BAB VI

IMPLEMENTASI MAP GENERATION MENGGUNAKAN GENETIC ALGORITHM

Menggunakan GeneticSharp oleh Giacomelli, bab ini akan menjelaskan dan menyelam menuju aspek Map Generationnya. Dalam bab ini, akan dijelaskan seluruh aspek Map Generation dari game Splatted, dimulai dari kromosom hingga seluruh fitness yang dipakai.

1. Chromosome / Representasi

Dimulai dengan dasar dari sebuah Genetic Algorithm, yaitu sebuah kromosom, dan di subbab ini akan dijelaskan implementasi Kromosom yang akan dipakai sebagai perantara solusi dalam Genetic Algorithm kita.

1. Tile Chromosome

Berikut adalah potongan kode untuk membuat kromosom tile generation :

Segmen Program 6.1 Class TileChromosome

1. public class TileChromosome : ChromosomeBase{
2. private readonly int m\_ukuranMap;
3. private readonly int m\_powerup;
4. public GameChromosome(int ukuranMap,int powerup) : base(ukuranMap){
5. int temp;
6. m\_ukuranMap = ukuranMap;
7. m\_powerup = powerup;
8. var mapValues = RandomizationProvider.Current.GetInts (ukuranMap, 0, 2);
9. for (int i = 0; i < ukuranMap; i++)
10. ReplaceGene(i, new Gene(mapValues[i]));
11. for (int i = 0; i < 5; i++){
12. temp = Mathf.FloorToInt(Random.Range(0, ukuranMap));
13. while (mapValues[temp] == 3)
14. temp = Mathf.FloorToInt(Random.Range(0, ukuranMap));
15. ReplaceGene(temp, new Gene(3));
16. }
17. for (int i = 0; i < powerup; i++){

Segmen Program 6.1 (Lanjutan)

1. temp = Mathf.FloorToInt(Random.Range(0, ukuranMap));
2. while (mapValues[temp] == 3 || mapValues[temp] == 2)
3. temp = Mathf.FloorToInt(Random.Range(0, ukuranMap));
4. ReplaceGene(temp, new Gene(2));
5. }
6. }
7. public override Gene GenerateGene(int geneIndex){
8. return new Gene(RandomizationProvider.Current.GetInt(0, 3));
9. }
10. public override IChromosome CreateNew(){
11. return new GameChromosome(m\_ukuranMap,m\_powerup);
12. }
13. public override IChromosome Clone(){
14. var clone = base.Clone() as GameChromosome;
15. return clone;
16. }
17. }

Penjelasan untuk masing masing baris kode :

* 1 : TileChromosome menggunakan class ChromosomeBase yang telah disediakan oleh GeneticSharp untuk membuat kromosom yang bisa dipakai untuk Genetic Algorithm nanti.
* 3–4&8-9 : Jujur ini ada karena wiki memasukkan ini di dalam tutorial, jadi kode ini tidak pasti memiliki kontribusi, tetapi dimasukkan aja agar aman.
* 6 : Constructor untuk membuat kromosom baru dengan parameter ukuran map yang berisi luas dari level dan powerup yang berisi jumlah powerup yang berada dalam map ini.
* 11 : Variabel mapValues akan diisi dengan sebuah array integer yang berukuran luas level dan akan diisi oleh angka dengan range 0 – 2.
* 12 – 13 : Fungsi ini adalah untuk mengisi Kromosom dari class ini (yang berada di class ChromosomeBase) dengan MapValues.
* 14 – 19 : Mengubah 5 gene dalam kromosom menjadi integer 3, dimana 3 adalah representasi atas seorang karakter, dan ini dilakukan untuk mempermudah proses Generasi.
* 20 – 25 : Sama seperti proses diatas, hanya ketimbang mengganti gene menjadi 3, gene diganti menjadi 2 yang merepresentasikan bola spesial di level.
* 28-30 : Override function yang dipakai untuk Membuat gene baru, sepertinya dipakai dalam beberapa Mutation yang disediakan.
* 32 – 33 : Fungsi yang dipakai untuk membuat kromosom baru? Sulit melihat kegunaan fungsi ini karena terdapat Clone beberapa baris dibawah.
* 36 – 38 : Fungsi ini dipakai untuk menduplikat kromosom, dimana fungsi ini tampaknya krusial dalam fase seleksi untuk menduplikat kromosom ke generasi baru.

1. Templated Map Chromosome

Sekarang setelah melihat kode Tile Chromosome, akan ditunjukkan kode yang digunakan untuk membuat Template Map Chromosome, tetapi sebelum itu, kromosom ini menggunakan class lain yang dipakai untuk mengganti angka – angka dalam gene menjadi template 5X5, berikut adalah class tersebut :

Segmen Program 6.2 Class PossibleTemplates

1. public static class PossibleTemplates{
3. public static int[][,] oneWayTemplates = new int[][,]{…}
4. public static int[][,] twoWayTemplates = new int[][,]{…}
5. public static int[][,] fourWayTemplates = new int[][,]{…}
7. public static int getTemplateAmount(){
8. return oneWayTemplates.Length + twoWayTemplates.Length \* 2 + fourWayTemplates.Length \* 4;
9. }
10. public static int[,] getTemplate(int id){
11. int tempID = id;
12. if (tempID < 0)
13. tempID = -tempID - 1;
14. int rotation;
15. int[,] chosenTemplate;
16. if (tempID < oneWayTemplates.Length){
17. rotation = 0;
18. chosenTemplate = oneWayTemplates[tempID];
19. }

Segmen Program 6.2 (Lanjutan)

1. else if (tempID < oneWayTemplates.Length + twoWayTemplates.Length \* 2){
2. tempID -= oneWayTemplates.Length;
3. rotation = tempID % 2;
4. chosenTemplate = twoWayTemplates[Mathf.FloorToInt(tempID / 2)];
5. }
6. else{
7. tempID = tempID - oneWayTemplates.Length - (twoWayTemplates.Length \* 2);
8. rotation = tempID % 4;
9. chosenTemplate = fourWayTemplates[Mathf.FloorToInt(tempID / 4)];
10. }
11. int[,] resultTemplate = new int[5, 5];
12. if (rotation == 0)
13. resultTemplate = (int[,])chosenTemplate.Clone();
14. else if (rotation == 1)
15. //rotasi ke kanan
16. for (int i = 0; i < 5; i++)
17. for (int j = 0; j < 5; j++)
18. resultTemplate[j, 4 - i] = chosenTemplate[i, j];
19. else if (rotation == 2)
20. //rotasi ke 180 derajat
21. for (int i = 0; i < 5; i++)
22. for (int j = 0; j < 5; j++)
23. resultTemplate[4 - i, 4 - j] = chosenTemplate[i, j];
24. else
25. //rotasi ke kiri
26. for (int i = 0; i < 5; i++)
27. for (int j = 0; j < 5; j++)
28. resultTemplate[4 - j, i] = chosenTemplate[i, j];
29. // kalau nomor yang diberi negatif beri power
30. if (id < 0 && resultTemplate[2, 2] == 0)
31. resultTemplate[2, 2] = 2;
32. return resultTemplate;
33. }

Sekarang untuk penjelasan mengenai kode diatas :

* 3 – 5 : Variabel one,two dan four way template akan diisi secara manual apa template – template yang bisa dipakai oleh sistem berdasarkan jumlah orientasi berbeda yang dapat dimiliki sebuah template. Alternatif lain adalah menempatkan template sebagai file external, tetapi dimasukkan dalam script secara hard code juga bekerja karena game tidak memerlukan daftar template yang bisa diganti.
* 7 – 9 : Fungsi ini dipakai untuk mengambil jumlah potensial dari seluruh template, dimana four way dikali 4, two way dikali 2 dan one way tidak diubah, ketiga angka tersebut dijumlah dan di return sebagai jumlah potensial template yang tersedia.
* 11 – 12 : Ini adalah fungsi untuk mengambil template berdasarkan dari id yang diberikan. Setelah memberi id tersebut, id akan disimpan ke dalam variabel tempID agar bisa diolah.
* 13 – 14 : Bila tempID bernilai negatif, maka tempID dimutlakkan lalu dikurangi satu. Bile tempID bernilai negatif, itu menandakan bila template tersebut bisa memiliki sebuah bola spesial di tengah template tersebut, jadi karena nilai negatif sudah disimpan di parameter id, maka tempID bisa diubah positif, lalu tempID dikurangi 1 karena ada 0 tetapi tidak ada -0, sehingga ini bekerja sebagai alternatif -0 tersebut.
* 19 – 32 : Di sini akan diambil berdasarkan tempID sebuah template yang akan direturn setelah diolah. Akan ada 2 variabel yang diisi, yaitu rotation yang mengatakan seberapa banyak rotasi template yang akan dilakukan nanti dan chosen template yang berisi template tersebut.

Tahap pertama dari metode ini adalah untuk melihat apabila tempID masuk dalam range one way, two way atau four way. Setelah ditentukan, tempID dikurangi dengan jumlah variabel dibawahnya, semisal bila tempID memasuki range four way dimana four way adalah yang paling terkhir diakses, maka tempID akan dikurangi panjang array one way dan two way dikali 2 (karena setiap isi array two way bisa diorientasikan 2 arah).

Dan setelah mendapat tempID, tempID tersebut akan di DIV oleh jumlah orientasi range array, bila kasus diatas maka akan didiv 4, lalu dari tempID div 4 maka akan diambil Template dari array template yang bersangkutan dan memasukkan template tersebut ke dalam variabel chosen template. Dan terakhir untuk menentukan rotasi dari template, rotation akan diisi dengan tempID di MOD dengan jumlah orientasi range array.

* 34 – 51 : Meski panjang, bagian ini sederhana, dimana template dirotasi sebanyak 90˚ \* variabel rotation ke arah kanan.
* 53 – 54 : Bila parameter ID bernilai negatif dan tile di paling tengah bernilai 0 (tidak ada apa apa), maka tile di tengah diberi bola spesial.

Dengan Class PossibleTemplates dijelaskan, maka kita bisa masuk ke class TemplatedChromosome, sebuah Chromosome berbasis template :

Segmen Program 6.3 Templated Map Chromosomes

1. public class TemplatedMapChromosome : ChromosomeBase{
2. private readonly int m\_ukuranMap;
3. public TemplatedMapChromosome(int ukuranMap) : base (ukuranMap){
4. m\_ukuranMap = ukuranMap;
5. var mapValues = RandomizationProvider.Current.GetInts(ukuranMap, -(PossibleTemplates.getTemplateAmount()), (PossibleTemplates.getTemplateAmount()) - 1);
6. for (int i = 0; i < ukuranMap; i++)
7. ReplaceGene(i, new Gene(mapValues[i]));
8. }
9. public override Gene GenerateGene(int geneIndex){
10. return new Gene(RandomizationProvider.Current.GetInt(-(PossibleTemplates.getTemplateAmount()) , (PossibleTemplates.getTemplateAmount())-1));
11. }
12. public override IChromosome CreateNew(){
13. return new TemplatedMapChromosome(m\_ukuranMap);
14. }
15. public override IChromosome Clone(){
16. var clone = base.Clone() as TemplatedMapChromosome;
17. return clone;
18. }
19. }

Dibandingkan Segmen Program 6.1 diatas kode ini kurang lebih sama dengan perbedaan utama terdapat di line 7, dimana MapValues ketimbang diisi 0-3 akan diisi dengan range -JumlahTemplate hingga JumlahTemplate-1. Selain itu, tidak terdapat perbedaan lain, mungkin salah satu perbedaan yang lain adalah ukuran map yang didapat oleh Kromosom ini adalah JumlahArea / 25, 25 merupakan luas area 5x5.

1. Main Code

Setelah Chromosome template dan tile, sekarang adalah kode yang digunakan untuk menjalankan Genetic Algorithm tersebut :

Segmen Program 6.4 Genetic Algorithm

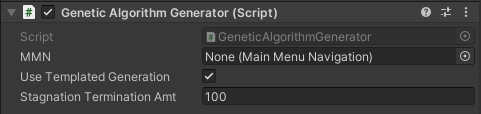
1. bool useMirrorFitness = true;
2. [SerializeField] MainMenuNavigation MMN;
3. [SerializeField] bool useTemplatedGeneration;
4. [SerializeField] int StagnationTerminationAmt;
5. InLoopFitnessBase[] fitnesses;
6. void algorithmStart(){
7. fitnesses = GetComponents<InLoopFitnessBase>();
8. useTemplatedGeneration = MainMenuNavigation.isTemplate;
9. for (int i = 0; i < fitnesses.Length; i++){
10. //Ngilangin Fitness yang ngga dipakai
11. if (!fitnesses[i].IsUsed || (useTemplatedGeneration && !fitnesses[i].ForTemplateGen) || (!useTemplatedGeneration && !fitnesses[i].ForTileGen))
12. fitnesses[i] = null;
13. }
14. fitnesses = fitnesses.Where(f => f != null).ToArray();
15. double fitness;
17. int generatedMapLength = SetObjects.getHeight() \* SetObjects.getWidth() / 2;
18. //Multithreading
19. var taskExecutor = new ParallelTaskExecutor();
20. taskExecutor.MinThreads = 12;
21. taskExecutor.MaxThreads = 12;
22. ChromosomeBase chromosome;
23. //Kromosom
24. if (useTemplatedGeneration)
25. chromosome = new TemplatedMapChromosome(Mathf.RoundToInt(generatedMapLength / 25));
26. else{
27. chromosome = new GameChromosome(generatedMapLength, Mathf.FloorToInt(GetComponent<PowerUpRatioFitness>().getRatio() \* generatedMapLength/2));
28. }
29. //Populasi
30. var population = new Population(50, 100, chromosome);
31. //Fitness
32. var fitnessfunc = new FuncFitness((c) =>{
33. var fc = c.GetGenes();
34. int[,] map;
35. //Menghasilkan map utuh yang siap diperiksa oleh fitness
36. if (useTemplatedGeneration)

Segmen Program 6.4 (Lanjutan)

1. map = putPlayerinTemplate(fc, SetObjects.getWidth() / 2, SetObjects.getHeight());
2. else
3. map = deflatten(fc, SetObjects.getWidth() / 2, SetObjects.getHeight());
4. fitness = fitnessFunction(map, fc);
5. return fitness;
6. });
7. //Metode milih ortu
8. var selection = new RouletteWheelSelection();
9. //Metode Crossover
10. var crossover = new UniformCrossover();
11. var mutation = new PartialShuffleMutation();
12. var termination = new FitnessStagnationTermination(StagnationTerminationAmt);
13. var ga = new GeneticAlgorithm(population, fitnessfunc, selection, crossover, mutation);
14. ga.Termination = termination;
15. ga.Start();
16. var a = ga.BestChromosome.GetGenes();
17. if (useTemplatedGeneration)
18. SetObjects.setMap(putPlayerinTemplate(a, SetObjects.getWidth() / 2, SetObjects.getHeight()), useMirrorFitness);
19. else
20. SetObjects.setMap(deflatten(a, SetObjects.getWidth() / 2, SetObjects.getHeight()), useMirrorFitness);
21. MMN.changeSceneIndex(-6);
22. }

Berikut penjelasannya :

* 1 : Mengatakan kepada GA bila fitness diukur berdasarkan hasil akhirnya, karena hasil generasi dari GA hanyalah sebelah kiri dari sebuah level, yang nantinya akan dicerminkan di sebelah kanan, membuat level yang jadi.
* 2 : Referensi Script MainMenuNavigation agar setelah Genetic Algorithm selesai bisa berganti level
* 3 : Menentukan apabila Genetic Algorithm menggunakan Template atau Tile Generation
* 4 : Menentukan berapa generasi tanpa perkembangan fitness sebelum Genetic Algorithm menyerah dalam menyari kromosom terbaik.
* 5 : Variabel yang akan menampung semua fitness
* 8 : Mengambil semua fitness yang tersedia untuk dipakai
* 9 : Mengambil data apabila player memilih Map Template atau tile
* 10 – 15 : Mengecek semua fitness yang ada di variabel fitnesses, lalu menghapus fitness yang ditentukan tidak cocok dengan permintaan player.
* 19 : Menentukan variabel MapLength yang berisi luas level / 2 dikarenakan itu adalah ketentuan jumlah Gene dalam sebuah kromosom.
* 21 - 24 : Digunakan untuk melakukan Multirhreading terhadap Genetic Algorithm, tetapi kurang diketahui bila potongan kode ini bekerja atau tidak.
* 28 – 32 : Mengecek apabila player meminta Template atau Tile level generation, dan membuat kromosom sesuai dengan permintaan player tersebut.
* 35 : Menentukan jumlah populasi dari kromosom yang ditentukan. Dalam game ini ditentukan populasi awal adalah 50 sementara populasi maksimum adalah 100.
* 37 – 47 : Menentukan fungsi fitness yang akan dipakai pada Genetic Algorithm.
* 38 – 44 : Variabel fc akan diisi dengan seluruh Gene dalam kromosom, lalu gene tersebut akan diubah menjadi sebuah array level yang berbentuk 2D sesuai dengan input player
* 45 – 46 : Variabel fitness akan diisi skor fitness menggunakan function fitnessFunction yang akan dijelaskan pada subbab berikutnya, lalu fitness itu akan direturn untuk dilakukan tahap seleksi.
* 49 : Menyuruh GA untuk memakai Roulette Wheel Selection.
* 52 : Menyuruh GA untuk memakai Uniform Crossover.
* 53 : Sama kayak diatas, GA disuruh memakai Partial Shuffle Mutation.
* 54 : Sama aja, GA memakai Fitness Stagnation Termination dengan toleransi sebanyak yang ditentukan StagnationTerminationAmt yang ada di baris 4.
* 56-57 : Menggabung semua aspek dari Genetic Algorithm (Populasi yang dibuat, Crossover yang dipilih dst) menjadi satu Genetic Algorithm yang siap dipakai.
* 59 : Genetic Algorithm mulai dijalankan.
* 61 : Variabel a akan diisi Kromosom terbaik yang dihasilkan oleh Genetic Algorithm
* 62 – 65 : Menggunakan script SetObjects, ubah kromosom yang didapat pada line 61 menjadi sebuah map jadi, sekali lagi menggunakan function berbeda sesuai dengan pilihan level generation player.
* 66 : Setelah semuanya selesai, maka akan di load scene level utama agar player bisa memainkan level yang degenerate.



Gambar 6.1

Isi Variabel dari segmen program 6.4

Selanjutnya adalah variabel yang dipakai untuk baris 2 – 4 dari segmen program diatas. MMN adalah script Main Menu Navigation yang akan diisi Gameobject Mengandung Script Main Menu Navigation di level tersebut, sehingga secara default null. Lalu Use Templated Generation, variabel ini secara default true, tetapi dapat diubah oleh user melalui menu pilihan. Terakhir adalah Stagnation Termination Amt yang menandakan bila GA akan berhenti setelah 100 generasi tanpa perkembangan yang bernilai pada fitness maksimal.

1. Function FitnessFunction

Sekarang untuk menjelaskan masing – masing fitness yang tersedia, setiap fitness menggunakan base Class FitnessFunction, yang berfungsi sebagai basis dari setiap fitness yang dibuat. Berikut adalah isi dari fungsi fitnessFunction :

Segmen Program 6.5 Fungsi FitnessFunction

1. double fitnessFunction(int[,] map , Gene[] original){
2. int playerAmount = 0;
3. InLoopFitnessBase[] currentFitnesses = (InLoopFitnessBase[])fitnesses.Clone();
4. //Khusus Template Variety
5. for (int i = 0; i < currentFitnesses.Length; i++)
6. if (currentFitnesses[i].GetType() == GetComponent<TemplateVarietyFitness>().GetType())
7. ((TemplateVarietyFitness)currentFitnesses[i]). getTemplateMap(original);
9. double[] fitnessScores = new double[fitnesses.Length];
10. foreach (InLoopFitnessBase item in currentFitnesses)
11. item.resetVariables();
12. Coordinate tempCoor;
13. for (int i = 0; i < SetObjects.getHeight(); i++)
14. for (int j = 0; j < SetObjects.getWidth(); j++) {
15. tempCoor = new Coordinate(j, i);
16. if (map[tempCoor.yCoor, tempCoor.xCoor] == 3)
17. playerAmount++;
18. foreach (InLoopFitnessBase item in currentFitnesses)
19. item.calculateFitness(map,tempCoor);
20. }
21. for (int i = 0; i < currentFitnesses.Length; i++)
22. fitnessScores[i] = currentFitnesses[i].getFitnessScore();
23. return Mathf.Pow((float)fitnessScores.Sum() / Mathf.Pow(MathF.Abs(playerAmount - 10) \* 5 + 1, 3), 3);
24. }

Berikut adalah penjelasannya setiap baris :

* 1 : Function ini memerlukan 2 parameter, yaitu map jadi level dan juga Kromosom originalnya.
* 3 : Dari variabel global fitnesses yang sebelumnya telah diisi di baris 10 – 15 kode utama Genetic Algorithm, diduplikat fitness tersebut ke dalam variabel currentFitnesses agar setiap kromosom memiliki perhitungan fitness yang tidak saling mengganggu.
* 6 – 8 : Bila terdapat fitness TemplateVariety dalam currentFitness, maka fitness tersebut akan diberikan kromosom original yang hanya berisi id – id template.
* 11 – 12 : Setiap fitnesses yang ada akan di reset variabel internalnya agar memulai dari nilai – nilai default yang ditentukan.
* 14 – 17 : Setiap tile dalam level akan diakses dan disimpan koordinatnya.
* 18 – 19 : Bila tile yang dipilih mengandung sebuah player maka jumlah player akan ditambah.
* 20 – 21 : Setiap fitness yang dimiliki akan melakukan calculateFitness, dimana calculataFitness adalah sebuah fungsi yang bisa berfungsi untuk menyimpan dan mengolah data untuk membantu dalam menghitung nilai fitness.
* 24 – 25 : Setiap fitness yang ada akan dikalkulasikan nilai fitness akhirnya, lalu setelah fitness tersebut dihitung, maka fitness tersebut akan dikembalikan ke array fitnessScores.
* 26 : Seluruh fitness akan dijumlah dan akan didapat nilai fitness total, tetapi sebelum direturn akan dilihat bila karakter berjumlah 10, bila jumlah karakter tidak berjumlah 10 maka fitness akan dibagi sehingga memiliki fitness total yang jauh lebih kecil.

1. Fitness Individual

Sekarang dengan seluruh sistem fitness dijelaskan, bagian terakhir yang belum dijelaskan adalah InLoopFitnessBase yang terdapat di segmen program 6.5 Fungsi FitnessFunction baris 3. InLoopFitnessBase memiliki base FitnessBase, yang dipakai bila ingin membuat sebuah fitness baru, berikut adalah potongan kodenya.

Segmen Program 6.6 Class Abstrak FitnessBase

1. public abstract class FitnessBase : MonoBehaviour{
2. public string FitnessName { get { return fitnessName; } }
3. public bool ForTemplateGen { get { return forTemplateGen; } }
4. public bool ForTileGen { get { return forTileGen; } }
5. public bool IsUsed { get { return isUsed; } }
6. [SerializeField] bool isUsed = true;
7. [SerializeField] bool forTileGen;
8. [SerializeField] bool forTemplateGen;
9. [SerializeField] string fitnessName;
10. [SerializeField] protected float weight = 1;
11. public abstract float getFitnessScore();
12. public abstract void resetVariables();
13. }

Berikut adalah kegunaan tiap variabel (Perlu diketahui bila SerializeField hanya berarti variabel tersebut dapat diedit di engine unity secara langsung, ini berguna agar bisa mudah mengganti isi variabel tanpa membuka script) :

* 2 – 5 : Bekerja sebagai versi public dari variabel 7 – 10
* isUsed : Menentukan apabila Fitness akan dipakai pada GA
* forTileGen :Menentukan bila fitness bisa dipakai saat menggunakan Tile Generation
* forTemplateGen : Menentukan bila fitness bisa dipakai saat menggunakan Template Generation
* fitnessName : Menampilkan nama fitness saat melakukan debug
* weight : Mengkalikan skor fitness dengan weight agar skor fitness dapat diubah besar pengaruhnya terhadap genetic Algorithm
* getFitnessScore : Sebuah fungsi yang memberikan nilai Fitness dari fitness tersebut.
* resetVariables : Sebuah fungsi yang mengulang semua variabel agar siap dipakai untuk perhitungan baru.

Sekarang class ini memiliki 2 anak, yaitu InLoopFitnessBase dan OutLoopFitnessBase, dimana InLoop berguna bila fitness tersebut memerlukan pemgecekan dalam setiap tile, sementara OutLoop hanya bekerja sekali di luar loop dan bisa langsung mengambil fitness. Di awal terdapat kedua Fitness tersebut tapi pada akhirnya hanya dipakai InLoopFitnessBase karena satu satunya fitness yang dapat memakai OutLoop hanyalah template Variety, dimana pada akhirnya diadaptasikan ke OutLoop karena lebih mudah. Sementara untuk isi InLoopBase :

Segmen Program 6.7 Class Abstrak InLoopFitnessBase

1. public abstract class InLoopFitnessBase : FitnessBase{
2. protected float fitnessTotal;
3. public abstract void calculateFitness(int[,] map, Coordinate currCoor);
4. }

Dalam segmen program 6.7 hanya terdapat fitnessTotal, variabel yang menampung akumulasi fitness apabila diperlukan dan calculateFitness, yang memerlukan parameter level dan koordinat loop sekarang ini untuk mengkalkulasi fitness.

Dengan seluruh kelas abstrak dijelaskan, mari kita masuk ke hidangan utama dari seluruh isi subbab 6.2 dan 6.3 yaitu kode dan penjelasan dari masing – masing fitness:

1. Rock Amount Fitness

Fitness ini menghitung jumlah batu dalam sebuah level dan membandingkan jumlah batu tersebut terhadap 2 range angka, dimana bila jumlah batu tersebut berada diluar range tersebut maka fitness akan berkurang. Fitness akan terus berkurang hingga fitness mencapai 0.

Berikut adalah class fitness Rock Amount :

Segmen Program 6.8 Class Rock Amount Fitness

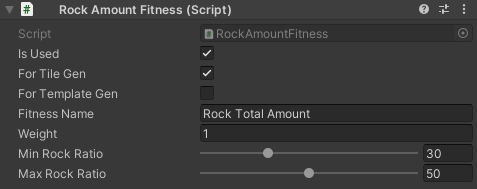
1. public class RockAmountFitness : InLoopFitnessBase{
2. [Range(0.0f, 100.0f)]
3. [SerializeField] float minRockRatio;
4. [Range(0.0f, 100.0f)]
5. [SerializeField] float maxRockRatio;
6. int rockAmount;
7. public override void calculateFitness(int[,] map, Coordinate currCoor){
8. if (map[currCoor.yCoor,currCoor.xCoor] == 1)
9. rockAmount++;
10. }
11. public override float getFitnessScore(){
12. int maxRock = Mathf.RoundToInt(SetObjects.getWidth() \* SetObjects.getHeight() \* maxRockRatio / 100);
13. int minrock = Mathf.RoundToInt(SetObjects.getWidth() \* SetObjects.getHeight() \* minRockRatio / 100);
14. if (maxRock < minrock){
15. Debug.LogWarning("Min dan max ditukar");
16. (minrock, maxRock) = (maxRock, minrock);
17. }
18. float nilaiMinus = 0;
19. if (rockAmount < minrock)
20. nilaiMinus = minrock - rockAmount;
21. else if (rockAmount > maxRock)

Segmen Program 6.8 (Lanjutan)

1. nilaiMinus = rockAmount - maxRock;
2. //ini untuk batas normalisasi
3. float nilaiMinusMax = SetObjects.getWidth() \* SetObjects.getHeight() - maxRock > minrock ? SetObjects.getWidth() \* SetObjects.getHeight() - maxRock : minrock;
4. nilaiMinus /= nilaiMinusMax;
5. float score = 1 - nilaiMinus;
6. return Mathf.Pow(score, 2) \* weight;
7. }
8. public override void resetVariables(){
9. rockAmount = 0;
10. }
11. }

Berikut adalah penjelasan dari kode diatas :

* 2 – 5 : Barisan kode ini memberi kita pengaturan atas seberapa banyak min% dan max% batu yang ada di dalam level melalui variabel minRockRatio dan maxRockRatio.
* 8 – 11 : Bila tile yang diberikan sekarang ini adalah sebuah batu, maka isi rockAmount akan bertambah.
* 14 & 15 : Memasukkan minRockRatio dan maxRockRatio yang berbentuk rasio ke dalam minRock dan Maxrock dalam bentuk bulatan jumlah batu.
* 16 – 19 : Bila maxRock lebih kecil dari minRock, maka isi dari kedua variabel tersebut akan ditukar.
* 21 – 25 : Terdapat variabel bernama nilaiMinus yang diisi dengan perbedaan antara jumlah batu (variabel rockamount) dan minRock atau MaxRock bila rockamount berada diluar range tersebut, sementara bila rockamount ada didalam range, maka nilaiMinus = 0.
* 28 : Variabel nilai nilaiMinusMax akan diisi dengan nilaiMinus maksimal yang bisa dirain, ini dilakukan dengan membandingkan luas level – maxRock dengan minRock – 0, dimana angka yang lebih besar akan dimasukkan ke dalam nilaiMinusMax
* 30 : nilaiMinus dinormalisasikan terhadap nilaiMinusMax agar memiliki nilai 0 – 1
* 31 – 32 : Direturn fitness bernilai 1-nilaiMinus, lalu fitness dikuadrat untuk memperbesar kesalahan yang dibuat, mendorong GA untuk mempercepat penemuan fitness terbaik, lalu setelah itu dikali dengan weight yang ditetapkan dan fitness direturn.
* 35 – 37 : Rockamount direset menjadi 0 untuk mempersiapkan perhitungan fitness.



Gambar 6.2

Isi Variabel dari Rock Amount Fitness

Dari 3 variabel pertama, ditentukan bila fitness ini hanya dipakai untuk Tile Generation, ditentukan Fitness name dan Weight nya, dimana Weight hanya bernilai 1, standar untuk segala fitness. Lalu untuk Min dan Max Rock Ratio, dibuat range nya diantara 30% – 50%, sehingga jumlah batu yang diharapkan dalam sebuah level sebanyak 30 hingga 50 persen dari luas level.

1. Rock Groups Size Fitness

Selanjutnya adalah fitness yang bekerja baik dengan Rock Amount Fitness, dimana Rock amount fitness mengatur berapa banyak batu yang diinginkan, Rock Groups Size Fitness mengatur jumlah batu dalam sebuah kelompok batu. Fitness ini berguna untuk mengurangi batu yang bersifat sendiri ataupun kelompok batu yang terlalu besar menyatu jadi 1.

Secara keseluruhan, fitness ini berguna untuk mengatur ukuran dari kelompok batu yang tersedia, tetapi tidak mengatur bentuk kelompok batu tersebut.

Berikut adalah class Rock Groups Size Fitness :

Segmen Program 6.9 Rock Groups Size Fitness

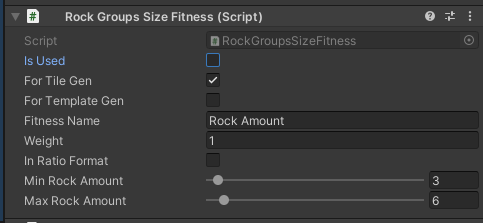
1. public class RockGroupsSizeFitness : InLoopFitnessBase{
2. [SerializeField] bool inRatioFormat;
3. [Range(0.0f, 100.0f)]
4. [SerializeField] float minRockAmount;
5. [Range(0.0f, 100.0f)]
6. [SerializeField] float maxRockAmount;
7. bool[,] ischecked;
8. int rockGroupAmount;
9. //Fitness ini mengambil ukuran dari sebuah area dan dibandingkan dengan rasio ukuran yang diminta
10. public override void calculateFitness(int[,] map, Coordinate currCoor){
11. if (map[currCoor.yCoor, currCoor.xCoor] != 1 || ischecked[currCoor.yCoor, currCoor.xCoor])
12. return;
13. rockGroupAmount++;
14. int size = 1, i = currCoor.yCoor, j = currCoor.xCoor;
15. ischecked[i, j] = true;
16. Queue<Coordinate> q = new();
17. Coordinate c, tempcoor;
18. q.Enqueue(new Coordinate(j, i));
19. //ngambil ukuran area 1 per 1
20. while (q.Count > 0){
21. c = q.Dequeue();
22. for (int k = -1; k < 2; k++){
23. for (int l = -1; l < 2; l++){
24. //Cek 8 Arah
25. tempcoor = new Coordinate(c.xCoor + l, c.yCoor + k);
26. if (tempcoor.xCoor >= 0 && tempcoor.yCoor >= 0 && tempcoor.yCoor < SetObjects.getHeight() && tempcoor.xCoor < SetObjects.getWidth() && map[tempcoor.yCoor, tempcoor.xCoor] == 1 && !ischecked[tempcoor.yCoor, tempcoor.xCoor]){
27. ischecked[tempcoor.yCoor, tempcoor.xCoor] = true;
28. q.Enqueue(tempcoor);
29. size++;
30. }
31. }
32. }
33. }
34. int maxRock, minrock;
35. if (inRatioFormat){

Segmen Program 6.9 (Lanjutan)

1. maxRock = Mathf.RoundToInt(map.GetLength(0) \* map.GetLength(1) \* maxRockAmount / 100);
2. minrock = Mathf.RoundToInt(map.GetLength(0) \* map.GetLength(1) \* minRockAmount / 100);
3. }
4. Else{
5. maxRock = Mathf.FloorToInt(maxRockAmount);
6. minrock = Mathf.FloorToInt(minRockAmount);
7. }
8. if (maxRock < minrock){
9. Debug.LogWarning("Min dan max ditukar");
10. (minrock, maxRock) = (maxRock, minrock);
11. }
12. float nilaiMinus = 0;
13. if (size < minrock)
14. nilaiMinus = minrock - size;
15. else if (size > maxRock)
16. nilaiMinus = size - maxRock;
17. float nilaiMinusMax = SetObjects.getWidth() \* SetObjects.getHeight() - maxRock > minrock ? SetObjects.getWidth() \* SetObjects.getHeight() - maxRock : minrock;
18. nilaiMinus /= nilaiMinusMax;
19. //Fitness Total akan ditambah dengan 1 - beda antara ekspektasi dan jumlah batu per kelompok
20. fitnessTotal += 1 - nilaiMinus;
21. }
22. override public void resetVariables(){
23. ischecked = new bool[SetObjects.getHeight(), SetObjects.getWidth()];
24. rockGroupAmount = 0;
25. fitnessTotal = 0;
26. }
27. override public float getFitnessScore(){
28. if (fitnessTotal > 0){
29. //Kembalikan rata - rata
30. return Mathf.Pow(fitnessTotal / rockGroupAmount, 2) \* weight;
31. }
32. else
33. return 0;
34. }
35. }

Berikut adalah penjelasan kodenya :

* 3 – 6 : Mirip dengan minRockAmount dan maxRockAmount di segmen program 6.8 (karena memang copas), dimana min dan maxRock dipakai untuk jumlah batu maksimal dalam sebuah grup, bukan dalam level
* 2 : Bool ini menentukan bila min dan maxRock merupakan sebuah bulatan atau sebuah persenan.
* 12 : Bila koordinat yang diberi bukanlah sebuah batu atau sudah pernah diperiksa, maka fungsi direturn duluan.
* 15 : Sementara bila koordinat merupakan batu yang belum pernah diperiksa fitness ini, maka fungsi dilanjutkan dan variabel rockGroupAmount yang mengandung jumlah kelompok batu ditambah 1.
* 17 : Variabel isChecked yang adalah sebuah array 2D yang menandakan apakah tile map tertentu telah diperiksa. Dalam baris ini koordinat i(koordinat y) , j(koordinat x) telah diperiksa dan diberi bool true di koordinat tersebut.
* 18 : Mempersiapkan Queue koordinat baru untuk dipakai nanti.
* 20 : Memasukkan koordinat sekarang ini ke Queue
* 22 – 35 : Dimulai dari koordinat pertama tersebut, fungsi akan melihat 8 tile sekitar koordinat pertama dan memasukkan batu – batu yang ditemukan ke dalam Queue, dan memberitahu variabel isChecked bahwa koordinat sekitar telah diperiksa. Terakhir untuk setiap batu yang ditemukan variabel size akan bertambah, dimana size merepresentasikan jumlah batu dalam satu kelompok. Ini akan diulang terus hingga Queue habis, dimana seluruh kelompok batu telah diperiksa.
* 40 – 43 : Bila inRatioFormat dinyatakan true, maka maxRock dan minRock akan diisi luas level dikali RockAmount% yang berhubungan.
* 44 – 47 : Tetapi bila inRatioFormat dinyatakan false maka maxRock dan minRock langsung diisi RockAmount yang berhubungan.
* 48 – 61 : Menggunakan sistem nilaiMinus dan nilaiMinusMax yang sama dengan Segman Program 6.8 diatas.
* 63 : Hasil fitness dari sistem nilaiMinus tidak di return, tetapi disimpan kedalam variabel fitnessTotal yang nanti akan dipakai pada perhitungan akhir.
* 66 – 70 : Mempersiapkan variabel untuk dipakai.
* 72 – 79 : Bila fitnessTotal diatas 0 (alias minimal ada 1 kelompok batu dalam level), maka fitnessTotal akan dibagi dengan jumlah kelompok batu, membuat rata rata fitness, lalu rata – rata tersebut dikuadrat ,dikali weight dan di return untuk dihitung Genetic Algorithm.



Gambar 6.3

Isi Variabel dari Rock Groups Size Fitness

Jadi cerita lucu, entah kenapa fitness ini lupa dinyalakan sebelum disebar ke teman – teman dan karena itu level yang degenerate sedikit berbeda dengan ekspetasi, tetapi kesalahan ini baru disadari saat membuat buku ini. Selain kesalahan konyol diatas, diberi nama fitness dan Weight sesuai standar, lalu In ratio format dimatikan. Karena In ratio format dimatikan, maka range dari jumlah batu dalam sebuah kelompok adalah 3 – 6 batu daripada 3% – 6% luas level .

1. Area Fitness

Selanjutnya adalah Area Fitness, guna dari fitness ini adalah untuk memberitahu Genetic Algorithm untuk meminimalisir area yang tidak dapat diraih, sehingga tidak terdapat kasus dimana setiap karakter dalam sebuah level tidak bisa saling bertemu karena diblok oleh dinding.

Berikut adalah potongan kodenya :

Segmen Program 6.10 Area Fitness

1. public class AreaFitness : InLoopFitnessBase{
2. ArrayList areaSize;
3. int totalSize;
4. bool[,] ischecked;
5. public override void calculateFitness(int[,] map, Coordinate currCoor){
6. int size = 1, i = currCoor.yCoor, j = currCoor.xCoor;
7. if (map[i, j] == 1 || ischecked[i, j])
8. return;
9. ischecked[i, j] = true;
10. Queue<Coordinate> q = new Queue<Coordinate>();
11. Coordinate c, tempCoor;
12. q.Enqueue(new Coordinate(j, i));
13. //Ngambil Ukuran area yang sekarang di koordinat currCoor
14. while (q.Count > 0) {
15. c = q.Dequeue();
16. for (int k = 0; k < 4; k++){
17. tempCoor = new Coordinate(c.xCoor + Mathf.RoundToInt(Mathf.Sin(k \* Mathf.PI / 2)), c.yCoor + Mathf.RoundToInt(Mathf.Cos(k \* Mathf.PI / 2)));
18. if (tempCoor.xCoor >= 0 && tempCoor.yCoor >= 0 && tempCoor.yCoor < SetObjects.getHeight() && tempCoor.xCoor < SetObjects.getWidth() && map[tempCoor.yCoor, tempCoor.xCoor] != 1 && !ischecked[tempCoor.yCoor, tempCoor.xCoor]) {
19. ischecked[tempCoor.yCoor, tempCoor.xCoor] = true;
20. q.Enqueue(tempCoor);
21. size++;
22. }
23. }
24. }
25. areaSize.Add(size);
26. }
27. public override float getFitnessScore(){
28. float biggest = -999;
29. // Ini aku pakai area yang bisa diakses player, bukan panjang \* lebar Arena
30. for (int i = 0; i < areaSize.Count; i++){
31. totalSize += (int)areaSize[i];
32. if ((int)areaSize[i] > biggest)
33. biggest = (int)areaSize[i];
34. }
35. fitnessTotal = biggest / totalSize;
36. return Mathf.Pow(fitnessTotal, 2) \* weight;
37. }

Segmen Program 6.10 (Lanjutan)

1. public override void resetVariables(){
2. areaSize = new ArrayList();
3. totalSize = 0;
4. ischecked = new bool[SetObjects.getHeight(), SetObjects.getWidth()];
5. }
6. }

Berikut adalah penjelasan kodenya :

* 3 – 4 : Arraylist areasize digunakan untuke menyimpan ukuran dari seluruh area – area dalam level yang ada, semisal terdapat 2 area yang terpisah, maka ukuran kedua area tersebut akan dimasukkan ke areasize. Selain itu, totalSize adalah total area dari semua areasize.
* 9 – 10 : Bila tile koordinat yang diberi merupakan sebuah batu atau sudah dicek (tidak menggunakan variabel ischecked milik segmen 6.9 diatas), maka direturn lebih awal.
* 12 -27 : Dilakukan pengambilan luas area seperti pengecekan kelompok batu di segmen 6.9 diatas, cuma pengambilan luas di sini hanya 4 tile sekitar ketimbang 8, dan tile yang diambil adalah tile apapun selain batu, jadi tile karakter dan tile bola spesial ikut dimasukkan dalam Queue.
* 28 : Ukuran dari area yang barusan diambil dimasukkan kedalam arraylist areaSize.
* 34 - 38 : Di bagian program ini terjadi 2 hal, yang pertama adalah pentotalan luas area yang didapat, lalu dimasukkan ke dalam totalSize, kedua adalah pengecekan luas area yang terbesar dalam areaSize, dan luas area terbesar tersebut dimasukkan ke variabel biggest.
* 41 – 42 : fitnessTotal merupakan area yang terbesar dibagi dengan area total, sehingga bila ada area yang tidak bisa diakses, maka fitnessTotal akan berkurang dari nilai 1 tergantung dari luas area yang tidak bisa diakses. Setelah mendapat fitnessTotal, fitnessTotal dikuadrat, dikali weight lalu di return.
* 45 – 49 : Persiapan variabel untuk perhitungan.

1. Power Up Access Fitness

Fitness ini gunanya sederhana, dimana ada sebuah spawner bola spesial, maka bola spesial tersebut harus bisa diraih setidaknya satu karakter. Prinsipnya mirip dengan Fitness Area dimana semakin banyak area / bola spesial yang tidak bisa diraih, maka semakin kecil fitness yang didapat. Berikut adalah class-nya :

Segmen Program 6.11 Power Up Access Fitness

1. public class PowerUpAccessFitness : InLoopFitnessBase{
3. int[,] map;
4. ArrayList lokasiPlayer;
5. ArrayList lokasiPowerUp;
6. public override void calculateFitness(int[,] map, Coordinate currCoor){
7. this.map = map;
8. int i = currCoor.yCoor, j = currCoor.xCoor;
9. if (map[i, j] == 2)
10. lokasiPowerUp.Add(new Coordinate(j, i));
11. if (map[i, j] == 3)
12. lokasiPlayer.Add(new Coordinate(j, i));
13. }
14. public override float getFitnessScore(){
15. int indexPlayer = 0;
16. float tempDistance, biggest;
17. if (lokasiPlayer.Count <= 0)
18. return 0;
19. for (int i = 0; i < lokasiPowerUp.Count; i++){
20. biggest = 999;
21. //Ambil player terdekat biar Astar tidak terlalu lama
22. for (int j = 0; j < lokasiPlayer.Count; j++){
23. tempDistance = Coordinate.Distance((Coordinate)lokasiPowerUp[i], (Coordinate)lokasiPlayer[j]);
24. if (tempDistance < biggest){
25. biggest = tempDistance;
26. indexPlayer = j;
27. }
28. }
29. if (AStarAlgorithm.doAstarAlgo((Coordinate)lokasiPowerUp[i], (Coordinate)lokasiPlayer[indexPlayer], map) != null)
30. fitnessTotal++;
31. }
32. if (lokasiPowerUp.Count > 0)
33. return Mathf.Pow(fitnessTotal / lokasiPowerUp.Count, 2) \* weight;
34. else
35. return 0;

Segmen Program 6.11 (Lanjutan)

1. }
2. public override void resetVariables(){
3. fitnessTotal = 0;
4. lokasiPlayer = new ArrayList();
5. lokasiPowerUp = new ArrayList();
6. }
7. }

Penjelasan dari class Powerup Access :

* 4 – 5 : Kedua arraylist tersebut dipakai untuk menyimpan secara berurutan koordinat seluruh player dan koordinat seluruh spawner bola spesial.
* 7 – 14 : Memberi salinan map kepada fitness, lalu mengecek koordinat map yang diberi, bila koordinat yang diberi merupakan sebuah tile karakter atau tile bola spesial, maka koordinat tersebut akan masuk ke arraylist yang sesuai.
* 19 – 20 : Bila tidak ketemu player, maka fitness dikembalikan 0, ini hanya untuk Template Generation dimana karakter di spawn setelah degenerate levelnya.
* 22 – 34 : Dilakukan pengecekan untuk mendapat jumlah bola spesial yang bisa diakses
* 23 – 31 : Diambil index dari akun yang terdekat dari bola spesial yang dipilih untuk mengurangi lama melakukan algoritma A\*
* 32 – 33 : Dilakukan algoritma A\* dari posisi player terdekat ke bola spesial yang dipilih sekarang ini, dan bila berhasil maka variabel fitnesstotal akan ditambah.
* 35 – 36 : Bila terdapat bola spesial, maka di return fitness berupa bola spesial yang bisa diakses dibagi dengan bola spesial dalam level, dikuadrat lalu dikali weight.
* 37 – 38 : Tetapi bila tidak ada power up, maka fitness akan mengembalikan nilai 0
* 41 – 45 : Mempersiapkan variabel untuk fitness, itu aja sih.

1. Power Up Ratio Fitness

Sekarang adalah PowerUpRatio Fitness, prinsipnya mirip Rock Amount Fitness, hanya daripada menghitung batu, kita menghitung jumlah bola spesial. Selain itu fitness ini juga menggunakan fitur pemilihan bentuk angka persenan atau bulat seperti Rock Groups Size Fitness.

Potongan kode :

Segmen Program 6.12 Power Up Ratio Fitness

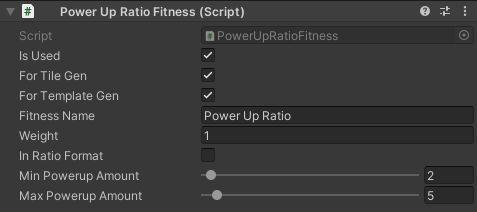
1. public class PowerUpRatioFitness : InLoopFitnessBase{
2. int powerUpAmount;
3. [SerializeField] bool inRatioFormat;
4. [Range(0.0f, 100.0f)]
5. [SerializeField] float minPowerupAmount;
6. [Range(0.0f, 100.0f)]
7. [SerializeField] float maxPowerupAmount;
8. public override void calculateFitness(int[,] map, Coordinate currCoor){
9. int i = currCoor.yCoor, j = currCoor.xCoor;
10. if (map[i, j] == 2)
11. powerUpAmount++;
12. }
13. public override float getFitnessScore(){
14. int minRatio, maxRatio;
15. if (inRatioFormat){
16. minRatio = Mathf.RoundToInt(SetObjects.getHeight() \* SetObjects.getWidth() \* minPowerupAmount / 100);
17. maxRatio = Mathf.RoundToInt(SetObjects.getHeight() \* SetObjects.getWidth() \* maxPowerupAmount / 100);
18. }
19. else{
20. minRatio = Mathf.FloorToInt(minPowerupAmount);
21. maxRatio = Mathf.FloorToInt(maxPowerupAmount);
22. }
23. if (maxRatio < minRatio){
24. Debug.LogWarning("Min dan max ditukar");
25. (minRatio, maxRatio) = (minRatio, maxRatio);
26. }
27. float nilaiMinus = 0;
28. if (powerUpAmount < minRatio)
29. nilaiMinus = minRatio - powerUpAmount;
30. else if (powerUpAmount > maxRatio)
31. nilaiMinus = powerUpAmount - maxRatio;
32. float nilaiMinusMax = SetObjects.getWidth() \* SetObjects.getHeight() - maxRatio > minRatio ? SetObjects.getWidth() \* SetObjects.getHeight() - maxRatio : minRatio;

Segmen Program 6.12 (Lanjutan)

1. nilaiMinus = nilaiMinus / nilaiMinusMax;
2. return MathF.Pow(1 - nilaiMinus, 2) \* weight;
3. }
4. public override void resetVariables(){
5. fitnessTotal = 0;
6. powerUpAmount = 0;
7. }
8. public float getRatio(){
9. return minPowerupAmount / 100;
10. }
11. }

Berikut adalah penjelasan kodenya :

* 3 – 8 : Seluruh variabel memiliki fungsi yang sama dengan variabel – variabel di Rock Group Size Fitness, kecuali min dan max PowerUpAmount yang bertindak sebagai min & max Rock Amount dari Rock Group Size Fitness
* 12 – 13 : Bila tile koordinat yang diberi berisi bola spesial, maka isi variabel powerupamount bertambah
* 16 – 41 : Fungsi ini persis baris 37 – 63 dari Rock Group Size Fitness, dengan satu satunya perbedaan hasil fitnessnya direturn.
* 43 – 45 : Mempersiapkan variabel.
* 48 – 50 : Fungsi ini akan dipakai oleh Genetic Algorithm, secara spesifik fungsi ini akan dipakai untuk menentukan berapa bola spesial yang muncul pada populasi pertama kromosom tile generation.



Gambar 6.4

Isi Variabel dari Power Up Ratio Fitness

Dari 3 variabel pertama, fitness ini dipakai apabila pada tile dan template generation, selanjutnya diberi nama dan Weight seperti biasa, terdapat In Ratio Format yang dimatikan untuk memudahkan perhitungan. Lalu terakhir range dari jumlah power up dalam sebuah level adalah diantara 2 hingga 5 spawner.

1. Template Variety Fitness

Dan fitness terakhir kita, Template Variety Finess yang hanya dipakai di Template Generation digunakan agar meminimalis template yang kembar, dan alhasil terdapat lebih banyak variasi yang bisa dibuat oleh Genetic Algorithm ketimbangg beberapa template yang sama di beberapa tempat.

Berikut adalah kodenya :

Segmen Program 6.13 Template Variety Fitness

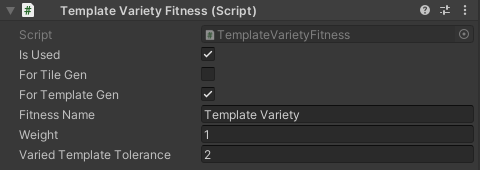
1. public class TemplateVarietyFitness : InLoopFitnessBase{
2. [SerializeField] int variedTemplateTolerance;
3. int[] savedMap;
4. public override void calculateFitness(int[,] map, Coordinate currCoor){
5. return;
6. }
7. //Khusus Template Variety karena dia pake template map bukan map biasanya
8. public void getTemplateMap(Gene[] map{
9. savedMap = new int[map.Length];
10. for (int i = 0; i < map.Length; i++)
11. savedMap[i] = (int)map[i].Value;
12. }
13. public override float getFitnessScore(){
14. SortedList<int, int> templateOccurences = new();
15. for (int i = 0; i < savedMap.Length; i++){
16. if (templateOccurences.ContainsKey(savedMap[i]))
17. templateOccurences[savedMap[i]]++;
18. else
19. templateOccurences.Add(savedMap[i], 1);
20. }
21. int[] scores = new int[templateOccurences.Count];
22. int temp;
23. for (int i = 0; i < templateOccurences.Count; i++){
24. temp = templateOccurences.Values[i] - variedTemplateTolerance < 0 ? -(templateOccurences.Values[i] - variedTemplateTolerance) : 0;

Segmen Program 6.13 (Lanjutan)

1. scores[i] = 1 + temp;
2. }
3. return Mathf.Pow(scores.Sum() / scores.Length,2) \* weight;
4. }
5. public override void resetVariables(){
6. return;
7. }
8. }

Dan berikut adalah penjelasannya :

* 3 : Variabel ini menandakan toleransi fitness terhadap template yang kembar. Misalkan variedTemplateTolerance diisi 2, maka bila terdapat sebuah template yang muncul 2 kali maka fitness akan tetap 1, tetapi 3 keatas maka fitness akan berkurang.
* 4 : Variabel yang menyimpan kromosom template, bukan map yang jadi.
* 6 – 8 : Function yang tidak digunakan, karena fitness ini hanya perlu kromosom template.
* 10 – 15 : Mengisi variabel savedMap dengan kromosom map yang diberikan.
* 18 : Membuat sortedList templateOccurences dengan sistem key dan value, dimana key adalah id template dan value adalah jumlah kemunculan template tersebut.
* 19 – 24 : Menggunakan sortedList dan sistem key value, seluruh id template dimasukkan ke templateOccurences dengan key template id yang sedang diperiksa. Bila key tersebut tidak ada maka dibuat key tersebut dan value diisi 1, sementara bila ada value +1.
* 25 - 30 : Setiap jumlah kemunculan dicek, dan bila jumlah kemunculan dibawah atau sama dengan toleransi fitness, maka akan dimasukkan skor 1 ke dalam array skor, tapi bila lebih banyak dari toleransi fitness, maka akan diberi skor 0 kebawah.
* 32 : Nilai skor di rata – rata sesuai jumlah macam template yang muncul, dikuadrat lalu dikali weight.
* 35 – 37 : Tidak ada variabel yang perlu dipersiapkan, sehingga diberi return saja.



Gambar 6.5

Isi Variabel dari Template Variety Fitness

Dan berikut adalah variabel – variabel yang digunakan. Selain seluruh variabel standar, terdapat 2 perbedaan. Pertama adalah fitness ini hanya dipakai untuk template generation, dikarenakan ini memang hanya bisa dipakai di template, lalu Varied Template Tolerance, yang menandakan bila fitness bisa mentoleransi 2 kemunculan template yang kembar sebelum mengurangi nilai fitness dari Genetic Algorithm.