**BAB V**

**IMPLEMENTASI PROGRAM**

Bab ini akan menjelaskan mengenai proses implementasi program hingga menjadi satu kesatuan program berbentuk game yang dapat dimainkan. Implementasi dilakukan berdasarkan hasil analisa dan desain sistem dengan menggunakan bahasa pemprograman. Hal yang akan dibahas melingkupi sistem pencatat dan pembaca log, penyesuai tingkat kesulitan rintangan, karakter, dan rintangan.

1. **Sistem Pencatat dan Pembaca Log**

Pada subbab 5.1 ini akan menjelaskan segmen program yang berkaitan dengan sistem pencatatan dan pembacaan log. Segmen program yang dibahas yaitu sistem pencatatan log, dan sistem pembacaan log.

1. **Sistem Pencatatan Log**

Bagian ini menjelaskan sistem pencatatan log berdasarkan variabel yang ditampung selama melewati satu level dari berbagai jenis rintangan yang dilewati. Terdapat 2 segmen program pada bagian sistem pencatatan log yaitu segmen 5.1 Pencatatan Emosi, dan segmen 5.2 Pencatatan Log. Penjelasan segmen program akan dijabarkan setelah segmen program.

Segmen Program 5.1 Pencatatan Emosi

1. public string deteksi\_emosi()
2. {
3. string emotion = "";
4. value\_marah = EmotionsManager.Emotions.angry;
5. value\_netral = EmotionsManager.Emotions.neutral;
6. value\_takut = EmotionsManager.Emotions.scared;
7. if (value\_netral-value\_takut > value\_takut && value\_netral > value\_marah){
8. emotion = "Netral";
9. }
10. else if (value\_takut > value\_netral - value\_takut && value\_takut > value\_marah){

**Segmen Program 5.1 (Lanjutan)**

1. emotion = "Takut";
2. }
3. else if (value\_marah > value\_netral && value\_takut < value\_marah)
4. {
5. emotion = "Marah";
6. }
7. return emotion;
8. }

Segmen Program 5.1 merupakan baris program yang digunakan untuk mencatat emosi yang dideteksi. Segmen program ini akan dipanggil setiap kali collider karakter pemain bertabrakan dengan collider rintangan dan ketika terkena serang hantu. Pada baris 1 merupakan nama function yang dipanggil oleh script collider rintangan dan berisi perintah hingga baris ke 17. Baris 3 merupakan deklarasi awal variable emosi. Baris 4 hingga 6 berisi deklarasi variable yang diisi oleh value deteksi emosi library MoodMe. Kemudian pada baris ke 7 hingga baris ke 9 merupakan kondisi pengecekan untuk menentukan emosi yang dideteksi yaitu “Netral”. Baris ke 10 hingga baris ke 12 merupakan fungsi yang serupa yaitu untuk mendeteksi emosi “Takut”. Baris ke 13 hingga baris ke 16 merupakan fungsi yang serupa yaitu untuk mendeteksi emosi “Marah”. Kemudian baris ke 17 berfungsi untuk mengembalikan value emotion untuk dibaca oleh skrip pencatat log.

Segmen Program 5.2 Pencatatan Log

1. void AddData(string emotion)
2. {
3. if (hantu)
4. {
5. obstacle = "hantu";
6. status = hantu.status\_karakter();
7. }
8. else if (jebakan)
9. {
10. obstacle = "duri";
11. status = jebakan.status\_karakter();
12. }
13. else if (jumpscares)
14. {
15. obstacle = "jumpscare\_penampakan";
16. status = jumpscares.status\_karakter();
17. }
18. else if (bambus)
19. {

**Segmen Program 5.2 (Lanjutan)**

1. obstacle = "bambu";
2. status = bambus.status\_karakter();
3. }
4. Data data = new Data();
5. data.emotion = emotion;
6. data.obstacle = obstacle;
7. data.status = status;
8. dataList.Add(data);
9. }
10. }
11. string ConvertListToJson(List<Data> list)
12. {
13. string json = JsonConvert.SerializeObject(new { logEntries = list }, Formatting.Indented);
14. return json;
15. }
16. void SaveJsonToFile(string json, string filename)
17. {
18. string path = Path.Combine(Application.persistentDataPath, filename);
19. File.WriteAllText(path, json);
20. }

Segmen Program 5.2 merupakan baris program yang digunakan untuk mencatat jenis rintangan, status, dan emosi rintangan yang dicatat dalam bentuk json. Baris 1 hingga baris 30 merupakan function yang mengolah bentuk data pencatatan. Baris ke 3,8,13, dan 18 merupakan pengecekan jenis rintangan. Baris ke 5 dan ke 6 akan mengisi jenis rintangan dan status dari jenis rintangan hantu. Baris ke 10 dan ke 11 akan mengisi jenis rintangan dan status dari jenis rintangan duri. Baris ke 15 dan ke 16 akan mengisi jenis rintangan dan status dari jenis rintangan jumpscare. Baris ke 20 dan ke 21 akan mengisi jenis rintangan dan status dari jenis rintangan bambu. Baris ke 23 hingga baris 28 berfungsi untuk menampung data menjadi bentuk array yang selanjutnya akan dikonversi menjadi json dengan function baris ke 31. Baris ke 31 merupakan function yang dipanggil untuk melakukan convert json, cara kerjanya pada baris ke 33 memanggil library JsonConvert dan melakukan serialize object yang berisi data array dengan nama list yang digunakan untuk menampung data jenis rintangan, status dan emosi. Kemudian baris ke 34 melakukan return hasil json. Baris ke 36 merupakan function untuk mengenerate file json dengan entri data berdasarkan isian json pada baris ke 34. Baris ke 38 berisi path yang dipilih dimana pada lokasi tersebut file akan diexport, kemudian baris ke 39 melakukan write file dengan data dari json ke direktori yang ditentukan oleh path pada baris ke 38.

1. **Sistem Pembacaan Log**

Bagian ini menjelaskan sistem pembacaan log json yang telah digenerate. Terdapat 3 segmen program pada bagian sistem pembacaan log yaitu segmen 5.3 Pengecekan File Log, Segmen 5.4 Sistem Pembaca Log dan segmen 5.5 Pembacaan Log. Penjelasan segmen program akan dijabarkan setelah segmen program.

Segmen Program 5.3 Pengecekan File Log

1. public string LogFilePath
2. {
3. get
4. {
5. if (string.IsNullOrEmpty(logFilePath))
6. {
7. logFilePath = Path.Combine(Application.persistentDataPath, "output.json");
8. }
9. return logFilePath;
10. }
11. }

Segmen Program 5.3 merupakan baris program yang digunakan untuk melakukan pengecekan apakah file log terdeteksi ada pada sistem. Baris ke 1 merupakan nama function yang dipanggil ketika log akan dibaca. Baris ke 5 melakukan pengecekan apakah filepath telah diset, jika belum maka file path pada baris ke 7 diisi oleh path default appData windows dan mencari file dengan nama output.json. Kemudian path direturn pada baris ke 9.

Segmen Program 5.4 Sistem Pembaca Log

1. public void ReadJsonLog()
2. {
3. try
4. {
5. string jsonContent = System.IO.File.ReadAllText(LogFilePath);
6. LogWrapper logWrapper = JsonUtility.FromJson<LogWrapper>(jsonContent);
7. if (logWrapper != null && logWrapper.logEntries != null)
8. {

**Segmen Program 5.4 (Lanjutan)**

1. logEntries.AddRange(logWrapper.logEntries);
2. }
3. }
4. catch (System.Exception e)
5. {
6. Debug.LogError("Error reading JSON log: " + e.Message);
7. }
8. }

Segmen Program 5.4 merupakan baris sistem yang digunakan untuk melakukan pembacaan file log json. Baris ke 3 melakukan try-catch pada file json, akan masuk kondisi catch jika tidak bisa membaca file log pada baris ke 13 dengan memunculkan debug error pada baris ke 14. Jika log dapat dibaca maka baris ke 5 akan melakukan perintah membaca semua teks dengan memanfaatkan library system.io dengan path yang telah dicari pada segmen program 5.3. Kemudian baris ke 6 memanggil library JsonUtility yang membaca konten hasil dari baris ke 5. Baris ke 9 melakukan penambahan data ke logEntries berdasarkan data dari baris ke 5 dan ke 6.

Segmen Program 5.5 Pembacaan Log

1. public void BacaLog()
2. {
3. pembacaLogs = GetComponent<PembacaLog>();
4. List<PembacaLog.LogEntry> logEntries = pembacaLogs.LogEntries;
5. foreach (PembacaLog.LogEntry entry in logEntries)
6. {
7. if (entry.obstacle == "hantu")
8. {
9. if (entry.status == "Lolos")
10. {
11. ctr\_hantu\_sukses++;
12. if (entry.emotion == "Netral")
13. {
14. skor\_emosi\_hantu += 2;
15. }
16. else if (entry.emotion == "Marah")
17. {
18. skor\_emosi\_hantu++;
19. }
20. }
21. else if (entry.status == "Gagal")
22. {
23. ctr\_hantu\_gagal++;
24. }
25. hantu\_tidak\_injak--;

**Segmen Program 5.5 (Lanjutan)**

1. }
2. else if (entry.obstacle == "jumpscare\_penampakan")
3. {
4. if (entry.status == "Aktif")
5. {
6. ctr\_penampakan\_muncul++;
7. if (entry.emotion == "Netral")
8. {
9. skor\_emosi\_penampakan += 2;
10. }
11. else if (entry.emotion == "Marah")
12. {
13. skor\_emosi\_penampakan += 1;
14. }
15. }
16. penampakan\_tidak\_injak--;
17. }
18. else if (entry.obstacle == "duri")
19. {
20. if (entry.status == "Gagal")
21. {
22. ctr\_duri\_gagal++;
23. if (entry.emotion == "Marah")
24. {
25. skor\_emosi\_duri += 1;
26. }
27. }
28. else if (entry.status == "Sukses")
29. {
30. ctr\_duri\_berhasil++;
31. skor\_emosi\_duri += 2;
32. }
33. duri\_tidak\_injak--;
34. }
35. else if (entry.obstacle == "bambu")
36. {
37. ctr\_masuk\_bambu++;
38. if (entry.emotion == "Marah")
39. {
40. skor\_emosi\_bambu += 1;
41. }
42. else if (entry.emotion == "Normal")
43. {
44. skor\_emosi\_bambu += 2;
45. }
46. bambu\_tidak\_dimasuki--;
47. }
48. }
49. HitungSkorDDARintangan(sukses, gagal, emosi, noninjak);
50. }

Segmen Program 5.5 merupakan baris program yang digunakan untuk memetakan menjadi kategori rintangan berdasarkan file json yang dibaca. Baris ke 3 melakukan get component apakah PembacaLog terbaca. Baris ke 5 melakukan looping untuk membaca entri data pada logEntries yang dideklarasi pada baris ke 4. Baris ke 7, 27, 43, dan ke 60 berfungsi untuk memetakkan jenis rintangan. Baris ke 9, 21, 29, 45, dan 53 melakukan pengecekan log berisi status. Jika memenuhi syarat maka counter akan diubah seperti pada baris ke 11, 23, 25, 31, 41, 47, 55, 58, 62, dan 71. Jika gagal maka baris ke 21 akan berjalan dan counter hantu gagal akan ditambahkan pada baris ke 23. Kemudian baris ke 12, 16, 32, 36, 48, 63, dan 67 melakukan pengecekan emosi yang terdeteksi pada masing-masing kategori. Jika emosi netral maka skor\_emosi\_kategori akan ditambah 2, jika emosi marah maka skor\_emosi\_kategori akan ditambah 1. Kemudian jika baris log sudah habis terbaca, berikutnya dilakukan pemanggilan function DDA rintangan dengan parameter yang dibutuhkan pada baris ke 74.

1. **Sistem Perubah Tingkat Kesulitan Rintangan**

Pada subbab 5.2 ini akan menjelaskan segmen program yang berkaitan dengan sistem perubah tingkat kesulitan rintangan. Segmen program yang dibahas yaitu sistem penentuan kategori pemain, perhitungan skor emosi pemain, dan sistem penyesuai tingkat kesulitan rintangan.

1. **Sistem Penentuan Kategori Pemain**

Bagian ini menjelaskan sistem penentuan kategori pemain berdasarkan performa pemain selama melewati satu level dengan berbagai jenis rintangan yang diolah DDA. Terdapat 1 segmen program pada bagian sistem penentuan kategori pemain yaitu segmen 5.6 Penentuan Kategori Pemain. Penjelasan segmen akan dijabarkan setelah segmen program.

Segmen Program 5.6 Penentuan Kategori Pemain

1. jumlahJalanHantu = GlobalStatic.ctr\_pjhantu;
2. float htg\_jalanhantu = jumlahJalanHantu\*100;
3. float htg\_hantusukses = (ctr\_hantu\_sukses+hantu\_tidak\_injak)\*100;
4. float htg\_hantugagal = ctr\_hantu\_gagal\*100;
5. string kategori\_pemain = "";
6. if (jumlahJalanHantu == 0)

**Segmen Program 5.6 (Lanjutan)**

1. {
2. kategori\_pemain = "Normal";
3. }
4. else if(hantu\_tidak\_injak==jumlahJalanHantu)
5. {
6. kategori\_pemain = "Mahir";
7. }
8. else
9. {
10. float persentase\_sukses = (htg\_hantusukses / htg\_jalanhantu)\*100;
11. float persentase\_gagal = (htg\_hantugagal / htg\_jalanhantu)\*100;
12. if (persentase\_sukses<39)
13. {
14. kategori\_pemain = "Pemula";
15. }
16. else if (persentase\_sukses == persentase\_gagal)
17. {
18. kategori\_pemain = "Normal";
19. }
20. else if (persentase\_sukses > persentase\_gagal && persentase\_sukses > 50)
21. {
22. kategori\_pemain = "Normal";
23. }
24. else if (persentase\_sukses > persentase\_gagal && persentase\_sukses > 75)
25. {
26. kategori\_pemain = "Mahir";
27. }
28. else
29. {
30. kategori\_pemain = "Mahir";
31. }
32. }

Segmen Program 5.6 merupakan baris program untuk menentukan kategori pemain berdasarkan kategori yang telah dipetakkan. Segmen Program 5.6 ini mengambil contoh kategori hantu. Baris ke 1 hingga baris ke 5 merupakan variable deklarasi awal sebelum perhitungan. Baris ke 6 melakukan pengecekan apakah pada level tidak tedapat jalan hantu, jika iya maka kategori pemain akan dibuat normal yang berfungsi untuk merandom jalan dengan weight ceiling normal sehingga jalan tidak menjadi 0 lagi. Baris ke 10 melakukan pengecekan apakah jalan dilewati semua, jika iya maka performa pemain akan dibuat menjadi mahir. Kemudian baris ke 14 merupakan kondisi normal dan melakukan perhitungan didalamnya. Baris ke 16 dan 17 melakukan perhitungan persentase keberhasilan dan kegagalan pemain dalam melewati rintangan. Kemudian pada baris ke 18, 22, 26, 30 dan 34 melakukan pengecekan performa pemain berdasarkan perbandingan persentase keberhasilan dengan kegagalan pemain. Kemudian kategori pemain akan ditentukan pada baris ke 20, 24, 28, 32, dan 36. Selanjutnya data kategori pemain akan digunakan untuk Segmen Program 5.7 Perhitungan Skor Emosi.

1. **Sistem Perhitungan Skor Emosi**

Bagian ini menjelaskan sistem perhitungan skor emosi pemain oleh DDA. Terdapat 1 segmen program pada bagian perhitungan skor emosi DDA yaitu segmen 5.7 Perhitungan Skor Emosi.

Segmen Program 5.7 Perhitungan Skor Emosi

1. skor\_emosi\_hantu = skor\_emosi\_hantu+(hantu\_tidak\_injak\*2);
2. float persentaseKetakutan = Mathf.Clamp01((float)skor\_emosi\_hantu / jumlahJalanHantu);
3. if(hantu\_tidak\_injak==jumlahJalanHantu)
4. {
5. skor\_emosi\_hantu=2\*jumlahJalanHantu;
6. kategori\_pemain = "Mahir";
7. }
8. else if (kategori\_pemain=="Pemula" && jumlahJalanHantu != 0)
9. {
10. if (persentaseKetakutan < 1f)
11. {
12. kategori\_pemain = "Pemula";
13. }
14. else
15. {
16. kategori\_pemain = "Normal";
17. }
18. }
19. else if (kategori\_pemain=="Normal" && jumlahJalanHantu != 0)
20. {
21. if (persentaseKetakutan > 1.5f)
22. {
23. kategori\_pemain = "Mahir";
24. }
25. else
26. {
27. kategori\_pemain = "Normal";
28. }

**Segmen Program 5.7 (Lanjutan)**

1. }
2. else if (kategori\_pemain == "Mahir" && jumlahJalanHantu != 0)
3. {
4. if (persentaseKetakutan < 1.5f)
5. {
6. kategori\_pemain = "Normal";
7. }
8. else
9. {
10. kategori\_pemain = "Mahir";
11. }
12. }
13. else
14. {
15. kategori\_pemain = "Mahir";
16. }

Segmen Program 5.7 merupakan baris program untuk memvalidasi skor emosi pemain yang terdeteksi. Baris 1 dan 2 merupakan deklarasi awal variable untuk menghitung persentase ketakutan. Baris ke 3 melakukan pengecekan hantu\_tidak\_injak, jika sama dengan jumlahJalanHantu maka skor emosi akan dikalikan 2 sebagai representasi emosi normal atau biasa saja. Kemudian baris ke 6 melakukan penggolongan pemain “Mahir”. Baris ke 8 dijalankan apabila kondisi baris ke 3 tidak terpenuhi dan kategori pemain yang didapat dari Segmen Program 5.6 hasilnya “Pemula”. Kemudian baris ke 10 melakukan pengecekan apakah persentase ketakutan yang diperoleh dibawah 1, jika iya maka pemain dianggap takut dan kategori pemainnya tetap “Pemula”, jika tidak maka baris perintah ke 14 akan berjalan dan men-set kategori pemain menjadi “Normal” pada baris ke 16. Baris ke 19 dijalankan apabila kondisi baris ke 10 tidak terpenuhi dan kategori pemain yang didapat dari Segmen Program 5.6 hasilnya “Normal”. Baris ke 21 melakukan pengecekan apakah persentase ketakutan yang diperoleh diatas 1.5, jika iya maka kategori pemain akan diubah menjadi “Mahir”, jika tidak akan diubah tetap menjadi “Normal”. Kemudian baris ke 30 dijalankan apabila kondisi baris ke 19 tidak terpenuhi dan kategori pemain yang didapat dari Segmen Program 5.6 hasilnya “Mahir”. Pengecekan dilakukan pada baris ke 32 apakah persentase ketakutan pemain dibawah 1.5, jika iya maka kategori pemain akan di-set “Normal” pada baris ke 34. Sedangkan baris ke 38 dijalankan jika tidak lolos pengecekan pada baris ke 32.

1. **Sistem Penyesuai Tingkat Kesulitan Rintangan**

Bagian ini menjelaskan sistem penyesuai tingkat kesulitan rintangan oleh DDA. Terdapat 1 segmen program pada bagian penyesuai kesulitan rintangan yaitu segmen 5.8 Sistem Penyesuai Tingkat Kesulitan.

Segmen Program 5.8 Sistem Penyesuai Tingkat Kesulitan

1. int[] wc\_hantu = new int[] { 2, 4, 5 };
2. if (kategori\_pemain == "Pemula")
3. {
4. GlobalStatic.init\_jumlahJalanHantu = wc\_hantu[0];
5. }
6. else if (kategori\_pemain == "Normal")
7. {
8. GlobalStatic.init\_jumlahJalanHantu = wc\_hantu[1];
9. }
10. else if (kategori\_pemain == "Mahir")
11. {
12. GlobalStatic.init\_jumlahJalanHantu = wc\_hantu[2];
13. }

Segmen Program 5.8 merupakan baris program untuk mengatur weight ceiling dari rintangan kategori hantu. Baris ke 1 merupakan array yang menampung batas maksimum rintangan hantu yang akan spawn. Baris ke 2, 6 dan 10 merupakan baris pengecekan kategori pemain. Baris ke 4, 8, dan 12 memodifikasi nilai inisialisasi jumlah\_JalanHantu pada variabel global yang akan dibaca oleh generator level.

1. **Karakter**

Subbab 5.3 ini akan menjelaskan mengenai segmen program yang berkaitan dengan karakter. Segmen program yang dibahas yaitu karakter bergerak, merangkak, melompat, memukul dan karakter saat terkena serang.

1. Kontrol Gerak Karakter

Bagian ini menjelaskan kontrol pemain untuk menggerakkan karakter ke kiri dan kekanan, berjalan, dan berlari. Karakter dalam game ini digerakkan menggunakan kontrol keyboard. Isi dari kontrol gerak karakter akan dibahas dalam segmen program 5.9.

Segmen Program 5.9 Kontrol Gerak Karakter

1. gerak = Input.GetAxisRaw("Horizontal");
2. if (Input.GetKeyDown(KeyCode.S))
3. {
4. if (jongkok)
5. {
6. modeBerdiri();
7. }
8. else
9. {
10. modeJongkok();
11. }
12. }
13. if (Input.GetKey(KeyCode.LeftShift) && !jongkok)
14. {
15. modeLari();
16. }
17. else if (!Input.GetKey(KeyCode.LeftShift) && !jongkok)
18. {
19. modeJalan();
20. }
21. if (Input.GetButtonDown("Jump") && !jongkok)
22. {
23. modeLompat();
24. }
25. if (Input.GetKey(KeyCode.O) && isAttack == false)
26. {
27. modePukul();
28. }
29. public void modeBerdiri()
30. {
31. if (!isCrawling)
32. {
33. jongkok = false;
34. modeSerang = 0.8f;
35. if (lari)
36. {
37. audioSource.clip = runSound;
38. moveSpeed = dxLari;
39. }
40. }
41. }

Segmen Program 5.9 merupakan baris program yang digunakan untuk menggerakan karakter didalam game. Pada baris 1 dilakukan pengambilan input dari keyboard yang telah diatur dengan menggunakan tombol A untuk bergerak kekiri dan tombol D untuk bergerak kekanan. Baris ke 4 hingga baris ke 11 merupakan pengecekan apakah karakter berada dalam kondisi merangkak. Jika pemain menekan tombol S saat posisi karakter berada dalam kondisi merangkak, maka karakter akan kembali kedalam mode berdiri. Jika pemain menekan tombol S pada saat karakter dalam posisi lain, maka karakter akan masuk kedalam mode merangkak. Baris ke 13 hingga baris ke 20 merupakan pengecekan kontrol karakter berlari. Terdapat pengecekan apakah pemain menekan tombol Shift dan tidak dalam mode merangkak, jika iya maka karakter masuk kedalam mode lari. Jika tidak maka karakter akan berada dalam mode jalan biasa. Baris ke 21 hingga baris ke 24 merupakan baris program untuk melakukan lompat pada karakter. Terdapat pengecekan apakah pemain menekan tombol Space dan tidak dalam mode merangkak, jika iya maka karakter masuk mode lompat. Baris ke 25 hingga baris ke 28 merupakan pengecekan apakah pemain menekan tombol O, jika iya maka karakter akan masuk ke mode pukul.

Kemudian pada baris ke 30 merupakan function pengaturan saat pemain berdiri pemain dapat melakukan aksi lari dan pengaturan serangan menjadi mode memukul. Pada baris ke 32 dilakukan pengecekan apakah karakter saat ini sedang dalam mode menunduk, jika lolos pengecekan maka jongkok difalsekan dan mode serang diatur menjadi mode memukul pada baris ke 35 hingga baris ke 36. Kemudian pada baris ke 36 dilakukan pengecekan apakah karakter saat ini dalam mode lari, jika iya maka mainkan suara lari dan speed karakter diganti menjadi speed lari pada pada baris ke 38 dan baris ke 39.

1. Karakter Merangkak

Bagian ini menjelaskan segmen program saat pemain menekan tombol aksi untuk membuat karakter merangkak. Terdapat 2 segmen program pada ini, segmen 5.10 Mode Merangkak, dan segmen 5.11 Kontrol Merangkak.

Segmen Program 5.10 Mode Merangkak

1. public void modeJongkok()
2. {
3. if (isGrounded)
4. {
5. jongkok = true;
6. modeSerang = 0f;
7. }
8. }

Segmen Program 5.10 merupakan baris program yang digunakan untuk membuat karakter melakukan aksi merangkak. Pada baris 1 merupakan nama function yang dipanggil ketika pemain menekan tombol S dan berisi perintah hingga baris ke 8. Baris 3 hingga baris 5 berisi pengecekan apakah pemain berada diatas tanah, jika iya maka karakter dapat melakukan aksi merangkak, dan mode serangnya dijadikan 0.

Segmen Program 5.11 Kontrol Merangkak

1. if (jongkok)
2. {
3. isCrawling = Physics2D.OverlapCircle(crawlCheck.position, upRadius, whatisTop);
4. }

Segmen Program 5.11 merupakan baris program yang digunakan untuk membuat karakter melakukan aksi merangkak. Baris 1 berisi pengecekan apakah pemain sedang dalam mode merangkak, jika iya maka collider circle pemain diisi oleh crawlcheck, upradius dan whatistop saat itu. What is top merupakan variable yang diisi oleh gameobject yang ditentukan oleh pemain, agar bisa menseleksi layer platform diatas pemain jika ada. Up radius merupakan variable yang mengatur seberapa setinggi collider karakter saat mode merangkak.

1. Karakter Melompat

Bagian ini menjelaskan segmen program saat pemain menekan tombol Space untuk membuat karakter melompat. Terdapat 2 segmen program pada bagian karakter melompat, segmen 5.12 Mode Loncat, dan segmen 5.13 Kontrol Lompat. Penjelasan setiap segmen akan dijabarkan setelah segmen program.

Segmen Program 5.12 Mode Lompat

1. public void modeLompat()
2. {
3. if (!jongkok && !lompat && (isGrounded || upWater))
4. {
5. lompat = true;
6. }
7. }

Segmen Program 5.12 merupakan baris program yang digunakan untuk membuat karakter melakukan aksi loncat. Pada baris 1 merupakan nama function yang dipanggil ketika pemain menekan tombol Space dan berisi perintah hingga baris ke 7. Baris 3 hingga baris 5 berisi pengecekan jika karakter tidak dalam mode merangkak dan tidak lompat, dan sedang berada di atas tanah dan di atas permukaan air, maka pemain dapat melakukan aksi lompat dengan cara mengubah mode lompat menjadi true.

Segmen Program 5.13 Kontrol Lompat

1. if (lompat)
2. {
3. lompat = false;
4. obj\_sound.GetComponent<soundGlobal>().doLompatSound();
5. if (upWater)
6. {
7. rb.velocity = Vector2.up \* 1.5f \* jumpVelocity;
8. obj\_sound.GetComponent<soundGlobal>().dobangkit();
9. }
10. else
11. {
12. rb.velocity = Vector2.up \* jumpVelocity;
13. }
14. }

Segmen Program 5.13 merupakan baris program yang digunakan untuk membuat karakter melakukan aksi loncat. Baris ke 1 berisi perintah pengecekan apakah saat ini status lompat dari karakter. Jika true, maka pada baris ke 3 lompat diganti menjadi false dan mainkan suara karakter lompat. Baris ke 5 hingga baris ke 9 merupakan pengecekan jika karakter berada diair maka akan merubah jumlah tinggi lompatan dan memainkan suara karakter keluar dari air. Pada baris ke 10 hingga baris ke 14 merupakan baris karakter loncat diatas tanah, lompatan dilakukan dengan cara mengganti value rigidbody karakter dengan fungsi vector2.up yang dimodifikasi dengan jumpvelocity yang sudah ditentukan.

1. Karakter Memukul

Bagian ini menjelaskan segmen program saat pemain menekan tombol O untuk membuat karakter memukul rintangan duri dan menghancurkan halangan yang ada. Terdapat 3 segmen program pada bagian karakter memukul yaitu segmen 5.14 Mode Pukul, segmen 5.15 Serangan Jenis Pukul dan segmen 5.16 Kontrol Pukul. Penjelasan setiap segmen akan dijabarkan setelah segmen program.

Segmen Program 5.14 Mode Pukul

1. public void modePukul()
2. {
3. if (!isAttack)
4. {
5. init\_delaySerang = init\_delaySerangPukul;
6. delaySerang = init\_delaySerang;
7. seranganPukul();
8. }
9. }

Segmen Program 5.14 merupakan baris program yang digunakan untuk melakukan aksi memukul. Pada baris 1 merupakan nama function yang dipanggil ketika pemain menekan tombol O dan berisi perintah hingga baris ke 9. Baris ke 3 mengecek apakah pemain saat ini tidak dalam mode memukul, jika lolos pengecekan maka pada baris ke 5 hingga baris ke 6 terdapat perintah untuk memberi value delay serangan dengan delay serangan jenis pukulan yang sudah ditentukan. Kemudian pada baris ke 7 dilakukan pemanggilan function serangan jenis pukulan.

Segmen Program 5.15 Serangan Jenis Pukul

1. private void seranganPukul()
2. {
3. if (!isAttack && isGrounded)
4. {
5. isAttack = true;
6. if (jongkok && isGrounded)
7. {
8. modeSerang = 0.8f;
9. areaPukul.GetComponent

<AreaPukulKarakter>().doPukul();

1. }
2. areaPukul.SetActive(true);
3. an.SetFloat("AttackType", modeSerang);
4. an.SetBool("isAttack", true);
5. }
6. }

Segmen Program 5.15 merupakan baris program yang dipanggil ketika pemain lolos pengecekan function modePukul(). Segmen program ini terdiri atas 15 baris perintah, setiap baris perintah akan dijelaskan pada kalimat berikut. Pada baris 3 dilakukan pengecekan jika pemain tidak sedang menyerang dan sedang berada diatas tanah, maka pada baris ke 5 mode pemain diubah menjadi attack sama dengan true. Kemudian pada baris ke 6 dilakukan pengecekan lagi jika pemain berada dalam mode jongkok dan sedang berada diatas tanah, maka serangan berupa bentuk pukulan pada baris ke 8. Baris ke 9 memanggil skrip AreaPukulKarakter dimana ini fungsinya serupa seperti baris 11 yaitu melakukan pukulan pada deteksi area pukul. Apabila layer yang dipukul terdeteksi sebagai layer rintangan, maka akan dilakukan function doPukul() yang menghilangkan gameobject rintangan yang dipukul. Kemudian pada baris 14 dan 15 memainkan animasi mode serang pukul dari karakter.

Segmen Program 5.16 Kontrol Pukul

1. if (isAttack)
2. {
3. delaySerang = delaySerang - Time.deltaTime;
4. if (delaySerang <= 0)
5. {
6. if (modeSerang == 1 || modeSerang == 0.6f)
7. {
8. modeSerang = 0f;
9. }
10. else
11. {
12. areaPukul.SetActive(false);
13. }
14. isAttack = false;
15. an.SetBool("isAttack", false);
16. delaySerang = init\_delaySerang;
17. }
18. }

Segmen Program 5.16 merupakan baris program yang digunakan untuk melakukan aksi memukul. Baris ke 1 berisi perintah pengecekan apakah saat ini status attack dari karakter yang dimainkan sama dengan true. Jika true, maka pada baris ke 3 delay serang diubah valuenya berdasarkan variable time yang sudah disiapkan. Kemudian pada baris ke 4 dilakukan pengecekan apakah waktu delayserang lebih kecil daripada 0 atau sama dengan 0. Jika iya pada baris ke 6 dilakukan pengecekan lagi apakah mode serang saat ini 1 atau modeserangnya sama dengan 0.6f, jika iya maka pada baris ke 8 mode serangnya direset menjadi 0f. Jika tidak areaPukul difalsekan pada baris ke 12. Kemudian pada baris ke 14 dan 15 is attack dan animasi attack dari karakter difalse kan kembali. Dan pada baris 16 delay serang dikembalikan dengan value init delayserang awal.

1. Karakter Kena Serang

Bagian ini menjelaskan segmen program saat pemain terkena serang oleh hantu atau rintangan yang terdapat pada level. Terdapat 3 segmen program pada bagian karakter terkena serang yaitu segmen 5.17 Kena Serangan, segmen 5.18 Karakter Mati dan segmen 5.19 Karakter Langsung Mati. Penjelasan setiap segmen akan dijabarkan setelah segmen program.

Segmen Program 5.17 Kena Serangan

1. public void kenaSerangan(GameObject paramObj = null)
2. {
3. if (isHurting == false)
4. {
5. isHurting = true;
6. healthPoins -= 1;
7. dirx = 0;
8. if (healthPoins > 0)
9. {
10. if (audioSource.isPlaying && !inWater)
11. {
12. audioSource.Stop();
13. }
14. gameObject.GetComponent

<SpriteRenderer>().color = Color.red;

1. an.SetTrigger("isHurting");
2. rb.velocity = Vector2.zero;
3. if (paramObj.tag == "Jebakan")
4. {
5. if (facingRight)
6. {
7. rb.AddForce(new Vector2(-200f, 200f));
8. }
9. else
10. {
11. rb.AddForce(new Vector2(200f, 200f));
12. }
13. }
14. }
15. else
16. {
17. if (audioSource.isPlaying && !inWater)
18. {
19. audioSource.Stop();
20. }

**Segmen Program 5.17 (Lanjutan)**

1. isDead = true;
2. StartCoroutine("Die");
3. }
4. }
5. }

Segmen Program 5.17 merupakan baris program yang digunakan untuk mendeteksi jika pemain sedang terkena serangan. Pada baris 1 merupakan nama function yang dipanggil ketika pemain terkena serang dengan parameter paramObj yang dinull kan valuenya. Segmen ini berisi perintah hingga baris ke 39. Pada baris ke 2 terdapat pengecekan apakah saat ini pemain sedang terkena serang, jika iya maka akan masuk ke baris 3. Baris ke 3 mengubah value isHurting menjadi true kemudian jumlah darah pemain dikurang 1 pada baris ke 4 dan gerakan karakter pada variable dirx dikembalikan menjadi 0. Pada baris ke 8 dilakukan pengecekan apakah sisa nyawa pemain diatas 0, jika iya maka pada baris ke 10 hingga baris ke 13 bertugas menghentikan suara karakter terkena serang. Baris ke 14 memberikan efek filter merah kepada pemain saat terkena serang. Baris ke 15 digunakan untuk memainkan animasi pemain terkena serang. Baris ke 16 membuat karakter tidak lanjut bergerak kedepan saat terkena serang. Baris ke 17 hingga baris ke 27 berfungsi untuk memberikan efek terpental saat pemain terkena rintangan jebakan, efek terpental diberikan sesuai dengan arah karakter saat itu. Kemudian baris ke 31 hingga baris ke 34 berfungsi sama seperti baris ke 10 hingga baris ke 13, yaitu menghentikan suara setelah pemain terkena serang. Baris ke 35 berfungsi untuk memberi status karakter mati ketika nyawanya mencapai 0. Baris ke 36 menjalankan segmen program Karakter Mati.

Segmen Program 5.18 Karakter Mati

1. IEnumerator Die()
2. {
3. yield return new WaitForSeconds(2f);
4. Debug.Log("karakter mati");
6. }

Segmen Program 5.18 merupakan baris program yang berguna untuk memberi jeda waktu saat karakter mati. Baris ke 1 merupakan penamaan fungsi IEnumerator Die. Baris ke 3 berfungsi untuk memberi jeda 2 detik sebelum perintah baris ke 4 dijalankan. Baris ke 4 berfungsi untuk memberikan status bahwa karakter mati.

Segmen Program 5.19 Karakter Langsung Mati

1. public void langsungMati()
2. {
3. healthPoins = 0;
4. isDead = true;
5. an.SetTrigger("isDead");
6. StartCoroutine("Die");
7. }

Segmen Program 5.19 merupakan baris program yang dipanggil saat karakter ditentukan akan langsung mati tanpa memperhitungkan sisa nyawa yang tersisa. Baris ke 1 merupakan penamaan fungsi langsungMati. Baris ke 3 membuat sisa nyawa pemain menjadi 0, kemudian pada baris ke 4 status pemain mati dijadikan true. Baris ke 5 mengatur agar animasi mati pemain dijalankan. Baris ke 6 menjalankan segmen program Karakter Mati.

1. **Rintangan**

Subbab 5.3 ini akan menjelaskan mengenai segmen program yang berkaitan dengan rintangan yang dimunculkan selama permainan. Segmen program yang dibahas yaitu rintangan hantu, rintangan duri, rintangan jumpscare penampakan, dan tempat bersembunyi.

1. Rintangan Hantu

Bagian ini menjelaskan rintangan hantu berkerja dalam game. Rintangan jenis ini memiliki 3 segmen program yaitu segmen program 5.20 Generate Pijakan Hantu, 5.21 Cek Penginjakan, dan 5.22 Memunculkan Hantu. Penjelasan setiap segmen akan dijabarkan setelah segmen program.

Segmen Program 5.20 Generate Pijakan Hantu

1. public void randomjalan\_PCG(int x, int id)
2. {
3. int min = 0;
4. int max = 7;

Segmen Program 5.20 (Lanjutan)

1. int randomValue = Random.Range(min, max);
3. if (randomValue == id)
4. {
5. randomValue++;
6. randomjalan\_PCG(x, id);
7. }
8. else
9. {
10. if (randomValue == 0)
11. {
12. jumlahJalanHantu--;
13. if (jumlahJalanHantu >= 0)
14. {
15. buatObject(jalan\_hantu\_single, x, tinggi,

layer\_jalan);

1. }
2. else
3. {
4. randomjalan\_PCG(x, 0);
5. }
6. }
7. else
8. {
9. buatObject(rumput, x, tinggi, layer\_jalan);
10. }
11. }
12. }

Segmen Program 5.20 merupakan baris program yang digunakan untuk membuat balok platform pijakan hantu. Baris ke 1 penamaan fungsi random jalan platform. Baris ke 3 hingga baris ke 5 merupakan deklarasi variable pengacak. Baris ke 9 mengecek apakah hasil random sama dengan value variable id, jika sama maka random value akan ditambah 1 pada baris ke 11 dan akan menjalankan fungsi randomjalan\_PCG sekali lagi. Jika hasil random tidak sama dengan value variable id, maka akan dilakukan pengecekan pada baris ke 16, apakah hasil random merupakan angka 0, jika iya maka variable jumlahJalanHantu akan dikurang 1 pada baris 18. Kemudian pada baris ke 19 di cek, apakah jumlahJalanHantu nilainya masih diatas atau sama dengan 0, jika iya maka akan membuat object jalan hantu single pada baris ke 21 dengan memanggil fungsi buatObject, jika tidak maka akan menjalankan fungsi randomjalan\_PC dengan value id 0. Jika hasil random bukan 0 maka perintah baris ke 29 dijalankan, yaitu memanggil fungsi buatObject yang diisi dengan gameobject platform blok rumput.

Segmen Program 5.21 Cek Penginjakan

1. private void OnTriggerEnter2D(Collider2D collision)
2. {
3. penangkapEmosi.deteksi\_emosi();
4. if (useItems\_ != null)
5. {
6. if (useItems\_.GetComponent<glob\_Items>

().getPakaiGunting() == true || sudahDipijak == true){

1. }
2. else
3. {
4. if (collision.gameObject.CompareTag

("Player") && sudahDipijak == false){

1. if (hantu != null){
2. if (hantu.GetComponent

<Hantu\_single>() != null && hantu.GetComponent

<Hantu\_single>().startMuncul == false){

1. hantu.GetComponent<Hantu\_single>().munculHantu();
2. sudahDipijak = true;
3. }
4. }
5. }
6. }
7. }
8. }

Segmen Program 5.21 merupakan baris program yang digunakan untuk melakukan cek penginjakan blok oleh karakter. Baris ke 3 mencatat emosi saat pemain melakukan penginjakan. Baris ke 4 melakukan pengecekan apakah pemain memiliki item. Jika lolos pengecekan maka perintah baris ke 6 dilakukan pengecekan apakah pemain menggunakan item gunting saat pijakan terinjak, jika iya maka akan hantu tidak akan muncul. Pada baris ke 10 terdapat pengecekan apakah pemain menginjak pijakan saat kondisi pijakan tidak pernah diinjak, jika iya maka dilakukan pengecekan pada baris ke 12. Baris ke 12 melakukan pengecekan apakah hantunya tidak null, jika iya maka masuk baris perintah ke 13 yang melakukan pengecekan apakah Hantu\_single sudah diisi gameobject dan sedang tidak muncul. Ketika lolos penecekan maka baris ke 15 memanggil fungsi munculHantu() pada gameobject Hantu\_single yang sudah memiliki script dengan fungsi munculHantu(), kemudian variable sudahDipijak diberi value true.

Segmen Program 5.22 Memunculkan Hantu

1. public void munculHantu()
2. {
3. doAttack = true;
4. gameObject.transform.position = karakter.transform.position;
5. if (camera != null){
6. camera.GetComponent<Camera\_Karakter>().SetFovKarakterDekat();
7. }
8. startMuncul = true;
9. poinLight.SetActive(true);
10. karakter\_hantu.SetActive(true);
11. int min = 0;
12. int max = 100;
13. int randomValue = Random.Range(min, max);
14. if (randomValue % 2 != 0 ){
15. nempelPemain = true;
16. randomValue = Random.Range(min, max);
17. if (randomValue % 2 != 0 ){
18. if (karakter.transform.localScale.x > 0){
19. karakter\_hantu.transform.localScale = new Vector3(-0.3f, 0.3f, 0.3f);
20. karakter\_hantu.transform.parent = karakter.transform;
21. karakter\_hantu.transform.position = new Vector2(karakter\_hantu.transform.position.x - 1.5f, karakter\_hantu.transform.position.y+0.5f);
22. poinLight.transform.parent = karakter.transform;
23. }
24. else{
25. karakter\_hantu.transform.localScale = new Vector3(0.3f, 0.3f, 0.3f);
26. karakter\_hantu.transform.parent = karakter.transform;
27. karakter\_hantu.transform.position = new Vector2(karakter\_hantu.transform.position.x + 1.5f,karakter\_hantu.transform.position.y+0.5f);
28. poinLight.transform.parent = karakter.transform;
29. }
30. }
31. }
32. else{
33. if (karakter.transform.localScale.x < 0){
34. karakter\_hantu.transform.localScale = new Vector3(-0.3f, 0.3f, 0.3f);
35. karakter\_hantu.transform.parent = karakter.transform;
36. karakter\_hantu.transform.position = new Vector2(karakter\_hantu.transform.position.x - 2f, karakter\_hantu.transform.position.y);
37. poinLight.transform.parent = karakter.transform;
38. }

**Segmen Program 5.22 (Lanjutan)**

1. else
2. {
3. karakter\_hantu.transform.localScale = new Vector3(0.3f, 0.3f, 0.3f);
4. karakter\_hantu.transform.parent = karakter.transform;
5. karakter\_hantu.transform.position = new Vector2(karakter\_hantu.transform.position.x + 2f, karakter\_hantu.transform.position.y);
6. poinLight.transform.parent = karakter.transform;
7. }
8. }
9. else
10. {
11. nempelPemain = false;
12. }
13. }

Segmen Program 5.22 merupakan baris program yang digunakan untuk memunculkan hantu setelah karakter menginjak blok pijakan berisi hantu. Baris ke 1 merupakan penamaan fungsi munculHantu(). Baris ke 3 mengatur doAttack menjadi true dan baris ke 4 mengubah posisi gameobject sesuai dengan posisi karakter saat itu. Baris ke 5 mengatur pengecekan, jika kamera tidak null maka baris ke 6 mengatur fov dari kamera untuk menyempit disekitar karakter. Baris ke 8 mengubah variable muncul hantu menjadi true, kemudian memberi sorot point light pada baris ke 9 pada karakter hantu yang diatur pada baris ke 10. Baris ke 11 hingga baris ke 13 berguna untuk melakukan random value untuk memunculkan hantu jenis apa yang keluar, apakah model menempel dibelakng atau muncul didepan karakter. Jika hasil random lolos pengecekan baris ke 17, maka baris perintah ke 18 akan mengecek lagi apakah posisi karakter sedang menghadap kekiri, jika iya maka baris ke 19 hingga baris ke 21 akan mengatur pergerakan hantu kekiri dengan cara mengurangi value x. Sedangkan baris ke 22 merupakan pengaturan pointLight pada hantu yang mengikuti karakter. Sedangkan baris perintah ke 24 hingga ke 28, merupakan mekanisme pergerakan hantu jika karakter sedang menghadap kekanan. Value x pada transform akan ditambah pada baris perintah ke 25. Sedangkan baris ke 32 hingga baris ke 45, merupakan baris perintah untuk memunculkan hantu yang berada dibelakang pemain. Untuk pergerakannya diatur pada baris ke 33 jika pemain menghadap kekiri, makan baris perintah ke 34 hingga baris ke 36 mengatur posisi x agar berkurang. Sedangkan jika pemain menghadap kanan maka baris perintah ke 39 akan dijalankan. Pengaturan gerak hantu diatur pada baris ke 41 hingga baris ke 43, dimana posisi x akan ditambah. Dan yang terakhir yaitu baris ke 47 jika hasil random yang dihasilkan tidak memenuhi syarat penyaringan, maka jenis hantu yang keluar hanya diam ditempat blok pijakan berhantu dengan cara memberi value false pada baris perintah ke 49.

1. Rintangan Duri

Bagian ini menjelaskan rintangan duri yang muncul didalam game. Rintangan jenis ini memiliki 2 segmen program yaitu segmen program 5.23 Generate Rintangan Duri, dan 5.24 Pemain Terkena Duri. Penjelasan berada diakhir segment.

Segmen Program 5.23 Generate Rintangan Duri

1. public void PCG\_Jebakan()
2. {
3. int min = 0;
4. int max = 10;
5. int randomValue = Random.Range(min, max);
6. if (randomValue % 2 != 0)
7. {
8. index = 0;
9. }
10. else
11. {
12. index = 1;
13. }
14. sp.sprite = image\_jebakan[index];
15. randomValue++;
16. }

Segmen Program 5.23 merupakan baris program yang digunakan untuk membuat rintangan duri pada platform level. Baris ke 1 merupakan penamaan fungsi untuk membuat platform duri. Baris ke 3 hingga baris ke 5 merupakan variable untuk merandom keputusan. Baris ke 6 merupakan pengecekan value variable random pada baris ke 3 hingga baris ke 5, jika lolos pengecekan maka index atau jenis duri akan dimunculkan sesuai dengan baris perintah ke 8. Baris perintah ke 10 hingga ke 13 mengatur index atau jenis duri yang akan dimunculkan, dengan mengubah value index menjadi 1 pada baris ke 12. Kemudian baris perintah ke 14 memunculkan sprite duri sesuai dengan index yang didapat sebelumnya. Terakhir baris ke 15 membuat value random ditambah 1.

Segmen Program 5.24 Pemain Terkena Duri

1. private void OnTriggerEnter2D(Collider2D collision)
2. {
3. if (collision.gameObject.CompareTag("Player"))
4. {
5. penangkapEmosi.deteksi\_emosi();
6. karakter.GetComponent<Karakter>().

kenaSerangan(gameObject);

1. gameObject.SetActive(false);
2. }
3. }

Segmen Program 5.24 merupakan baris program yang digunakan untuk mendeteksi apakah pemain terkena rintangan duri. Baris ke 3 merupakan pengecekan apakah duri mengalami collision dengan gameobject dengan tag “Player” yang sudah dipasang pada karakter. Jika iya maka perintah baris ke 5 hingga baris ke 7 akan dijalankan. Baris ke 5 menyimpan deteksi wajah pemain saat terkena rintangan duri. Baris ke 6 memberi atribut kenaSerangan pada karakter. Dan kemudian pada baris ke 7 gameobject duri akan dihilangkan jika telah diinjak karakter pemain.

1. Jumpscare Penampakan

Bagian ini menjelaskan tentang blok jumpscare penampakan didalam game. Rintangan jenis ini memiliki 2 segmen program yaitu segmen program 5.25 Generate Rintangan Jumpscare Penampakan, dan 5.26 Pemain Menginjak Jumpscare Penampakan. Penjelasan setiap segmen akan dijabarkan setelah segmen program.

Segmen Program 5.25 Generate Rintangan Jumpscare Penampakan

1. public void PCG\_kuburan\_sesajen()
2. {
3. int min = 0; // Nilai terkecil yang ingin Anda hasilkan (termasuk)
4. int max = image\_kuburan\_sesajen.Length-1;
5. int randomValue = Random.Range(min, max);
6. sp.sprite = image\_kuburan\_sesajen[randomValue];
7. randomValue++;
8. }

Segmen Program 5.25 merupakan baris program yang digunakan untuk membuat rintangan jumpscare penampakan pada platform level. Baris ke 1 merupakan penamaan fungsi untuk membuat platform jumpscare. Baris ke 3 hingga baris ke 5 merupakan variable untuk merandom bentuk sprites kuburan yang akan muncul pada level. Baris ke 6 digunakan mengenerate blok jumpscare penampakan berdasarkan random value yang didapat dari baris ke 3 hingga baris ke 5. Kemudian baris ke 7 digunakan untuk mengubah nilai random value.

Segmen Program 5.26 Pemain Menginjak Jumpscare

1. private void OnTriggerEnter2D(Collider2D collision)
2. {
3. if (collision.gameObject.CompareTag("Player") && useItems\_.GetComponent<glob\_Items>().getPakaiGunting() == false)
4. {
5. canvasJumpScare.GetComponent<CanvasJumpScare>().doJumpScare();
6. gameObject.GetComponent<BoxCollider2D>().enabled = false;
7. status = "Aktif";
8. pencatatEmosi.RecordData();
9. }
10. }

Segmen Program 5.26 merupakan baris program yang mengecek apakah pemain menginjak blok jumpscare penampakan. Baris ke 3 merupakan pengecekan apakah blok jumpscare mengalami collision dengan gameobject dengan tag “Player” yang sudah dipasang pada karakter dan status player sedang tidak menggunakan item gunting. Jika lolos pengecekan, maka baris ke 5 dijalankan yaitu memainkan jumpscare berdasarkan canvas jumpscare. Kemudian baris ke 6 melakukan disable gameobject blok jumpscare ketika sudah diinjak.. Kemudian memberi status telah aktif pada baris ke 7 guna diolah oleh pencatat emosi pada baris ke 8.

1. Tempat Bersembunyi

Bagian ini menjelaskan tempat bersembunyi yang sekaligus menjadi tempat yang aman bagi karakter pemain yang berbentuk semak bambu didalam game. Tempat sembunyi memiliki 2 segmen program yaitu segmen program 5.27 Generate Platform Bambu, dan 5.28 Pemain Bersembunyi. Penjelasan setiap segmen akan dijabarkan setelah segmen program.

Segmen Program 5.27 Generate Platform Bambu

1. public void PCG\_Bambu()
2. {
3. int min = 0;
4. int max = 10;
5. int randomValue = Random.Range(min, max);
6. if (randomValue % 2 != 0)
7. {
8. index = 0;
9. }
10. else
11. {
12. index = 1;
13. }
14. sp.sprite = image\_bambu[index];
15. randomValue++;
16. }

Segmen Program 5.27 merupakan baris program yang digunakan untuk membuat tempat persembunyian atau tempat aman pada platform level. Baris ke 1 merupakan penamaan fungsi untuk membuat platform tempat sembunyi. Baris ke 6 merupakan pengecekan value variable random pada baris ke 3 hingga baris ke 5, jika lolos pengecekan maka index atau jenis duri akan dimunculkan sesuai dengan baris perintah ke 8. Baris perintah ke 10 hingga ke 13 mengatur index atau jenis duri yang akan dimunculkan, dengan mengubah value index menjadi 1 pada baris ke 12. Kemudian baris perintah ke 14 memunculkan sprite bambu sesuai dengan index yang didapat sebelumnya. Terakhir baris ke 15 membuat value random ditambah 1.

Segmen Program 5.28 Pemain Bersembunyi

1. if (masuk == true)
2. {
3. if (index == 1)
4. {
5. if (karakter.GetComponent<Karakter>().jongkok == true){
6. karakter.GetComponent<Karakter>().setStatusHide(true);
7. }
8. else{
9. karakter.GetComponent<Karakter>().setStatusHide(false);
10. }

Segmen Program 5.28 (Lanjutan)

1. }
2. else
3. {
4. karakter.GetComponent<Karakter>().setStatusHide(true);
5. }
6. else
7. {
8. karakter.GetComponent<Karakter>().setStatusHide(false);
9. }

Segmen Program 5.x merupakan baris program yang mengecek apakah pemain statusnya sedang masuk didalam semak bambu. Baris ke 1 mendeteksi pemain dengan cara mengecek status trigger pemain apakah sedang true. Jika true, maka masuk di baris ke 3 yaitu pengecekan tipe bambu, jika bambu yang muncul dideteksi bambu pendek maka akan masuk baris perintah ke 5. Baris perintah ke 5 mengecek apakah karakter sedang jongkok, jika iya maka baris ke 6 mengubah statusHide karakter menjadi true. Jika tidak maka statusHide karakter diganti menjadi false pada baris ke 9. Kemudian pada baris ke 14 akan dijalankan jika bambu yang muncul berupa bambu panjang, jika karakter pemain mentrigger bambu maka statusHide pemain akan dijadikan true. Jika karakter statusnya tidak lolos pengecekan pada baris ke 1 maka statusHide karakter pada baris ke 18 akan diubah menjadi false.