BAB VI

IMPLEMENTASI MAP GENERATION MENGGUNAKAN GENETIC ALGORITHM

Menggunakan GeneticSharp oleh Giacomelli, bab ini akan menjelaskan dan menyelam menuju aspek Map Generation dari game Splatted. Dalam bab ini, akan dijelaskan seluruh implementasi dari unsur-unsur Map Generation dari game Splatted. Penjelasan akan dumulai dari Chromosome dari algoritma genetik :

1. Chromosome

Dimulai dengan dasar dari sebuah Genetic Algorithm, subbab ini akan menjelaskan mengenai implementasi Chromosome dari permasalahan Procedural Map Generation bagi Splatted. Chromosome yang digunakan memerlukan sebuah Class tersendiri dan juga menuruni Class ChromosomeBase agar bisa dipakai dalam algoritma GeneticSharp. Kedua metode generasi yang dijelaskan pada bab sebelumnya akan didalami pada subbab ini, dimulai dari Tile Generation Chromosome.

1. Tile Chromosome

Tile Chromosome adalah sebuah class yang digunakan sebagai Chromosome dari Tile Generation. Cara kerja Tile Chromosome adalah setiap Gene dalam sebuah Tile Chromosome merepresentasikan isi dari sebuah tile di dalam Map yang akan dibuat. Berikut adalah potongan kode yang digunakan untuk membuat Chromosome 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){

Segmen Program 6.1 (Lanjutan)

1. int temp;
2. m\_ukuranMap = ukuranMap;
3. m\_powerup = powerup;
4. var mapValues = RandomizationProvider.Current.GetInts (ukuranMap, 0, 2);
5. for (int i = 0; i < ukuranMap; i++)
6. ReplaceGene(i, new Gene(mapValues[i]));
7. for (int i = 0; i < 5; i++){
8. temp = Mathf.FloorToInt(Random.Range(0, ukuranMap));
9. while (mapValues[temp] == 3)
10. temp = Mathf.FloorToInt(Random.Range(0, ukuranMap));
11. ReplaceGene(temp, new Gene(3));
12. }
13. for (int i = 0; i < powerup; i++){
14. temp = Mathf.FloorToInt(Random.Range(0, ukuranMap));
15. while (mapValues[temp] == 3 || mapValues[temp] == 2)
16. temp = Mathf.FloorToInt(Random.Range(0, ukuranMap));
17. ReplaceGene(temp, new Gene(2));
18. }
19. }
20. public override Gene GenerateGene(int geneIndex){
21. return new Gene(RandomizationProvider.Current.GetInt(0, 3));
22. }
23. public override IChromosome CreateNew(){
24. return new GameChromosome(m\_ukuranMap,m\_powerup);
25. }
26. public override IChromosome Clone(){
27. var clone = base.Clone() as GameChromosome;
28. return clone;
29. }
30. }

Penjelasan untuk masing masing baris kode :

* 1 : TileChromosome menggunakan class ChromosomeBase yang telah disediakan oleh GeneticSharp untuk membuat kromosom yang bisa dipakai untuk algortima genetik.
* 3–4&8-9 : Kode ini terdapat di dalam tutorial milik Giacomelli sehingga lebih baik dimasuki saja meski mungkin tidak memiliki pengaruh.
* 6 : Constructor untuk membuat Chromosome baru dengan 2 parameter, yaitu ukuran map yang berisi luas Map yang akan dibuat dan powerup yang berisi jumlah powerup yang diinginkan di dalam Map ini.
* 11 : Variabel mapValues akan diisi dengan sebuah array integer yang berukuran luas level dan akan diisi oleh angka dengan range 0 hingga 2.
* 12-13 : Fungsi ini digunakan untuk mengisi Chromosome dari class ini (yang berada di dalam class ChromosomeBase) dengan MapValues.
* 14-19 : Mengubah 5 Gene dalam Chromosome menjadi integer 3. 3 adalah representasi tile yang memiliki seorang karakter. Aksi ini dilakukan untuk mempermudah proses Generasi dengan menyediakan karakter kita di awal Generasi sehingga Crossover dan Mutation hanya perlu mengubah posisi dari karakter yang diletakkan.
* 20-25 : Sama seperti proses diatas. Tetapi ketimbang mengganti Gene menjadi 3, Gene yang terpilih akan diganti isinya menjadi 2 yang merepresentasikan bola spesial di level.
* 28-30 : Melakukan Override terhadap function yang dipakai untuk Membuat Gene baru, dimana sepertinya fungsi ini dipakai dalam beberapa Mutation yang disediakan oleh Geneticsharp.
* 32-33 : Bisa jadi sebuah fungsi yang dipakai untuk membuat kromosom baru. Sulit melihat kegunaan secara spesifik dari fungsi ini karena terdapat Clone beberapa baris dibawah yang diduga memiliki fungsi yang sama.
* 36-38 : Fungsi ini dipakai untuk menduplikat kromosom, dimana fungsi ini tampaknya krusial dalam fase Selection untuk menduplikat kromosom ke Generasi baru.

1. Templated Map Chromosome

Sekarang setelah melihat kedalam kode milik Tile Chromosome, akan ditunjukkan kode yang digunakan untuk membuat Template Map Chromosome. Sebelum itu, kromosom ini memerlukan sebuah class lain yang dipakai untuk mengganti isi dalam setiap Gene menjadi template 5X5. Class tersebut adalah PossibleTemplates.

Kode untuk possible templates diletakkan di lampiran A.1. dimana Class PossibleTemplates memiliki 3 jenis template yang dimasukkan ke array masing-masing, yaitu Oneway, Twoway dan Fourway template. Masing-masing array template tersebut merepresentasikan berapa variasi template unik yang bisa dibuat bila template tersebut dirotasi, seperti contoh template milik Oneway bila dirotasi sebangan 90˚, maka template tersebut akan terlihat sama, dimana template Twoway dan Fourway bila dirotasi akan terlihat berbeda. Seluruh template yang tersedia dalam Splatted dapat dilihat pada lampiran C1 hingga C3.

Menggunakan ketiga array template diatas, dibuat 2 fungsi. Fungsi pertama adalah get Template Amount yang memberikan jumlah template unik yang bisa dibuat. Fungsi kedua adalah get template yang mengambil template berdasarkan id yang diberi..

Berikut ini adalah penjelasan terperinci mengenai potongan kode A.1:

* 3-5 : Variabel one,two dan four way template akan disiapkan terlebih dahulu dengan setiap template dibedakan berdasarkan jumlah Template unik yang dapat didapat dengan memutar sebuah template. Alternatif lain dari menyimpan Template dalam Script adalah dengan 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 Template unik yang bisa dibuat dari seluruh template yang tersedia,. Jumlah hasil yang dikembalikan adalah total dari isi Template four way dikali 4, two way dikali 2 dan one way, ketiga angka tersebut dijumlah dan di return sebagai jumlah template unik 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.
* 19-51 : Di baris ini sebuah template akan diambil berdasarkan tempID yang telah diberikan. Akan ada 2 variabel yang perlu yang diisi, yaitu rotation yang mengatakan berapa banyak rotasi 90˚ yang akan dilakukan terhadap Template. Variabel kedua dan terakhir adalah chosenTemplate yang berisi template yang akan dirotasi.

Tahap pertama dari metode ini adalah untuk melihat apabila tempID merupakan sebuah Template one way, two way atau four way. Setelah menemukan Template asli dari tempID. Setelah mendapat tempID**,** ambil template yang dimiliki oleh tempID dam masukkan Template tersebut kedalam variabel resultTemplate.

Tahap terakhir adalah untuk menentukan rotasi dari template. Variabel rotation akan diisi dengan tempID di MOD dengan jenis Template yang dimiliki pada resultTemplate. Setelah itu resultTemplate akan diputar sebanyak 90˚ sebanyak angka yang disimpan dalam rotasi. Dan dengan itu template siap digunakan oleh algoritma.

* 53-56 : Sebelum mengembalikan template ke algoritma, bila parameter ID bernilai negatif dan tile di paling tengah bernilai 0 (tidak ada apa apa), maka tile di tengah diberi bola spesial lalu template hasil jadi akan diberikan kepada algoritma.

Dengan Class PossibleTemplates dijelaskan, maka kita bisa memasuki class TemplatedMapChromosome. TemplatedMapChromosome adalah sebuah Chromosome berbasis Template dimana setiap isi Chromosome direpresentasikan oleh sebuah ID Template, dimana setiap ID Template akan diberikan kepada Class possibleTemplates diatas dan mengembalikan Tile 5X5 yang dapat diletakkan dalam sebuah Map. Berikut adalah isi dari Class TemplatedMapChromosome:

Segmen Program 6.2 Templated Map Chromosomes

1. public class TemplatedMapChromosome : ChromosomeBase{
2. private readonly int m\_ukuranMap;

Segmen Program 6.2 (Lanjutan)

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

Dibandingkan Segmen Program 6.1 diatas kode ini kurang lebih mirip dengan Chromosome milik Tile Generation, degan perbedaan utama diantara kedua Chromosome terdapat di line 7. MapValues daripada diisi 0 hingga 3 akan diisi dengan range dari -JumlahTemplate hingga JumlahTemplate-1. Selain itu, satu-satunya perbedaan yang lain adalah ukuran map yang didapat oleh Kromosom ini berukuran JumlahArea / 25, dimana 25 merupakan luas area 5x5 dari sebuah Template.

1. Main Code

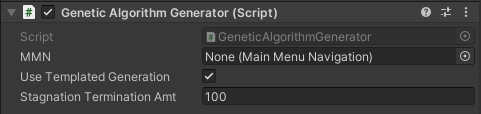
Setelah Chromosome Template dan Tile Generation, sekarang adalah kode yang digunakan untuk menjalankan Genetic Algorithm tersebut. Kode yang dipakai untuk menjalankan Genetic Algorithm tersebut memakai sebuah fungsi bernama AlgorithmStart. Fungsi AlgorithmStart dijalankan pada saat awal layar Loading setelah player memilih pengaturan dari sebuah Level.

Segmen program dari Fungsi AlgorithmStart yang dipindah ke A.2 memiliki beberapa tahap yang harus dilakukan dulu sebelum bisa menjalankan algoritma genetik ini. Tahap pertama adalah untuk menggunakan Class Chromosome berdasarkan metode generasi yang dipilih oleh player. Tahap selanjutnya adalah untuk mengambil Fitness-Fitness yang akan dipakai dalam algoritma genetik, dimana Fitness-Fitness tersebut akan dipakai untuk menghitung kualitas dari setiap Chromosome yang ada.

Setelah itu dibuat aspek-aspek lain dari algoritma genetik seperti metode seleksi, crossover dan syarat berhenti. Tahap terakhir dalam menjalankan algoritma genetik ini adalah dengan membuat merubah Chromosome yang dihitung Fitnessnya menjadi sebuah map yang jadi, agar Fitness-Fitness yang digunakan bisa melihat seberapa bagus Map tersbut. Ulangi seluruh tahap ini untuk setiap Chromosome yang tersedia dan algoritma genetik akan berjalan dengan lancar.

Berikut penjelasan lebih terperinci dari segmen program Genetic Algorithm di A.2:

* 1 : Dalam game Splatted sebuah Chromosome hanya berisi setengah dari sebuah Map yang akan dicerminkan untuk menghindari masalah dengan game balance. Dengan itu useMirrorFitness bila ditentukan sebagai true akan menghitung Fitness berdasarkan dari Map yang telah dicerminkan, bukan Map yang hanya memperlihatkan satu sisi saja.
* 2 : Referensi Script MainMenuNavigation agar setelah Genetic Algorithm selesai Unity bisa mengganti Scene menjadi Scene level utama.
* 3 : Variabel yang menentukan apabila Genetic Algorithm menggunakan Template atau Tile Generation.
* 4 : Menentukan berapa banyak generasi dimana Fitness tidak berkembang berlalu sebelum Genetic Algorithm menyerah dalam menyari Chromosome terbaik.
* 5 : Variabel yang akan menampung semua fitness
* 8 : Seluruh Fitness yang tersedia dimasukkan ke dalam variabel fitnesses
* 9 : Mengambil data apabila player memilih Template Generation atau Tile Generation
* 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 sebuah Chromosome hanya berisi setengah dari sebuah Map.
* 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 map generation, dan membuat Chromosome sesuai dengan permintaan player tersebut.
* 35 : Menentukan jumlah populasi dari Chromosome yang ditentukan. Dalam game ini ditentukan populasi minimum yang bisa dimiliki sebuah generasi adalah 50 sementara populasi maksimumnya adalah 100.
* 37-47 : Menentukan fungsi Fitness yang akan dipakai pada Genetic Algorithm.
* 38-44 : Variabel fc akan diisi dengan seluruh Gene dalam Chromosome, lalu Gene tersebut akan diubah menjadi sebuah array level yang berbentuk 2D.
* 45-46 : Variabel fitness akan diisi skor fitness menggunakan function fitnessFunction yang akan dijelaskan pada subbab berikutnya, lalu fitness itu akan direturn untuk melakukan tahap seleksi.
* 49 : GA diberitahu untuk memakai Roulette Wheel Selection.
* 52 : GA diberitahu untuk memakai Uniform Crossover.
* 53 : Sama seperti diatas, GA diberitahu untuk memakai Partial Shuffle Mutation.
* 54 : GA diberitahu untuk memakai Fitness Stagnation Termination dengan toleransi sebanyak isi variabel StagnationTerminationAmt di baris 4.
* 56-57 : Menggabung semua unsur Genetic Algorithm (Populasi yang dibuat, Crossover dst) kedalam Genetic Algorithm yang siap dipakai.
* 59 : Genetic Algorithm mulai dijalankan.
* 61 : Variabel a akan diisi Chromosome terbaik yang dihasilkan oleh algoritma genetik.
* 62-65 : Menggunakan script SetObjects, ubah kromosom yang didapat pada line 61 menjadi sebuah Map siap dipakai, sekali lagi menggunakan Function berbeda sesuai dengan pilihan Map Generation milik Player.
* 66 : Setelah semuanya selesai, maka akan di load Scene level utama agar player bisa memainkan level yang dibuat.



Gambar 6.1

Isi Variabel pengaturan dari algoritma genetik

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 isi dari variabel ini adalah null dikarenakan setiap Scene akan menggunakan Gameobject yang berbeda. Lalu Use Templated Generation yang digunakan untuk mengganti tipe Map Generation bersifat true, tetapi dapat diubah oleh user melalui pilihan yang tersedia. Terakhir adalah Stagnation Termination Amt yang menandakan bila GA akan berhenti setelah 100 generasi tanpa perkembangan yang signifikan pada Fitness maksimal.

1. Function FitnessFunction

Sekarang sebelum menjelaskan masing-masing fitness yang tersedia, Terdapat fungsi bernama FitnessFunction, yang berfungsi untuk menjalankan seluruh Fitness yang dipilih. Fitness Function memeriksa setiap Tile dalam Map untuk menghitung nilai dari setiap Fitness. Setelah itu, pada akhir perhitungan akan diambil nilai Fitness yang telah dihitung dan seluruh nilai tersebut akan dijumlahkan bersama sebagai nilai Fitness dari Chromosome yang telah diperiksa. Berikut adalah isi dari fungsi fitnessFunction :

Segmen Program 6.3 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 : Fungsi FitnessFunction ini memerlukan 2 parameter, yaitu Map jadi level dalam bentuk array 2 dimensi bersama dengan bentuk Chromosomenya.
* 3 : Dari variabel Global Fitnesses yang sebelumnya telah diisi di baris 10-15 kode utama Genetic Algorithm(A.2), diduplikat Fitness-Fitness tersebut ke dalam variabel currentFitnesses agar setiap kromosom memiliki perhitungan Fitness yang tidak saling mengganggu perhitungan dari Fitness-Fitness Chromosome lainnya..
* 6-8 : Bila terdapat fitness TemplateVariety dalam currentFitness, maka fitness tersebut akan diberikan Chromosome yang hanya berisi ID-ID template.
* 11-12 : Setiap Fitness yang ada akan di reset variabel internalnya agar dapat mulai menghitung dengan nilai-nilai bawaan yang ditentukan.
* 14-17 : Setiap tile dalam level akan diakses dan disimpan koordinatnya.
* 18-19 : Bila tile yang dipilih mengandung sebuah player maka isi dari variabel playerAmount akan ditambah.
* 20-21 : Setiap Fitness yang dimiliki akan melakukan calculateFitness pada Tile yang sekarang ini diperiksa, dimana calculataFitness adalah sebuah fungsi yang digunakan untuk menyimpan dan mengolah data untuk membantu dalam menghitung nilai fitness.
* 24-25 : Setiap fitness yang ada akan dikalkulasikan nilai Fitness akhirnya, dan setelah Fitness tersebut dihitung, maka setiap nilai Fitness tersebut akan dimasukkan kedalam Array fitnessScores.
* 26 : Seluruh Fitness akan dijumlah dan akan didapat nilai Fitness total, tetapi sebelum direturn akan dilihat bila jumlah karakter dalam map berjumlah 10, bila jumlah karakter tidak berjumlah 10 maka Fitness akan dibagi sehingga memiliki Fitness total yang jauh lebih kecil dibandingkan dengan Fitness lain yang memiliki 10 karakter dalam Map.

1. Fitness Individual

Sekarang dengan seluruh cara menghitung Fitness keseluruhan dijelaskan, bagian selanjutnya yang perlu dijelaskan adalah InLoopFitnessBase yang terdapat di segmen program 6.3 Fungsi FitnessFunction baris 3. InLoopFitnessBase memiliki base FitnessBase, yang memiliki fungsi–fungsi yang memudahkan perhitungan Fitness dan dipakai bila ingin membuat sebuah fitness baru. Berikut adalah potongan kode milik Class FitnessBase.

Segmen Program 6.4 Class Abstrak FitnessBase

1. public abstract class FitnessBase : MonoBehaviour{
2. public string FitnessName { get { return fitnessName; } }

Segmen Program 6.1 (Lanjutan)

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

Berikut adalah kegunaan tiap variabel (Perlu diketahui bila SerializeField hanya berarti variabel tersebut dapat diedit di engine Unity secara langsung, ini berguna agar Developer bisa 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 nilai Fitness dapat diubah ukuran pengaruhnya terhadap algoritma genetik, Weight bernilai 1 secara default.
* 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. 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 6.5 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 6.5 hanya terdapat 2 unsur tambahan. Pertama adalah fitnessTotal, sebuah variabel yang menampung akumulasi fitness apabila diperlukan. Kedua adalah fungsi 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 dimulai dengan:

1. Rock Amount Fitness

Fitness ini menghitung jumlah batu dalam sebuah Map dan membandingkan jumlah batu tersebut terhadap 2 range angka, dimana bila jumlah batu tersebut berada diluar range tersebut maka fitness akan berkurang, tetapi bila jumlah batu berada didalam range yang ditentukan, maka fitness akan bernilai 1. Semakin besar selisih antara range dan jumlah batu, semakin kecil nilai Fitness dengan batas minimum Fitness 0. Berikut adalah isi dari class fitness Rock Amount :

Segmen Program 6.6 Class Rock Amount Fitness

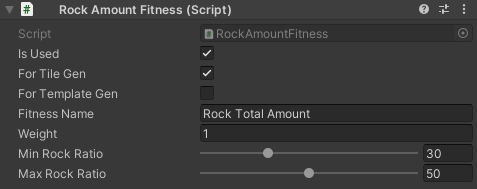
1. public class RockAmountFitness : InLoopFitnessBase{
2. [Range(0.0f, 100.0f)]
3. [SerializeField] float minRockRatio;
4. [Range(0.0f, 100.0f)]
5. [SerializeField] float maxRockRatio;
6. int rockAmount;

Segmen Program 6.6 (Lanjutan)

1. public override void calculateFitness(int[,] map, Coordinate currCoor){
2. if (map[currCoor.yCoor,currCoor.xCoor] == 1)
3. rockAmount++;
4. }
5. public override float getFitnessScore(){
6. int maxRock = Mathf.RoundToInt(SetObjects.getWidth() \* SetObjects.getHeight() \* maxRockRatio / 100);
7. int minrock = Mathf.RoundToInt(SetObjects.getWidth() \* SetObjects.getHeight() \* minRockRatio / 100);
8. if (maxRock < minrock){
9. Debug.LogWarning("Min dan max ditukar");
10. (minrock, maxRock) = (maxRock, minrock);
11. }
12. float nilaiMinus = 0;
13. if (rockAmount < minrock)
14. nilaiMinus = minrock - rockAmount;
15. else if (rockAmount > maxRock)
16. nilaiMinus = rockAmount - maxRock;
17. //ini untuk batas normalisasi
18. float nilaiMinusMax = SetObjects.getWidth() \* SetObjects.getHeight() - maxRock > minrock ? SetObjects.getWidth() \* SetObjects.getHeight() - maxRock : minrock;
19. nilaiMinus /= nilaiMinusMax;
20. float score = 1 - nilaiMinus;
21. return Mathf.Pow(score, 2) \* weight;
22. }
23. public override void resetVariables(){
24. rockAmount = 0;
25. }
26. }

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 Map melalui variabel minRockRatio dan maxRockRatio.
* 8-11 : Bila Tile yang diberikan sekarang ini adalah sebuah batu, maka isi rockAmount akan bertambah.
* 14 & 15 : Mengubah nilai minRockRatio dan maxRockRatio yang berbentuk rasio menjadi minRock dan Maxrock berbentuk 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 selisih antara jumlah batu (variabel rockamount) dan range minRock-MaxRock bila rockamount berada diluar range tersebut. Tetapi bila nilai rockamount berada di dalam range, maka nilaiMinus = 0.
* 28 : Variabel nilai nilaiMinusMax akan diisi dengan nilaiMinus maksimal yang bisa diraih, ini dilakukan dengan membandingkan nilai luas Map - maxRock dengan nilai minRock, dimana angka yang lebih besar akan dimasukkan ke dalam nilaiMinusMax
* 30 : nilaiMinus dinormalisasikan terhadap nilaiMinusMax agar memiliki jangkauan 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 nilai Fitness diberikan kepada fungsi algoritma genetik.
* 35-37 : Rockamount direset menjadi 0 untuk mempersiapkan perhitungan Fitness Chromosome selanjutnya.



Gambar 6.2

Isi Variabel dari Rock Amount Fitness

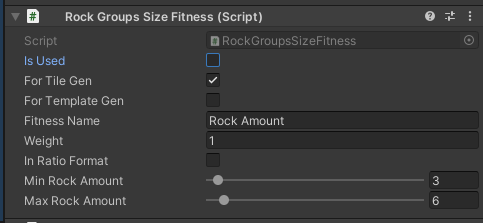
Dari 3 variabel pertama, ditentukan bila fitness ini hanya dipakai untuk Tile Generation. Lalu setelah itu ditentukan Fitness name , standar untuk segala fitness. Terakhir adalah variabel Min dan Max Rock Ratio, ditentukan range jumlah batunya diantara 30%-50%, sehingga jumlah batu yang diharapkan dalam sebuah level sebanyak 30 hingga 50 persen dari luas Map.

1. Rock Groups Size Fitness

Selanjutnya adalah Fitness yang bekerja baik dengan Rock Amount Fitness yaitu Rock Group Size Fitness. Rock amount fitness mengatur berapa banyak batu yang diinginkan, sementara Rock Groups Size Fitness mengatur jumlah batu dalam sebuah kelompok batu, dan dengan itum, jumlah batu dalam Map bisa diatur dengan mudah. Fitness ini berguna untuk mengurangi batu yang berdiri sendirian dan juga kelompok batu yang berukuran terlalu besar.

Secara keseluruhan, fitness yang diletakkan di lampiran A.3 ini berguna untuk mengatur ukuran dari kelompok batu yang tersedia, tetapi tidak mengatur bentuk kelompok batu tersebut. Berikut adalah isi dari Class Rock Groups Size Fitness

* 3-6 : Mirip dengan minRockAmount dan maxRockAmount di segmen program 6.6 (karena memang copas), dengan perbedaan 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 isi Tile dalam koordinat yang diberi bukanlah sebuah batu atau sudah pernah diperiksa, maka fungsi akan diberhentikan.
* 15 : Sementara bila koordinat berisi batu yang belum pernah diperiksa oleh 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) dan j(koordinat x) telah diperiksa dan diberi bool true di dalam koordinat isChecked sesuai dengan koordinat i dan j.
* 18 : Mempersiapkan Queue koordinat baru untuk dipakai.
* 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 nilai luas Map dikali dengan persentase RockAmount.
* 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 6.6 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 (minimal ada 1 kelompok batu dalam level), maka fitnessTotal akan dibagi dengan jumlah kelompok batu, membuat rata-rata Fitness. Lalu rata-rata tersebut akan dikuadrat ,dikali weight dan di return untuk dihitung Genetic Algorithm.



Gambar 6.3

Isi Variabel dari Rock Groups Size Fitness

Berikut adalah isi dari variabel di dalam Fitness Rock Group Size Fitness. Pertama 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 Map .

1. Area Fitness

Selanjutnya adalah Area Fitness yang diletakkan di lampiran A.4. Fungsi ini mencari area terberbesar di dalam Map yang dapat diakses oleh Player. Oleh karena itu, fungsi ini berguna 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 penjelasan mengenai kode di A.4 :

* 3-4 : Arraylist areasize digunakan untuk menyimpan ukuran dari seluruh area-area dalam Map yang ada. Misalkan terdapat 2 area yang terpisah, maka ukuran kedua area tersebut akan dimasukkan ke areasize. Selain itu, totalSize berisi jumlah seluruh Tile yang bukan batu di dalam Map.
* 9-10 : Bila Tile koordinat yang diberi merupakan sebuah batu atau sudah dicek, maka fungsi dihentikan lebih awal.
* 12-27 : Dilakukan pengambilan luas area seperti pengecekan kelompok batu di segmen A.3, 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 diperiksa dimasukkan kedalam arraylist areaSize.
* 34-38 : Di bagian program ini terjadi 2 hal. Pertama adalah penjumlahan 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 di dalam map. Bila ada area yang tidak bisa diakses, maka fitnessTotal akan berkurang tergantung dari luas area yang tidak bisa diakses. Setelah mendapat nilai fitnessTotal, fitnessTotal dikuadrat lalu di return.
* 45-49 : Persiapan variabel untuk perhitungan.

1. Power Up Access Fitness

Fitness berikutnya adalah Power Up Access Fitness. Fitness ini gunanya sederhana, dimana ada sebuah spawner bola spesial, maka bola spesial tersebut harus bisa diraih setidaknya satu karakter di dalam Map. Prinsipnya mirip dengan Fitness Area dimana semakin banyak bola spesial yang tidak bisa diraih, maka semakin kecil Fitness yang didapat. Berikut adalah isi dari Class-nya :

Segmen Program 6.7 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;

Segmen Program 6.7 (Lanjutan)

1. indexPlayer = j;
2. }
3. }
4. if (AStarAlgorithm.doAstarAlgo((Coordinate)lokasiPowerUp[i], (Coordinate)lokasiPlayer[indexPlayer], map) != null)
5. fitnessTotal++;
6. }
7. if (lokasiPowerUp.Count > 0)
8. return Mathf.Pow(fitnessTotal / lokasiPowerUp.Count, 2) \* weight;
9. else
10. return 0;
11. }
12. public override void resetVariables(){
13. fitnessTotal = 0;
14. lokasiPlayer = new ArrayList();
15. lokasiPowerUp = new ArrayList();
16. }
17. }

Penjelasan dari class Powerup Access :

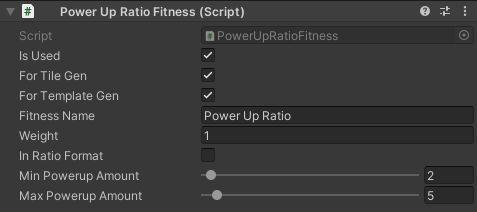
* 4-5 : Kedua arraylist tersebut dipakai untuk menyimpan secara koordinat seluruh player dan koordinat seluruh spawner bola spesial.
* 7-14 : Memberi salinan map kepada Fitness, lalu mengecek koordinat map yang diberi, bila koordinat yang diberi merupakan sebuah Tile karakter atau Tile bola spesial, maka koordinat tersebut akan masuk ke arraylist yang sesuai.
* 19-20 : Bila tidak menemukan seorang karakter, maka Fitness mengembalikan nilai 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 karakter terdekat dari bola spesial yang dipilih untuk mengurangi lama melakukan algoritma A\*
* 32-33 : Dilakukan algoritma A\* dari posisi karakter terdekat ke bola spesial yang dipilih sekarang ini, dan bila berhasil maka variabel fitnessTotal akan ditambah.
* 35-36 : Bila terdapat bola spesial, maka dikembalikan nilai Fitness berupa jumlah bola spesial yang bisa diakses dibagi dengan jumlah bola spesial dalam level, lalu nilai Fitness tersebut dikuadrat.
* 37-38 : Tetapi bila tidak ada power up yang ditemukan oleh Fitness, maka Fitness akan mengembalikan nilai 0
* 41-45 : Mempersiapkan variabel untuk fitness, itu aja sih.

1. Power Up Ratio Fitness

Sekarang akan diperdalam mengenai PowerUpRatio Fitness. Prinsip dari Fitness ini mirip dengan Rock Amount Fitness, hanya daripada menghitung batu, Fitness ini menghitung jumlah bola spesial. Selain itu Fitness yang diletakkan di lampiran A.5 ini juga menggunakan fitur pemilihan bentuk angka persenan atau bulat seperti Rock Groups Size Fitness, sehingga bisa dipilih diantara memanggil bola special secara bulat atau mengikuti ukuran Map.

Berikut adalah penjelasan dari segmen program A.5 :

* 3-8 : Seluruh variabel memiliki fungsi yang sama dengan variabel-variabel di Rock Group Size Fitness, kecuali min dan max PowerUpAmount yang bertindak seperti min & max Rock Amount dari Rock Group Size Fitness
* 12-13 : Bila Tile koordinat yang diberi berisi bola spesial, maka isi variabel powerUpAmount ditambah sebanyak 1.
* 16-41 : Fungsi ini persis baris 40-65 dari Rock Group Size Fitness(A.3), dengan satu satunya perbedaan diantara kedua segmen program adalah hasil Fitness dari fungsi ini direturn.
* 43-45 : Mempersiapkan variabel untuk pengambilan Fitness.
* 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 Chromosome Tile Generation.



Gambar 6.4

Isi Variabel dari Power Up Ratio Fitness

Sekarang akan dijelaskan mengenai variabel yang dipakai oleh Power Up Ratio Fitness. Dari 3 variabel pertama, Fitness ini dipakai apabila bila Player menggunakan Tile ataupun template generation. Selanjutnya diberi nama dan Weight seperti biasa. Lalu terdapat In Ratio Format yang dimatikan untuk memudahkan perhitungan. Variabel terakhir adalah range dari jumlah power up dalam sebuah level yang telah ditentukan diantara 2 hingga 5 spawner.

1. Template Variety Fitness

Dan fitness terakhir kita adalah Template Variety Fitness. Template Variety Finess yang dipakai di dalam Template Generation digunakan untuk meminimalis template yang kembar dari muncul di dalam Map. Tujuan dari menggunakan Fitness ini adalah untuk mendapatkan lebih banyak variasi template yang ada di dalam Map. Berikut adalah isi dari Class tersebut :

Segmen Program 6.8 Template Variety Fitness

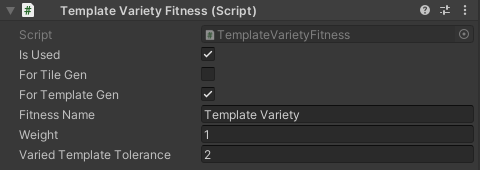
1. public class TemplateVarietyFitness : InLoopFitnessBase{
2. [SerializeField] int variedTemplateTolerance;
3. int[] savedMap;
4. public override void calculateFitness(int[,] map, Coordinate currCoor){
5. return;
6. }
7. //Khusus Template Variety karena dia pake template map bukan map biasanya

Segmen Program 6.8 (Lanjutan)

1. public void getTemplateMap(Gene[] map{
2. savedMap = new int[map.Length];
3. for (int i = 0; i < map.Length; i++)
4. savedMap[i] = (int)map[i].Value;
5. }
6. public override float getFitnessScore(){
7. SortedList<int, int> templateOccurences = new();
8. int temp;
9. for (int i = 0; i < savedMap.Length; i++){
10. temp = savedMap[i] >= 0 ? savedMap[i] : -savedMap[i] - 1;
11. if (templateOccurences.ContainsKey(temp))
12. templateOccurences[temp]++;
13. else
14. templateOccurences.Add(temp, 1);
15. }
16. int[] scores = new int[templateOccurences.Count];
17. for (int i = 0; i < templateOccurences.Count; i++){
18. temp = variedTemplateTolerance - templateOccurences.Values[i] < 0 ? variedTemplateTolerance - templateOccurences.Values[i] : 0;
19. scores[i] = 1 + temp;
20. }
21. float sum = scores.Sum();
22. if (sum < 0)
23. sum = 0;
24. return Mathf.Pow(sum / scores.Length,2) \* weight;
25. }
26. public override void resetVariables(){
27. return;
28. }
29. }

Dan berikut adalah penjelasannya :

* 3 : Variabel ini menandakan toleransi Fitness terhadap template yang kembar. Misalkan variedTemplateTolerance diisi 2, bila terdapat sebuah Template yang sama muncul 2 kali maka Fitness akan tetap 1, tetapi bila jumlah kemunculannya 3 keatas maka Fitness akan mulai berkurang.
* 4 : Variabel yang akan menyimpan Chromosome dari Map.
* 6-8 : Function yang tidak digunakan, karena fitness ini hanya memerlukan Chromosome Template.
* 10-15 : Mengisi variabel savedMap dengan Chromosome map yang diberikan.
* 18 : Membuat sortedList templateOccurences dengan sistem Key dan Value, dimana Key merupakan ID Template dan Calue adalah jumlah kemunculan template tersebut di dalam Map.
* 20-26 : Menggunakan sortedList dan sistem Key & Value, seluruh ID Template dimasukkan ke templateOccurences dengan Key dan Template ID yang sedang diperiksa. Bila Key tersebut tidak ada maka dibuat Key tersebut dan value diisi 1, sementara bila Key sudah tersedia maka value dari Key tersebut akan ditambah 1.
* 21 : Kode pada baris ini dilakukan agar Fitness tidak membedakan antara template negatif (yang memiliki bola spesial) dan positif (tidak memiliki bola spesial).
* 28 - 31 : Setiap jumlah kemunculan Template dicek, dan bila jumlah kemunculan dibawah atau sama dengan toleransi Fitness, maka akan dimasukkan nilai 1 ke dalam array nilai, tapi bila lebih banyak dari toleransi fitness, maka akan diberi skor 0.
* 32-34 : Masukkan jumlah Fitness yang dimasukkan ke dalam variabel sum, dan buat pengecekan. Bila nilai dari sum dibawah 0, maka buat nilai sum menjadi 0. Ini dilakukan agar tidak ada kesalahan dimana nilai negatif dikuadratkan menjadi positif.
* 35 : Nilai skor di rata-rata sesuai jumlah jenis template yang muncul, lalu nilai Fitness tersbut dikuadrat lalu dikali weight.
* 38-40 : 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 Generation. Lalu variabel selanjutnya yang perlu diketahui adalah Varied Template Tolerance, yang menandakan bila Fitness bisa mentoleransi kemunculan template 2 yang kembar sebelum mengurangi nilai Fitness dari Genetic Algorithm. Selain 2 variabel tersbut variabel lainnya memiliki fungsi yang standar seperti Fitness lainnya.