka Wahyu Hidayat, dan Andi Nur Rachman. "Penerapan Metode Collision Detection pada Game Platformer Mr. Hoax." ISSN: 2621-1416.

Musfiroh, L., Jazuli, A., & Latubessy, A. (2014). Penerapan Algoritma Collision Detection dan Boids pada Game Dokkaebi Shooter. Prosiding SNATIF Ke-1 Tahun 2014. ISBN: 978-602-1180-04-4. Fakultas Teknik, Universitas Muria Kudus.

Repository.bsi.ac.id (2018). Daftar Simbol. Diakses pada 19 Juli 2022, dari https://repository.bsi.ac.id/index.php/unduh/item/205310/File_9-Daftar-Simbol.pdf

Santoso, Prasetyo, dan Siti Madinah Ladjamuddin. "Rancang Bangun Game First Person Shooter Berburu Monster di Tengah Hutan dengan Unity 5.6.3." Incomtech, vol. 9, no. 2, Desember 2020, ISSN 2337-6805. Program Studi Teknik Informatika, Fakultas Sains dan Teknologi Informasi, Institut Sains dan Teknologi Nasional. URL: https://ejournal.istn.ac.id/index.php/incomtech/article/view/913/651.

Sulistyanto Laili R & Dodik Arwin Dermawan. (2018). Implementasi Algoritma Collision Detection dan Markov Chain untuk Menentukan Behaviour NPC dan Karakter Player pada Game Higeia. ISSN: 2686-2220.

Taurusta, Cindy, dan Yulian Findawati. "Rancang Bangun Game Algoritma dan Struktur Data Berbasis Role Playing Game (RPG) Sebagai Media Pembelajaran Mahasiswa Teknik Informatika Universitas Muhammadiyah Sidoarjo." Kinetik, vol. 2, no. 3, Agustus 2017, hal. 175-188. ISSN 2503-2259. E-ISSN 2503-2267.

Wiwit Mararizki. (2023). Perancangan dan Pembuatan Game "JUMP CHICKEN" Berbasis Android. ISSN: 2808-5027.