**LAMPIRAN A**

**KODE IMPLEMENTASI ALGORITMA GENETIK**

Segmen Program A.1 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. }
20. else if (tempID < oneWayTemplates.Length + twoWayTemplates.Length \* 2){
21. tempID -= oneWayTemplates.Length;
22. rotation = tempID % 2;
23. chosenTemplate = twoWayTemplates[Mathf.FloorToInt(tempID / 2)];
24. }
25. else{
26. tempID = tempID - oneWayTemplates.Length - (twoWayTemplates.Length \* 2);
27. rotation = tempID % 4;
28. chosenTemplate = fourWayTemplates[Mathf.FloorToInt(tempID / 4)];
29. }
30. int[,] resultTemplate = new int[5, 5];
31. if (rotation == 0)
32. resultTemplate = (int[,])chosenTemplate.Clone();
33. else if (rotation == 1)
34. //rotasi ke kanan
35. for (int i = 0; i < 5; i++)
36. for (int j = 0; j < 5; j++)
37. resultTemplate[j, 4 - i] = chosenTemplate[i, j];
38. else if (rotation == 2)
39. //rotasi ke 180 derajat
40. for (int i = 0; i < 5; i++)
41. for (int j = 0; j < 5; j++)
42. resultTemplate[4 - i, 4 - j] = chosenTemplate[i, j];
43. else
44. //rotasi ke kiri
45. for (int i = 0; i < 5; i++)
46. for (int j = 0; j < 5; j++)

Segmen Program A.1 (Lanjutan)

1. resultTemplate[4 - j, i] = chosenTemplate[i, j];
2. // kalau nomor yang diberi negatif beri power
3. if (id < 0 && resultTemplate[2, 2] == 0)
4. resultTemplate[2, 2] = 2;
5. return resultTemplate;
6. }

Segmen Program A.2 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

Segmen Program A.2 (Lanjutan)

1. if (useTemplatedGeneration)
2. map = putPlayerinTemplate(fc, SetObjects.getWidth() / 2, SetObjects.getHeight());
3. else
4. map = deflatten(fc, SetObjects.getWidth() / 2, SetObjects.getHeight());
5. fitness = fitnessFunction(map, fc);
6. return fitness;
7. });
8. //Metode milih ortu
9. var selection = new RouletteWheelSelection();
10. //Metode Crossover
11. var crossover = new UniformCrossover();
12. var mutation = new PartialShuffleMutation();
13. var termination = new FitnessStagnationTermination(StagnationTerminationAmt);
14. var ga = new GeneticAlgorithm(population, fitnessfunc, selection, crossover, mutation);
15. ga.Termination = termination;
16. ga.Start();
17. var a = ga.BestChromosome.GetGenes();
18. if (useTemplatedGeneration)
19. SetObjects.setMap(putPlayerinTemplate(a, SetObjects.getWidth() / 2, SetObjects.getHeight()), useMirrorFitness);
20. else
21. SetObjects.setMap(deflatten(a, SetObjects.getWidth() / 2, SetObjects.getHeight()), useMirrorFitness);
22. MMN.changeSceneIndex(-6);
23. }

Segmen Program A.3 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++;

Segmen Program A.3 (Lanjutan)

1. int size = 1, i = currCoor.yCoor, j = currCoor.xCoor;
2. ischecked[i, j] = true;
3. Queue<Coordinate> q = new();
4. Coordinate c, tempcoor;
5. q.Enqueue(new Coordinate(j, i));
6. //ngambil ukuran area 1 per 1
7. while (q.Count > 0){
8. c = q.Dequeue();
9. for (int k = -1; k < 2; k++){
10. for (int l = -1; l < 2; l++){
11. //Cek 8 Arah
12. tempcoor = new Coordinate(c.xCoor + l, c.yCoor + k);
13. 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]){
14. ischecked[tempcoor.yCoor, tempcoor.xCoor] = true;
15. q.Enqueue(tempcoor);
16. size++;
17. }
18. }
19. }
20. }
21. int maxRock, minrock;
22. if (inRatioFormat){
23. maxRock = Mathf.RoundToInt(map.GetLength(0) \* map.GetLength(1) \* maxRockAmount / 100);
24. minrock = Mathf.RoundToInt(map.GetLength(0) \* map.GetLength(1) \* minRockAmount / 100);
25. }
26. Else{
27. maxRock = Mathf.FloorToInt(maxRockAmount);
28. minrock = Mathf.FloorToInt(minRockAmount);
29. }
30. if (maxRock < minrock){
31. Debug.LogWarning("Min dan max ditukar");
32. (minrock, maxRock) = (maxRock, minrock);
33. }
34. float nilaiMinus = 0;
35. if (size < minrock)
36. nilaiMinus = minrock - size;
37. else if (size > maxRock)
38. nilaiMinus = size - maxRock;
39. float nilaiMinusMax = SetObjects.getWidth() \* SetObjects.getHeight() - maxRock > minrock ? SetObjects.getWidth() \* SetObjects.getHeight() - maxRock : minrock;
40. nilaiMinus /= nilaiMinusMax;

Segmen Program A.3 (Lanjutan)

1. //Fitness Total akan ditambah dengan 1 - beda antara ekspektasi dan jumlah batu per kelompok
2. fitnessTotal += 1 - nilaiMinus;
3. }
4. override public void resetVariables(){
5. ischecked = new bool[SetObjects.getHeight(), SetObjects.getWidth()];
6. rockGroupAmount = 0;
7. fitnessTotal = 0;
8. }
9. override public float getFitnessScore(){
10. if (fitnessTotal > 0){
11. //Kembalikan rata - rata
12. return Mathf.Pow(fitnessTotal / rockGroupAmount, 2) \* weight;
13. }
14. else
15. return 0;
16. }
17. }

Segmen Program A.4 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. }

Segmen Program A.4 (Lanjutan)

1. }
2. }
3. areaSize.Add(size);
4. }
5. public override float getFitnessScore(){
6. float biggest = -999;
7. // Ini aku pakai area yang bisa diakses player, bukan panjang \* lebar Arena
8. for (int i = 0; i < areaSize.Count; i++){
9. totalSize += (int)areaSize[i];
10. if ((int)areaSize[i] > biggest)
11. biggest = (int)areaSize[i];
12. }
13. fitnessTotal = biggest / totalSize;
14. return Mathf.Pow(fitnessTotal, 2) \* weight;
15. }
16. public override void resetVariables(){
17. areaSize = new ArrayList();
18. totalSize = 0;
19. ischecked = new bool[SetObjects.getHeight(), SetObjects.getWidth()];
20. }
21. }

Segmen Program A.5 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);

Segmen Program A.5 (Lanjutan)

1. }
2. if (maxRatio < minRatio){
3. Debug.LogWarning("Min dan max ditukar");
4. (minRatio, maxRatio) = (minRatio, maxRatio);
5. }
6. float nilaiMinus = 0;
7. if (powerUpAmount < minRatio)
8. nilaiMinus = minRatio - powerUpAmount;
9. else if (powerUpAmount > maxRatio)
10. nilaiMinus = powerUpAmount - maxRatio;
11. float nilaiMinusMax = SetObjects.getWidth() \* SetObjects.getHeight() - maxRatio > minRatio ? SetObjects.getWidth() \* SetObjects.getHeight() - maxRatio : minRatio;
12. nilaiMinus = nilaiMinus / nilaiMinusMax;
13. return MathF.Pow(1 - nilaiMinus, 2) \* weight;
14. }
15. public override void resetVariables(){
16. fitnessTotal = 0;
17. powerUpAmount = 0;
18. }
19. public float getRatio(){
20. return minPowerupAmount / 100;
21. }
22. }