**BAB IV**

**DESAIN SISTEM**

Bab ini akan dijelaskan tentang tahapan pembuatan desain yang dilakukan selama pembentukan game. Berdasarkan dari konsep atau teori yang telah dibahas sebelumnya. Penulis menerapkannya kedalam game melalui Desain Arsitektural, Desain Interface, dan Desain Procedural berbentuk algoritma pada game yang dibuat.

1. **Desain Arsitektural**

Pada desain arsitektural digambarkan dengan beberapa modul-modul. Berikut merupakan gambar desain arsitektural pada game yang dibuat.



Gambar 4.1

Desain Arsitektural Pengolahan Rintangan Oleh DDA

Gambar 4.1 merupakan desain arsitektural dari game. Pada saat level dijalankan emotion detector memberikan data emosi yang terdeteksi selama rintangan dilewati oleh karakter pemain. Setelah level telah berhasil diselesaikan oleh pemain, data yang dicatat oleh pengolah log akan dikirim sebagai input pengolahan rintangan oleh DDA.

Didalam proses pengolahan rintangan, DDA membaca jenis rintangan yang muncul dengan atribut setiap masing-masing jenis. Kemudian skor pemain dibandingkan dengan data weight clipping yang sudah dibentuk. Data weight clipping mengatur intensitas spawn rintangan pada level berikutnya.

Setelah pengolahan data selesai dilakukan, dilakukan penulisan log rintangan yang telah diubah dan kemudian dilakukan pembacaan pada skrip level generator. Sebelum level berikutnya dimainkan, level dibentuk terlebih dahulu dengan suplai data yang telah diolah DDA. Kemudian proses tersebut diulang dari level 2 hingga semua level terselesaikan.

1. **Skenario Game**

Skenario pada game yaitu di hutan Gunung Lali Jiwo Arjuno, Jawa Timur yang ditandai pembentukan level. Karakter yang dimainkan harus berjalan melewati rintangan serta menghindari kejaran hantu yang terus mengancam pemain. Bentuk level akan berubah setiap kali dimainkan. Rintangan yang muncul akan diubah sesuai dengan log performa pemain. Perubahan mempengaruhi rate spawn rintangan dan tempat berlindung. Rate diatur berdasarkan nilai *weight clipping* rintangan yang sudah ditetapkan oleh penulis sehingga sedari awal sudah terbentuk konsep rintangan dengan tingkatan mudah, sedang, dan susah.

Karena level yang selalu berubah, start dan finish level pada game ini dilakukan dengan berpatokan checkpoint. Level 1 dimulai setelah karakter menyelesaikan level tutorial. Pada level 1 karakter akan *spawn* di hutan dengan finishnya pemain harus mencapai ujung kanan level. Pada level 2 dan selanjutnya, garis startnya berada di hutan melanjut dengan garis finish yang ditandai dengan poin merah di ujung kanan level.

Pada setiap level terdapat area aman atau pengusir hantu yang ditandai dengan pohon bambu kuning. Terdapat juga spawn item pengusir hantu pada beberapa tempat pada level. Selain itu juga terdapat berbagai jenis rintangan, dan platform naik turun dimana pemain dituntut untuk lihai dan diharapkan beruntung karena beberapa jenis rintangan melibatkan faktor keberuntungan pemain.

Permainan akan berlangsung hingga waktu habis. Pemain akan mendapat ending baik ketika berhasil melewati 3 level atau lebih hingga waktu yang ditentukan habis. Pemain akan mendapat ending buruk jika sisa nyawa yang diberikan habis dan gagal menyelesaikan 3 level setelah waktu habis.

1. **Arsitektur Pengaturan Rintangan Menggunakan DDA**

Dalam penggunaannya untuk mengatur rintangan, DDA pada game ini terbagi atas 2 tahap yaitu tahap processing dan obstacle generation.

1. TAHAP PROCESSING

Tahap *processing* melingkupi proses pengolahan data dari seluruh parameter yang sudah ditangkap dan dilakukan pencatatannya menjadi log. Dimulai dari pembacaan masing-masing parameter, melakukan scoring, dan kemudian membandingkannya dengan data weight clipping[[1]](#footnote-1). Berdasarkan dari parameter yang ditangkap pada setiap masing-masing jenis rintangan, DDA akan melakukan skoring pada performa pemain.

1. SKORING DDA

Skoring yang dilakukan akan dijabarkan dibawah ini. Skoring dibagi menjadi 2 tahapan. Tahapan pertama yaitu pelabelan performa pemain kemudian dilakukan perhitungan skor emosi pemain rintangan. Tahapan diterapkan pada masing-masing kategori rintangan.

Pada tahapan pelabelan performa pemain, digunakan rumus yang telah dipakai didalam penelitian oleh Missura, O pada 2015 yang berjudul *Dynamic difficulty adjustment*[[2]](#footnote-2). Rumus disesuaikan dengan beberapa parameter yang diubah pada game sehingga menjadi model sebagai berikut :

1. Pemain dengan performa “Mahir”

Rumus dibawah ini digunakan untuk melabeli pemain mahir dengan cara menghindari semua rintangan, rumus yang digunakan yaitu jika rintangan kena sama dengan 0 maka pemain akan diberi label “Mahir”.

(4.1)

Sebagai simulasi pemain A telah menyelesaikan level tanpa kegagalan dengan tidak terkena rintangan yang ada, maka performa yang diberikan oleh sistem DDA kepada pemain yaitu ‘Mahir’. Namun ketika pemain B telah menyelesaikan level dengan satu atau lebih kegagalan pada rintangan yang terdapat pada level, maka akan dilakukan perhitungan performa pemain dengan rumus berikutnya.

Rumus dibawah ini digunakan untuk melabeli pemain mahir dengan cara menghitung persentase kesuksesan dalam melewati rintangan, jika lebih besar dibandingkan dengan jumlah kegagalannya dan memiliki persentase kesuksesan diatas 75 maka akan diberi label “Mahir”.

(4.2)

Sistem DDA mencatat counter kesuksesan pemain dalam melewati rintangan yang ditampung dalam variable ctr\_sukses pada masing-masing jenis rintangan, jumlah rintangan kategori tertentu juga akan dicatat yang ditampung pada variable jum\_jalan, dan yang terakhir dilakukan pencatatan counter pemain gagal dalam melewati jenis rintangan tertentu. Untuk mensimulasikan setiap kemungkinan yang mungkin terjadi pada rumus diatas, maka digunakan tabel simulasi dibawah ini.

**Tabel 4.1**

**Tabel Simulasi Perhitungan Performa Mahir**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Ctr\_sukses** | **Ctr\_gagal** | **Jum\_jalan** | **Perhitungan** | **Keterangan** |
| 1 | 5 | 0 | 5 | (1×100)>(0×100)∧  (1×100)>75 | Mahir. |
| 2 | 4 | 1 | 5 | (0.8×100)>(0.2×100)∧  (0.8×100)>75 | Mahir. |
| 3 | 3 | 2 | 5 | (0.6×100)>(0.4×100)∧  (0.6×100)>75 | Tidak memenuhi syarat kedua. |
| 4 | 2 | 3 | 5 | (0.4×100)>(0.6×100)∧  (0.4×100)>75 | Tidak memenuhi kedua syarat. |
| 5 | 1 | 4 | 5 | (0.2×100)>(0.8×100)∧  (0.2×100)>75 | Tidak memenuhi kedua syarat. |
| 6 | 0 | 5 | 5 | (0×100)>(1×100)∧  (0×100)>75 | Tidak memenuhi kedua syarat. |

Pada contoh nomer 1 dan 2 didapati bahwa syarat rumus yang pertama yaitu nilai persentase kesuksesan lebih besar daripada nilai kegagalan dan nilai kesuksesan diatas range 75 terpenuhi semua. Maka DDA akan memberi skor pemain “Mahir”. Sedangkan pada contoh nomer 3 hingga nomer 6 memiliki keterangan tidak memenuhi syarat. Pada contoh nomer 3, syarat pertama terpenuhi namun persentase kesuksesan yang didapat berada dibawah 75. Sedangkan pada contoh nomer 4 hingga nomer 6 kedua syarat yang diberikan tidak dapat dipenuhi. Apabila kondisi tidak memenuhi syarat, maka akan dilakukan perhitungan menggunakan rumus berikutnya yaitu label “Normal”.

1. Pemain dengan performa “Normal”

Rumus dibawah ini digunakan untuk melabeli pemain dengan cara menghitung persentase kesuksesan dalam melewati rintangan. Jika nilai persentase kesuksesan yang didapat lebih besar dibandingkan dengan persentase kegagalannya dan memiliki persentase kesuksesan dibawah 75 maka akan diberi label “Normal”.

(4.3)

Sistem DDA mencatat counter kesuksesan pemain dalam melewati rintangan yang ditampung dalam variable ctr\_sukses pada masing-masing jenis rintangan, jumlah rintangan kategori tertentu juga akan dicatat yang ditampung pada variable jum\_jalan, dan yang terakhir dilakukan pencatatan counter pemain gagal dalam melewati jenis rintangan tertentu. Untuk mensimulasikan setiap kemungkinan yang mungkin terjadi pada rumus diatas, maka digunakan tabel simulasi dibawah ini.

**Tabel 4.2**

**Tabel Simulasi Perhitungan Performa Normal 1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Ctr\_sukses** | **Ctr\_gagal** | **Jum\_jalan** | **Perhitungan** | **Keterangan** |
| 1 | 5 | 0 | 5 | (1×100)>(0×100)∧  (1×100)<75 | Tidak memenuhi syarat kedua. |
| 2 | 4 | 1 | 5 | (0.8×100)>(0.2×100)∧  (0.8×100)<75 | Tidak memenuhi syarat kedua. |
| 3 | 3 | 2 | 5 | (0.6×100)>(0.4×100)∧  (0.6×100)<75 | Normal |
| 4 | 2 | 3 | 5 | (0.4×100)>(0.6×100)∧  (0.4×100)<75 | Tidak memenuhi syarat pertama. |
| 5 | 1 | 4 | 5 | (0.2×100)>(0.8×100)∧  (0.2×100)<75 | Tidak memenuhi syarat pertama. |
| 6 | 0 | 5 | 5 | (0×100)>(1×100)∧  (0×100)<75 | Tidak memenuhi syarat pertama. |

Pada contoh nomer 1 dan 2 didapati bahwa syarat rumus yang kedua yaitu nilai persentase dibawah range 75 tidak terpenuhi. Pada contoh nomer 3 didapati bahwa syarat rumus yang pertama yaitu nilai persentase kesuksesan lebih besar daripada nilai kegagalan dan nilai kesuksesan dibawah range 75 terpenuhi semua sehingga pemain diberi label “Normal”. Pada contoh nomer 4 hingga nomer 6 syarat pertama tidak terpenuhi karena persentase kesuksesan yang didapat lebih kecil daripada persentase kegagalan. Apabila kondisi masuk kedalam tidak memenuhi syarat pertama, maka akan dilakukan perhitungan menggunakan rumus berikutnya.

Rumus dibawah ini digunakan untuk melabeli pemain degan cara menghitung persentase kesuksesan dalam melewati rintangan, jika sama dengan jumlah persentase kegagalannya, maka akan diberi label “Normal”.

(4.4)

Sebagai simulasi pemain A telah menyelesaikan level dengan beberapa kegagalan. Langkah berikutnya DDA akan membandingkan persentase jumlah kegagalan pemain apakah sama dengan persentase jumlah keberhasilan pemain. Jika memenuhi syarat tersebut, maka pemain akan diberi label “Normal”. Jika tidak memenuhi syarat akan dilakukan perhitungan dengan rumus berikutnya.

Rumus dibawah ini digunakan untuk melabeli pemain yang memiliki persentase keberhasilan dalam melewati rintangan lebih kecil dibandingkan dengan jumlah kegagalannya tetapi memiliki persentase kesuksesan diatas 35.

(4.5)

Sistem DDA mencatat counter kesuksesan pemain dalam melewati rintangan yang ditampung dalam variable ctr\_sukses pada masing-masing jenis rintangan, jumlah rintangan kategori tertentu juga akan dicatat yang ditampung pada variable jum\_jalan, dan yang terakhir dilakukan pencatatan counter pemain gagal dalam melewati jenis rintangan tertentu. Untuk mensimulasikan setiap kemungkinan yang mungkin terjadi pada rumus diatas, maka digunakan tabel simulasi dibawah ini.

**Tabel 4.3**

**Tabel Simulasi Perhitungan Performa Normal 2**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Ctr\_sukses** | **Ctr\_gagal** | **Jum\_jalan** | **Perhitungan** | **Keterangan** |
| 1 | 3 | 2 | 5 | (0.6×100)<(0.4×100)∧  (0.6×100)>35 | Tidak memenuhi syarat pertama. |
| 2 | 2 | 3 | 5 | (0.4×100)<(0.6×100)∧  (0.4×100))>35 | Normal |
| 3 | 1 | 4 | 5 | (0.2×100)<(0.8×100)∧  (0.2×100))>35 | Tidak memenuhi syarat kedua. |
| 4 | 0 | 5 | 5 | (0×100)<(1×100)∧  (0×100)>35 | Tidak memenuhi syarat kedua. |

Pada contoh nomer 1 syarat rumus yang pertama yaitu nilai persentase kesuksesan harus lebih rendah daripada persentase kegagalan tidak terpenuhi. Pada contoh nomer 2 didapati bahwa syarat rumus yang pertama yaitu nilai persentase kesuksesan lebih kecil daripada nilai kegagalan dan nilai kesuksesan diatas range 35 terpenuhi semua sehingga pemain diberi label “Normal”. Pada contoh nomer 3 dan nomer 4 syarat kedia tidak terpenuhi, yaitu persentase kesuksesan yang didapat lebih kecil dari range 35. Apabila kondisi masuk kedalam tidak memenuhi syarat kedua, maka akan dilakukan perhitungan rumus untuk menghitung label “Pemula”

1. Pemain dengan performa “Pemula”

Rumus dibawah ini digunakan untuk melabeli ‘Pemula’ pada pemain yang memiliki persentase kesuksesan dibawah 35.

(4.6)

Sistem DDA mencatat counter kesuksesan pemain dalam melewati rintangan yang ditampung dalam variable ctr\_sukses dan jumlah rintangan kategori tertentu yang ditampung pada variable jum\_jalan. Untuk mensimulasikan setiap kemungkinan yang mungkin terjadi pada rumus diatas, maka digunakan tabel simulasi dibawah.

**Tabel 4.4**

**Tabel Simulasi Perhitungan Performa Pemula 1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Ctr\_sukses** | **Ctr\_gagal** | **Jum\_jalan** | **Perhitungan** | **Keterangan** |
| 1 | 3 | 2 | 5 | (0.6×100)<35 | Tidak memenuhi syarat. |
| 2 | 2 | 3 | 5 | (0.4×100)<35 | Tidak memenuhi syarat. |
| 3 | 1 | 4 | 5 | (0.2×100)<35 | Pemula |
| 4 | 0 | 5 | 5 | (0×100)<35 | Pemula |

Pada contoh nomer 1 dan nomer 2 syarat rumus yaitu nilai persentase kesuksesan harus dibawah 35 tidak terpenuhi. Pada contoh nomer 3 dan 4 didapati bahwa syarat rumus perhitungan persentase kesuksesan lebih kecil dari 35 terpenuhi sehingga pemain diberi label “Pemula”.

Rumus digunakan untuk melabeli pemain yang gagal dalam semua rintangan yang diberikan.

(4.7)

Sebagai simulasi pemain A telah menyelesaikan level dengan sengaja mengenai semua rintangan jenis tertentu yang terdapat pada level. Langkah berikutnya DDA akan membandingkan counter kegagalan pemain apakah sama jumlah jalan rintangan yang muncul pada level. Jika hasilnya sama, maka pemain akan diberi label “Pemula”

Setelah tahapan pelabelan kemudian dilakukan perhitungan emosi rintangan pemain. Berikutnya dilakukan tahapan perhitungan emosi rintangan pemain, emosi diberi label sebagai berikut :

**Tabel 4.5**

**Tabel Label Nilai Emosi**

|  |  |  |
| --- | --- | --- |
| **No** | **Label Emosi** | **Nilai** |
| 1 | Takut | 0 |
| 2 | Marah | 1 |
| 3 | Netral | 2 |

Jika membagi nilai 2 yang merupakan range tertinggi emosi yaitu “Netral” dengan 3 jenis emosi yang ditangkap, maka akan didapati range 0.66 sebagai batas masing-masing kategori emosi. Ketika pemain mendapatkan nilai dengan range 0~0.66 maka akan digolongkan sebagai emosi “Takut”, berkelanjutan yaitu range 0.67~1.33 maka akan digolongkan sebagai emosi “Marah”, dan yang terakhir yaitu range 1.34~2 maka akan digolongkan sebagai emosi “Netral”.

Setelah menentukan batas pembagian nilai perkategori emosi, maka langkah selanjutnya dilakukan perhitungan nilai emosi pemain. Terdapat 3 rumus yang akan digunakan yang akan dijabarkan dibawah ini.

1. Rumus perhitungan skor emosi jalan tidak diinjak

Rumus digunakan ketika pemain memiliki nilai rintangan yang tidak diinjak. Nilai emosi rintangan dijumlah dengan jalan tidak diinjak dikali dengan nilai dari label emosi “Netral” kemudian dibagi jumlah jalan yang diinjak untuk mencari rata-rata emosi yang dideteksi.

(4.8)

Simulasinya sebagai berikut, Pemain A berhasil melewati 4 dari 5 rintangan hantu dengan “Takut” yang dibuktikan dengan rata-rata nilai emosi pada range 0,3 yang merupakan range emosi “Takut” dan pemain memiliki 1 rintangan hantu yang tidak diinjak. Dengan rumus diatas dilakukan perhitungan dengan tahapan 0,3+(1x2)/5=0,46. Hasil yang didapat dari perhitungan termasuk dalam range emosi “Takut” sehingga pemain akan diberi label “Takut” pada rintangan jenis hantu.

1. Rumus perhitungan skor emosi jalan dihindari semua

Rumus digunakan ketika pemain menghindari semua rintangan. Nilai emosi rintangan diisi dengan jumlahJalan \* 2 karena 2 merupakan skor emosi normal.

(4.9)

Rumus ini akan digunakan saat pemain menghindari semua rintangan tertentu. Dikarenakan berhasil menghindar maka pemain diberikan skor emosi “Netral”.

1. Rumus perhitungan rata-rata skor emosi

Rumus ini akan digunakan untuk menentukan rata-rata emosi pemain dalam melewati jenis rintangan tertentu dalam 1 level. Nilai ditampung dan dibuat pengecekan dominasi emosi dengan cara menghitung seberapa sering emosi tersebut terjadi.

(4.10)

Jika didapati emosi ‘Netral’ sering keluar, performa akan diubah naik 1 level. Jika didapati emosi ‘Takut’ sering keluar, maka performa akan diturunkan 1 level. Simulasinya sebagai berikut, Pemain A mendapat nilai emosi 12 setelah mengaktifkan 7 rintangan jenis hantu. Maka ketika perhitungan dilakukan dengan rumus diatas, rata-rata emosi yang didapat sebesar 1.7. Karena nilai yang didapat berada pada range 1.34~2, maka pemain akan diberi label emosi “Netral”.

1. DATA WEIGHT CLIPPING

Setelah dilakukan skoring, maka proses berikutnya yaitu membentuk data weight clipping pada masing-masing rintangan. Terdapat 4 data weight clipping yang terdapat pada game ini.

1. Rintangan hantu

Penjelasan pengaturan weight clipping spawn hantu akan dijabarkan melalui table dibawah.

**Tabel 4.6**

**Tabel Pengaturan Weight Clipping DDA Rintangan Spawn Hantu**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Item | Rate Spawn | Label |
| 1 | JumlahJalanHantu | 2 | Mudah |
| 2 | 4 | Sedang |
| 3 | 5 | Susah |

Rintangan hantu memiliki pengaturan batas maksimal rate spawn yang diberi label. Jika performa pemain mendapat skor ‘Mahir’ oleh sistem DDA, maka yang dipilih adalah weight clipping yang susah. Jika performa pemain mendapat skor ‘Sedang’ maka yang dipilih adalah weight clipping yang sedang, dan jika pemain mendapat skor ‘Pemula’ maka akan diturunkan kesulitannya menjadi mudah pada level berikutnya.

1. Rintangan duri

Penjelasan pengaturan rintangan duri akan dijabarkan melalui table dibawah.

**Tabel 4.7**

**Tabel Pengaturan Weight Clipping DDA Rintangan Duri**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Item | Rate Spawn | Label |
| 1 | jumlahJalanJebakan | 3 | Mudah |
| 2 | 4 | Sedang |
| 3 | 6 | Susah |

Rintangan duri memiliki pengaturan batas maksimal rate spawn yang diberi label. Jika performa pemain mendapat skor ‘Mahir’ oleh sistem DDA, maka yang dipilih adalah weight clipping yang susah. Jika performa pemain mendapat skor ‘Sedang’ maka yang dipilih adalah weight clipping yang sedang, dan jika pemain mendapat skor ‘Pemula’ maka akan diturunkan kesulitannya menjadi mudah pada level berikutnya.

1. Tempat bersembunyi

Penjelasan pengaturan tempat bersembunyi akan dijabarkan melalui table dibawah.

**Tabel 4.8**

**Tabel Pengaturan Weight Clipping DDA Tempat Bersembunyi**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Item | Rate Spawn | Label |
| 1 | jumlahJalanBambu | 4 | Mudah |
| 2 | 3 | Sedang |
| 3 | 1~2 | Susah |

Tempat bersembunyi memiliki pengaturan batas maksimal rate spawn yang diberi label. Jika performa pemain mendapat skor ‘Mahir’ oleh sistem DDA, maka yang dipilih adalah weight clipping yang susah. Jika performa pemain mendapat skor ‘Sedang’ maka yang dipilih adalah weight clipping yang sedang, dan jika pemain mendapat skor ‘Pemula’ maka akan diturunkan kesulitannya menjadi mudah.

1. Rintangan jumpscare penampakan

Penjelasan pengaturan rintangan jumpscare penampakan akan dijabarkan melalui table dibawah.

**Tabel 4.9**

**Tabel Pengaturan Weight Clipping DDA**

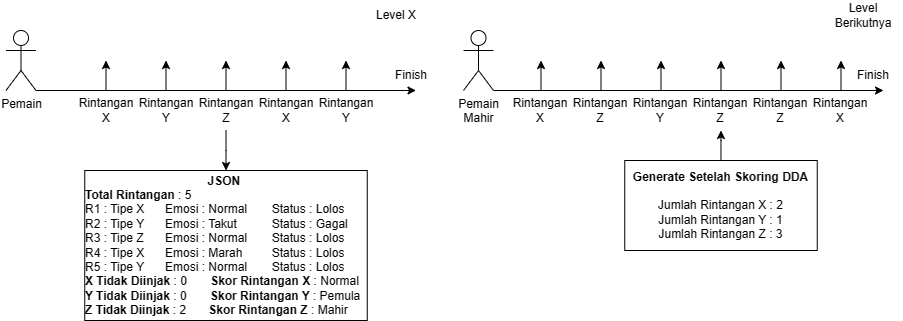
**Rintangan Jumpscare Penampakan**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Item | Rate Spawn | Label |
| 1 | jumlahJalanJumpscare | 2 | Mudah |
| 2 | 4 | Sedang |
| 3 | 8 | Susah |

Rintangan jumpscare penampakan memiliki pengaturan batas maksimal rate spawn yang diberi label. Jika performa pemain mendapat skor ‘Mahir’ oleh sistem DDA, maka yang dipilih adalah weight clipping yang susah. Jika performa pemain mendapat skor ‘Sedang’ maka yang dipilih adalah weight clipping yang sedang, dan jika pemain mendapat skor ‘Pemula’ maka akan diturunkan kesulitannya menjadi mudah pada level berikutnya.

1. TAHAP OBSTACLE GENERATION

Setelah DDA melakukan perancangan rintangan yang pada tahap *processing*, langkah selanjutnya dilakukan pembuatan level berikutnya. Penyesuaian Rintangan yang akan dimunculkan pada level berikutnya akan digambarkan melalui ilustrasi di bawah ini.



Gambar 4.2

Simulasi Pembentukan Rintangan Level

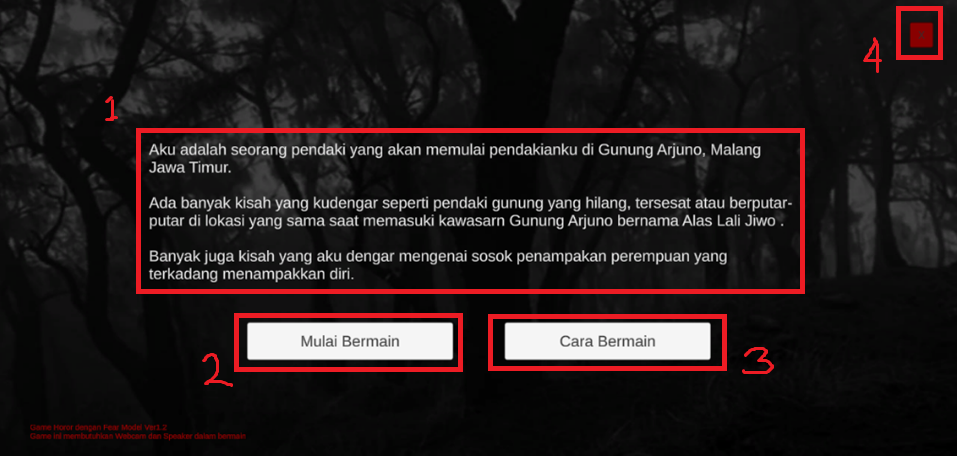
Pada tahap ini content generator akan disuplai data baru yang telah di olah DDA. Rancangan rintangan level berikutnya untuk pemain yang mahir pada rintangan jenis tertentu rintangan tersebut akan dimunculkan semakin banyak. Content generator akan membentuk jalan dengan rintangan berdasarkan instruksi yang telah didapat sehingga menjadi sebuah level. Siklus ini selalu diulang sebanyak level terdapat didalam game. Jadi setiap level yang akan muncul akan selalu berbeda dikarenakan kemunculan rintangan yang dilakukan berdasarkan data performa pemain pada tahapan sebelumnya.

1. **Desain Interface**

Subbab ini menjelaskan desain interface dari tampilan menu utama, narasi penutup, cara bermain, tampilan hud, dan deteksi wajah.

1. Menu Utama

Halaman menu utama akan ditampilkan setiap permainan dibuka, dan setelah pemain telah menekan tombol Main Menu pada narasi penutup. Interface ini digunakan untuk membantu pemain dalam menjelajahi sistem permainan.



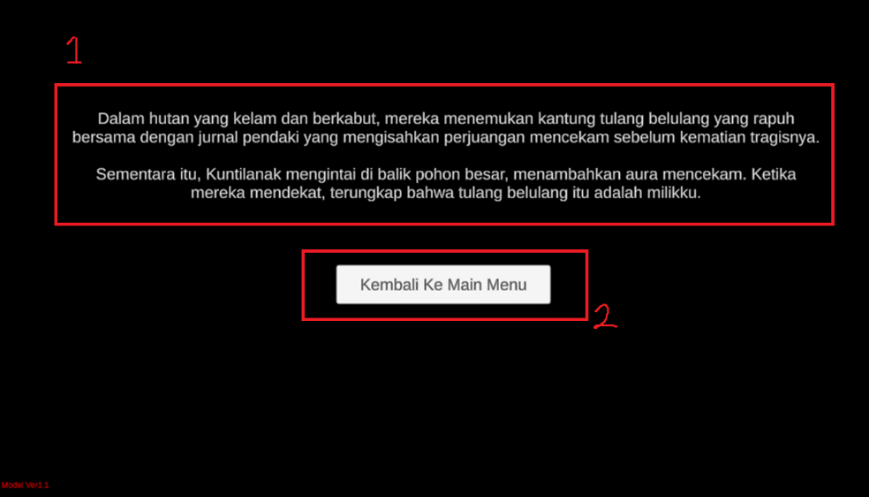
Gambar 4.3

Tampilan Menu Utama

Di halaman menu utama terdapat beberapa komponen penting pada halaman ini yaitu narasi pembuka, tombol mulai bermain, tombol cara bermain, dan tombol exit. Penjelasan lebih lengkapnya diuraikan sebagai berikut:

1. Narasi pembuka seputar cerita karakter didalam game yang tersesat.
2. Tombol mulai bermain, berfungsi untuk memulai permainan.
3. Tombol cara bermain, berfungsi untuk menjelaskan kepada pemain mengenai kontrol pemain dan bagaimana game harus dimainkan.
4. Tombol exit, pada ujung kanan layar berfungsi untuk menutup permainan.
5. Narasi Penutup

Halaman narasi penutup akan ditampilkan setiap pemain berhasil menyelesaikan semua level. Interface ini membantu pemain untuk memahami cerita dari keseluruhan game.



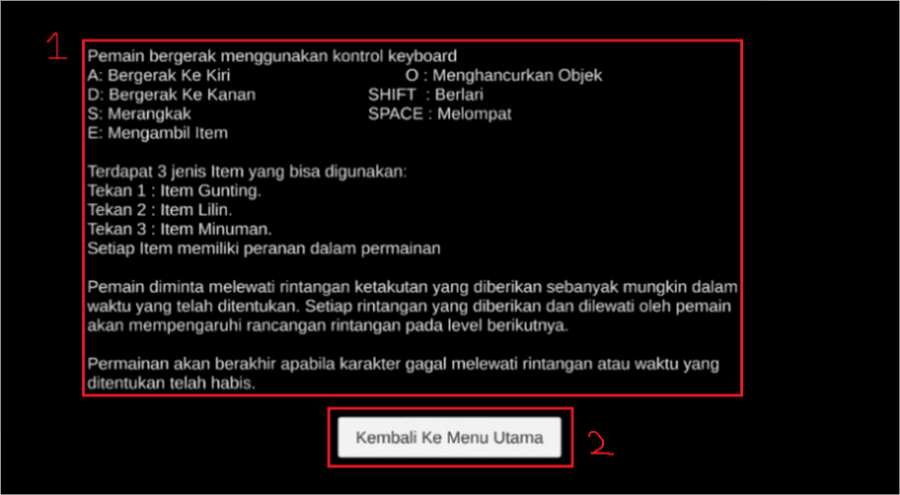
Gambar 4.4

Tampilan Narasi Penutup

Di halaman narasi penutup terdapat sebuah tombol yaitu tombol main menu, dan teks narasi tentang penutup cerita game. Penjelasan lebih lengkapnya diuraikan sebagai berikut :

1. Narasi penutup seputar cerita kondisi karakter setelah permainan dimainkan.
2. Tombol main menu, berfungsi untuk membawa pemain ke halaman utama.
3. Cara Bermain

Halaman cara bermain akan muncul setiap pemain memainkan level. Fungsi interface ini agar pemain dapat mempelajari kontrol game, memahami tujuan game, dan memahami mekanik dasar game yang akan dimainkan.



Gambar 4.5

Tampilan Cara Bermain

Pada gambar terdapat penanda dengan nomor yang akan membantu untuk memberi penjelasan tentang hal-hal yang terdapat pada tampilan cara bermain. Penjelasan lengkapnya diuraikan sebagai berikut :

1. Merupakan penjelasan cara bermain dan pengetahuan singkat tentang game.
2. Tombol untuk kembali menuju menu utama.
3. Interface HUD Karakter

Interface hud player akan muncul setiap pemain memainkan level. Fungsi interface ini agar pemain dapat memonitor status karakter yang dimainkannya. Status yang ditampilkan mencakup darah, waktu, dan benda yang telah diambil.



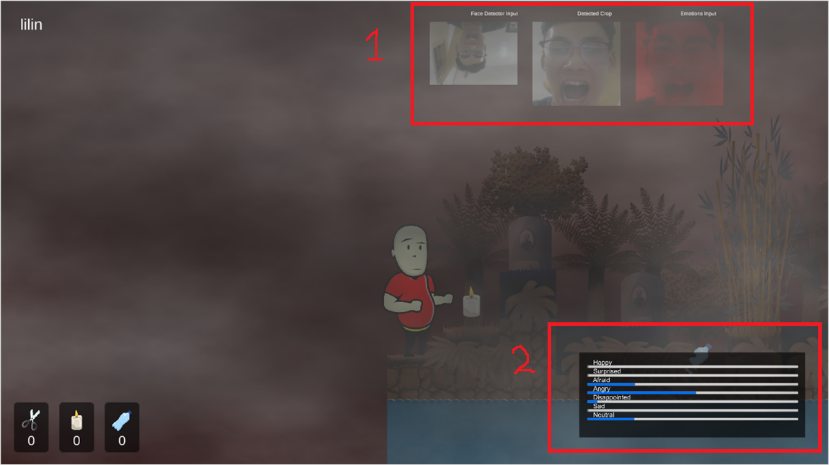
Gambar 4.6

Tampilan HUD Karakter

Pada gambar terdapat penanda dengan nomor yang akan membantu untuk memberi penjelasan tentang hal-hal yang terdapat pada hud karakter. Penjelasan lengkapnya diuraikan sebagai berikut :

1. Merupakan tampilan poin darah karakter yang dimainkan, apabila darah habis maka karakter akan mati.
2. Merupakan tampilan benda yang diambil pemain, jika hendak memakai benda yang diambil maka pemain harus menekan tombol “E” pada keyboard.
3. Merupakan timer sesi permainan, jika habis maka akan dimainkan scene ending yang berbeda-beda berdasarkan dengan nilai performa pemain.
4. Interface Pendeteksian Wajah

Interface pendeteksian wajah akan muncul selama sesi permainan dijalankan. Fungsi interface ini agar membantu penulis memonitor ekspresi pemain selama sesi permainan dijalankan.



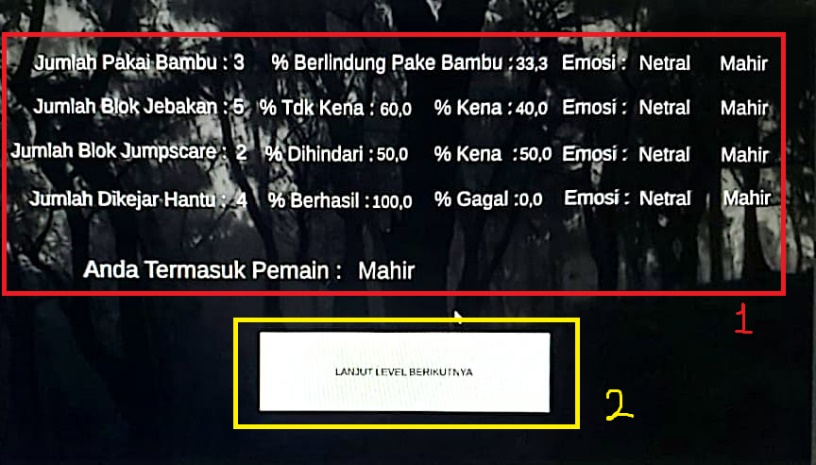
Gambar 4.7

Tampilan Pendeteksi Wajah

Pada gambar terdapat penanda dengan nomor yang akan membantu untuk memberi penjelasan tentang hal-hal yang terdapat pada interface pendeteksian wajah. Penjelasan lengkapnya diuraikan sebagai berikut :

1. Merupakan tampilan penangkapan wajah pemain, ini dapat ditampilkan atau disembunyikan.
2. Merupakan bar emosi yang terdeteksi.
3. Interface Report Performa Pemain

Interface report akan muncul setiap pemain telah berhasil melewati level. Fungsi interface ini agar pemain mengetahui skor yang didapatkannya berdasarkan penilaian DDA.



Gambar 4.8

Tampilan Report Performa Pemain

Pada gambar terdapat penanda dengan nomor yang akan membantu untuk memberi penjelasan tentang hal-hal yang terdapat pada interface report performa. Penjelasan lengkapnya diuraikan sebagai berikut :

1. Merupakan tampilan poin hasil penilaian pemain yang diberikan oleh DDA.
2. Merupakan tombol untuk memulai level berikutnya.
3. **Desain Procedural**

Sub bab ini menjelaskan tentang algoritma-algoritma modul inti yang digunakan didalam game. Penjelasan berupa algoritma bagaimana modul harus berjalan sesuai dengan tujuan yang ditentukan penulis sebelum dilanjutkan menjadi penulisan kode. Algoritma yang akan dibahas yaitu algoritma karakter, algoritma rintangan, dan algoritma DDA.

Algoritma karakter akan membahas algoritma dan segmen program dari kontrol karakter, physics karakter, dan animasi karakter. Algoritma rintangan akan membahas algoritma rintangan dan segmen program yang digunakan untuk mengatur segala jenis yang berkaitan dengan rintangan yang berupa spawn blok hantu, duri, jumpscare penampakan, tempat bersembunyi, item dan kabut. Algoritma DDA akan membahas algoritma dan segmen program dari sistem pencatatan log, penyesuaian rintangan, dan pembentukan rintangan level.

1. **Algoritma Kontrol Karakter**

Pada bagian ini akan dijelaskan langkah-langkah pemain menggerakan

karakter ke kanan atau ke kiri, berjalan, berlari. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.1.

Algoritma 4.1 Kontrol Karakter

1. Cek apakah pemain menekan tombol gerak kekiri (a) atau tombol gerak kekanan (d).
2. Cek apakah pemain tidak kena serang, tidak melompat, dan tidak merangkak.
3. Jika tombol gerak ke kiri ditekan maka flip badan karakter ke kiri, kemudian kurangi koordinat x selama tombol ditekan sesuai dengan speed yang telah ditetapkan.
4. Jika tombol gerak ke kanan ditekan maka flip badan karakter ke kanan, kemudian tambah koordinat x selama tombol ditekan sesuai dengan speed yang telah ditetapkan.
5. Cek apakah tombol sprint/lari (shift) ditekan, jika iya ubah speed karakter.
6. Ketika move true dan sprint true, munculkan animasi berlari, jika move true dan sprint false, munculkan animasi jalan.
7. Cek koordinat x dari karakter setiap karakter bergerak, jika mengalami collision dengan objek lain maka ubah speed karakter jadi 0.

Ketika pemain menekan tombol "a" atau "d", karakter akan bergerak ke kiri atau ke kanan sesuai dengan tombol yang ditekan. Jika pemain menekan tombol untuk melompat, karakter akan melompat ke atas. Jika pemain menggunakan tombol sprint, kecepatan karakter akan meningkat. Animasi berlari akan muncul jika karakter bergerak dan menggunakan sprint, sementara animasi berjalan akan muncul jika karakter bergerak tanpa menggunakan sprint. Setiap kali karakter bergerak, koordinatnya akan diperiksa untuk tabrakan dengan objek lain di lingkungan game. Jika terjadi tabrakan dengan objek, kecepatan karakter akan diubah menjadi 0, menghentikan pergerakannya.

1. **Algoritma Kontrol Merangkak**

Pada bagian ini akan dijelaskan langkah-langkah pemain dalam mode merangkak. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.2.

Algoritma 4.2 Kontrol Merangkak

1. Cek apakah pemain menekan tombol (s).
2. Cek apakah karakter sedang berada ditanah atau tidak.
3. Jika tombol s ditekan maka set mode menjadi merangkak.
4. Kemudian jalankan animasi merangkak.
5. Ubah collider karakter menjadi collider karakter mode merangkak.

Ketika pemain menekan tombol "s", sistem akan memeriksa apakah karakter berada di tanah. Jika iya, karakter akan beralih ke mode merangkak, dan animasi merangkak akan dijalankan. Selain itu, collider karakter akan diubah sesuai dengan mode merangkak untuk interaksi dengan objek di sekitarnya.

1. **Algoritma Bangkit Dari Mode Merangkak**

Pada bagian ini akan dijelaskan langkah-langkah karakter dalam kembali berdiri dari posisi merangkak. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.3.

Algoritma 4.3 Kontrol Bangkit Dari Mode Merangkak

1. Cek apakah pemain menekan tombol (s).
2. Cek apakah karakter sedang dalam mode merangkak.
3. Cek apakah karakter sedang menginjak tanah.
4. Cek apakah diatas karakter ada collider lain.
5. Jalankan animasi karakter bangkit dari merangkak.
6. Ubah collider karakter menjadi collider karakter mode berdiri.

Ketika pemain menekan tombol "s", sistem akan memeriksa apakah karakter sedang dalam mode merangkak. Jika iya, sistem akan memeriksa apakah karakter sedang menginjak tanah. Setelah itu, sistem akan memeriksa apakah di atas karakter ada collider lain. Jika semua kondisi terpenuhi, sistem akan menjalankan animasi karakter bangkit dari posisi merangkak dan mengubah collider karakter menjadi collider karakter dalam mode berdiri.

1. **Algoritma Kontrol Melompat**

Pada bagian ini akan dijelaskan langkah-langkah karakter dalam melakukan tindakan melompat. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.4.

Algoritma 4.4 Kontrol Melompat

1. Cek apakah pemain menekan tombol (space).
2. Cek apakah karakter sedang berada ditanah atau tidak.
3. Jalankan animasi melompat
4. Kurangi koordinat Y karakter berdasarkan yang angka yang sudah diset.

Ketika pemain menekan tombol "space", sistem akan memeriksa apakah karakter berada di tanah. Jika iya, sistem akan menjalankan animasi melompat. Setelah itu, sistem akan mengurangi koordinat Y karakter berdasarkan angka yang telah diset, sehingga karakter tampak melompat ke atas.

1. **Algoritma Karakter Mengambil Item**

Pada bagian ini akan dijelaskan langkah-langkah karakter dalam mengambil item. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.5.

Algoritma 4.5 Karakter Mengambil Item

1. Cek apakah karakter bertabrakan dengan trigger dari item.
2. Cek apakah pemain menekan tombol (e).
3. Cek jenis item yang diambil.
4. Tambah jumlah item yang dibawa karakter.
5. Hilangkan item berserta triggernya.

Ketika karakter bertabrakan dengan trigger dari suatu item, sistem akan memeriksa apakah pemain menekan tombol "e". Jika iya, sistem akan memeriksa jenis item yang diambil. Selanjutnya, sistem akan menambah jumlah item yang dibawa karakter sesuai dengan jenis item yang diambil, dan menghilangkan item beserta triggernya dari level permainan.

1. **Algoritma Karakter Bersembunyi**

Pada bagian ini akan dijelaskan mengenai karakter dalam bersembunyi. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.6.

Algoritma 4.6 Karakter Bersembunyi

1. Cek apakah karakter bertabrakan dengan trigger dari item.
2. Cek apakah karakter didalam area collider tempat sembunyi.
3. Cek apakah karakter dikejar hantu.
4. Status karakter diubah menjadi bersembunyi.

Ketika karakter bertabrakan dengan trigger dari suatu item, sistem akan memeriksa apakah karakter berada di dalam area collider tempat sembunyi. Jika iya, sistem akan memeriksa apakah karakter sedang dikejar oleh hantu. Jika kondisi tersebut terpenuhi, status karakter akan diubah menjadi "bersembunyi".

1. **Algoritma Karakter Dikejar Hantu**

Pada bagian ini akan dijelaskan mengenai karakter dikejar oleh hantu. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.7.

Algoritma 4.7 Karakter Dikejar Hantu

1. Cek apakah karakter menginjak trigger spawn hantu.
2. Random mode hantu yang mengejar karakter.
3. Mainkan suara efek horror
4. Mainkan efek suara dari jenis hantu yang mengejar.
5. Status karakter diubah menjadi dikejar hantu.
6. Cek mode hantu yang mengejar.
7. Munculkan animasi hantu yang mengejar karakter.

Ketika karakter menginjak trigger spawn hantu, sistem akan memeriksa dan memilih secara acak mode hantu yang akan mengejar karakter. Setelah itu, sistem akan memainkan suara efek horror dan efek suara yang sesuai dengan jenis hantu yang mengejar. Selain itu, status karakter akan diubah menjadi "dikejar hantu", dan sistem akan memeriksa mode hantu yang mengejar untuk menampilkan animasi hantu yang sesuai.

1. **Algoritma Karakter Terkena Serangan Hantu**

Pada bagian ini akan dijelaskan mengenai karakter terkena serang oleh hantu. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.8.

Algoritma 4.8 Karakter Terkena Serangan Hantu

1. Cek apakah karakter statusnya dikejar hantu.
2. Cek apakah pemain tidak menggunakan item dalam 3 detik.
3. Cek apakah karakter statusnya berubah menjadi bersembunyi dalam waktu 3 detik.
4. Cek jumlah nyawa karakter.
5. Mainkan suara hantu menangis.
6. Mainkan animasi hantu menyerang karakter.
7. Kurangi nyawa karakter.
8. Cek apakah karakter mati
9. Mainkan animasi karakter terserang.
10. Mainkan animasi karakter mati.
11. Status karakter diubah kembali menjadi tidak dikejar hantu.
12. Hilangkan animasi hantu mengejar.

Saat karakter sedang dikejar hantu, sistem akan memeriksa status karakter untuk mengetahui apakah pemain telah menggunakan item dalam 3 detik. Jika tidak, sistem akan memeriksa apakah status karakter telah berubah menjadi "bersembunyi" dalam waktu 3 detik. Selanjutnya, sistem akan memeriksa jumlah nyawa karakter dan memainkan suara hantu menangis serta animasi hantu menyerang karakter. Setelah itu, sistem akan mengurangi nyawa karakter dan memeriksa apakah karakter telah mati. Jika iya, sistem akan memainkan animasi karakter terserang dan animasi karakter mati, serta mengubah status karakter kembali menjadi tidak dikejar hantu. Terakhir, sistem akan menghilangkan animasi hantu mengejar.

1. **Algoritma Rintangan Hantu**

Pada bagian ini akan dijelaskan mengenai mekanisme rintangan hantu yang digenerate oleh content generator. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.9.

Algoritma 4.9 Rintangan Hantu

1. Cek apakah gameobject bertag “Player” menginjak collider blok jalan hantu
2. Cek apakah status blok sudah dipijak.
3. Cek apakah pemain tidak menggunakan item gunting.
4. Munculkan hantu.
5. Ubah status dipijak menjadi true.

Saat gameobject dengan tag "Player" menginjak collider blok jalan hantu, sistem akan memeriksa apakah status blok tersebut sudah dipijak. Jika blok belum dipijak dan pemain tidak menggunakan item gunting, maka sistem akan mulai memunculkan hantu. Setelah itu, sistem akan mengubah status dipijak menjadi true untuk blok tersebut.

1. **Algoritma Jumpscare Penampakan**

Pada bagian ini akan dijelaskan mengenai mekanisme rintangan jumpscare penampakan yang digenerate oleh content generator. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.10.

Algoritma 4.10 Rintangan Jumpscare Penampakan

1. Cek apakah gameobject bertag “Player” menginjak collider blok jumpscare.
2. Cek apakah status blok sudah dipijak.
3. Memanggil canvas jumpscare.
4. Memainkan suara jumpscare.
5. Mematikan collider blok.
6. Menghitung mundur canvas hingga hilang.
7. Menutup canvas jumpscare.

Ketika gameobject dengan tag "Player" menginjak collider blok jumpscare, sistem akan memeriksa apakah status blok tersebut sudah dipijak. Jika blok tersebut belum dipijak, maka sistem akan melakukan serangkaian tindakan. Pertama-tama, sistem akan memanggil canvas jumpscare untuk menampilkan efek jumpscare kepada pemain. Selanjutnya, sistem akan memainkan suara jumpscare untuk meningkatkan efek ketegangan. Setelah itu, sistem akan mematikan collider blok agar tidak ada lagi interaksi dengan pemain. Kemudian, sistem akan menghitung mundur pada canvas jumpscare hingga efeknya hilang. Setelah efek jumpscare hilang, sistem akan menutup canvas jumpscare untuk melanjutkan permainan.

1. **Algoritma Item**

Pada bagian ini akan dijelaskan mengenai mekanisme item yang digenerate oleh content generator. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.11.

Algoritma 4.11 Algoritma Item

1. Mengecek pemain apakah memiliki item yang disimpan dalam tas.
2. Mengecek pemain menekan tombol item apa.
3. Mengurangi jumlah item yang dipakai.
4. Memunculkan efek sprite item telah terpakai.
5. Melakukan hitung mundur sebagai delay penggunaan item.

Sistem akan memeriksa apakah pemain memiliki item yang disimpan dalam tas. Jika iya, sistem akan memeriksa tombol item apa yang ditekan oleh pemain. Setelah itu, sistem akan mengurangi jumlah item yang dipakai dan memunculkan efek sprite untuk menunjukkan bahwa item tersebut telah digunakan. Selain itu, sistem akan memulai hitung mundur sebagai delay sebelum pemain dapat menggunakan item kembali.

1. **Algoritma Tempat Bersembunyi**

Pada bagian ini akan dijelaskan mengenai mengenai mekanisme tempat bersembunyi yang digenerate oleh content generator. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.12.

Algoritma 4.12 Algoritma Tempat Bersembunyi

1. Cek apakah gameobject bertag “Player” menginjak collider blok bambu.
2. Memberi status masuk = true.
3. Cek apakah status masuk = true dan pemain tetap berada didalam.
4. Cek tipe bambu yang digunakan untuk bersembunyi.
5. Jika bambu pendek dan status karakter pemain jongkok = true
6. Memberi status bersembunyi = true.
7. Cek apakah karakter keluar dari trigger bambu.
8. Memberi status masuk = false.
9. Memberi status bersembunyi = false.

Saat gameobject dengan tag "Player" menginjak collider blok bambu, sistem akan memberi status masuk menjadi true. Selanjutnya, sistem akan memeriksa apakah status masuk sudah bernilai true dan apakah pemain masih berada di dalam collider bambu. Setelah itu, sistem akan memeriksa tipe bambu yang digunakan untuk bersembunyi. Jika bambu tersebut adalah tipe pendek dan status karakter pemain adalah jongkok (jongkok = true), maka sistem akan memberi status bersembunyi menjadi true. Kemudian, sistem akan memeriksa apakah karakter keluar dari trigger bambu. Jika karakter keluar dari trigger, sistem akan mengubah status masuk menjadi false dan status bersembunyi menjadi false.

1. **Algoritma Rintangan Duri**

Pada bagian ini akan dijelaskan mengenai mekanisme rintangan duri yang digenerate oleh content generator. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.13.

Algoritma 4.13 Rintangan Duri

1. Cek apakah gameobject bertag “Area Pukul” menginjak collider rintangan duri.
2. Mainkan suara menghancurkan objek.
3. Hapus objek rintangan duri.
4. Cek apakah gameobject bertag “Player” menginjak collider rintangan duri.
5. Memberi status karakter kena serangan.
6. Memberi status game object active = false.

Saat gameobject dengan tag "Area Pukul" menginjak collider rintangan duri, sistem akan memainkan suara menghancurkan objek dan menghapus objek rintangan duri tersebut. Selanjutnya, sistem akan memeriksa apakah gameobject dengan tag "Player" menginjak collider rintangan duri. Jika iya, sistem akan memberi status karakter kena serangan dan menonaktifkan game object dengan status active = false.

1. **Algoritma Pencatatan Log DDA**

Pada bagian ini akan dijelaskan mengenai langkah-langkah pencatatan log. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.14.

Algoritma 4.14 Pencatatan Log DDA

1. Cek apakah karakter menabrak trigger rintangan.
2. Cek jenis rintangan yang ditabrak collider karakter.
3. Cek emosi yang terdeteksi saat menginjak blok rintangan.
4. Cek status rintangan yang ditabrak.
5. Mencatat dalam array data yang telah ditangkap.
6. Cek jika level telah selesai dan checkpoint diaktifkan.
7. Melakukan generate json berdasarkan data yang ditampung dalam array.
8. Menyimpan log pada direktori yang dituju.

Saat karakter menabrak trigger rintangan, sistem akan memeriksa jenis rintangan yang ditabrak oleh collider karakter. Setelah itu, sistem akan memeriksa emosi yang terdeteksi saat karakter menginjak blok rintangan dan status rintangan yang ditabrak. Kemudian, sistem akan mencatat informasi tersebut dalam sebuah array data yang telah ditangkap. Selanjutnya, sistem akan memeriksa apakah level telah selesai dan checkpoint diaktifkan. Jika level telah selesai dan checkpoint diaktifkan, sistem akan melakukan generate JSON berdasarkan data yang telah ditampung dalam array. Terakhir, sistem akan menyimpan log pada direktori yang dituju.

1. **Algoritma Penyesuaian Rintangan DDA**

Pada bagian ini akan dijelaskan langkah-langkah rintangan disesuaikan oleh DDA. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.15.

Algoritma 4.15 Penyesuaian Rintangan DDA

1. Membuka file Log.
2. Cek berapa banyak blok rintangan.
3. Cek jenis blok rintangan yang dicatat.
4. Cek emosi yang terdeteksi saat menginjak blok rintangan.
5. Cek status karakter setelah melewati rintangan.
6. Cek berapa kali pemain berhasil melewati rintangan.
7. Cek berapa kali pemain gagal melewati rintangan.
8. Cek berapa banyak blok rintangan yang tidak diinjak.

Algoritma 4.15 (Lanjutan)

1. Memberi skor performa pemain.
2. Memberi skor emosi pemain.
3. Membandingkan dengan data *weight clipping* DDA yang dibuat.
4. Mengubah banyak blok rintangan pada level berikutnya.
5. Membersihkan file log DDA ketika level berikutnya telah dimuat.

Saat membuka file Log, sistem akan melakukan serangkaian pemeriksaan terkait data yang tercatat dalam log tersebut. Sistem akan memeriksa jumlah blok rintangan yang tercatat dalam file Log dan jenis blok rintangan yang dicatat. Saat karakter menginjak blok rintangan, sistem akan memeriksa emosi yang terdeteksi pada saat itu dan status karakter setelah melewati rintangan. Sistem akan mencatat berapa kali pemain berhasil dan gagal melewati rintangan, serta berapa banyak blok rintangan yang tidak diinjak. Berdasarkan data yang tercatat, sistem akan memberikan skor performa dan skor emosi pemain. Selanjutnya, sistem akan membandingkan data dengan pola atau aturan yang telah ditetapkan, seperti weight clipping Dynamic Difficulty Adjustment (DDA), dan mengubah banyaknya blok rintangan pada level berikutnya. Ketika level berikutnya dimuat, sistem akan membersihkan file log DDA yang terkait.

1. **Algoritma Pembentukan Rintangan Level**

Pada bagian ini akan dijelaskan langkah-langkah rintangan pada level berikutnya dibentuk dengan data dari log DDA. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.16.

**Algoritma 4.16 Pembentukan Rintangan Level**

1. Membaca hasil pengolahan weight ceiling DDA.
2. Membuat checkpoint pada awal level
3. Membuat gameobject air dengan panjang dan tinggi yang ditentukan.
4. Menghitung persentase blok rintangan muncul terhadap panjang platform.
5. Membuat gameobject blok rintangan dengan posisi random sebanyak jumlah yang ditentukan DDA.
6. Membuat jalan biasa dengan posisi random sebanyak jumlah sisa dari perhitungan persentase.
7. Membuat gameobject blok naik-turun pada ujung kiri setiap jalan yang tidak memiliki jalan biasa.
8. Membuat checkpoint pada akhir level.

Sistem akan membaca hasil pengolahan weight ceiling DDA untuk menyesuaikan tingkat kesulitan permainan. Kemudian, sistem akan membuat checkpoint pada awal level untuk memudahkan pemain dalam melanjutkan permainan. Selanjutnya, sistem akan membuat gameobject air dengan panjang dan tinggi yang telah ditentukan. Selanjutnya, sistem akan menghitung persentase blok rintangan yang muncul terhadap panjang platform. Berdasarkan perhitungan tersebut, sistem akan membuat gameobject blok rintangan dengan posisi acak sesuai dengan jumlah yang ditentukan oleh DDA. Selain itu, sistem akan membuat jalan biasa dengan posisi acak sesuai dengan jumlah sisa dari perhitungan persentase. Jika sebuah jalan tidak memiliki jalan biasa, sistem akan membuat gameobject blok naik-turun pada ujung kiri setiap jalan tersebut. Terakhir, sistem akan membuat checkpoint pada akhir level untuk menandai pencapaian pemain.

1. Hunicke, R. *The case for dynamic difficulty adjustment in games*. In Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology (pp. 429-433), 2005, hlm 14. [↑](#footnote-ref-1)
2. Missura, O. *Dynamic difficulty adjustment* (Doctoral dissertation, Universitäts-und Landesbibliothek Bonn), 2015, hlm 30-38. [↑](#footnote-ref-2)