Lesson 3 - Challenge

The goal of this challenge is simple, add in a size to your agents so that you can then train them to either go towards a population of larger or smaller people. Here I will provide some clues to help you along the way.

You will need a new DNA variable to control the selection of scale (*i.e.* size):

public float s;

Think about where this new DNA variable scale should be initialised. We can do that through the following:

// 1

this.transform.localScale = new Vector3(s,s,s);

// 1

Here we are using our Vector3 on a localScale of the person so that it scales to a uniform size across all dimensions for whatever size we feed into the algorithm. The scale will now need to be set up in the population manager. Once again, think about where this code belongs:

// 2

go.GetComponent<DNA>().s = Random.Range(0.1f,0.3f);

// 2

When the RGB values are set, a scale is also added ranging from 0.1 to 0.3 – this works well on my screen, you may need to adjust your ever so slightly!

Next, we will also have to adjust the breeding to include the scaling value:

// 3 Think about where this code belongs – note, you do not need to add another if/else or else

offspring.GetComponent<DNA>().s = Random.Range(0,10) < 5 ? dna1.s : dna2.s;

…else…

offspring.GetComponent<DNA>().s = Random.Range(0.1f,0.3f);

// 3

This code swaps the parents just as we did for the R, B, G values and then computing the random mutation.

Lesson 3 – Solutions

// DNA script

public float r;

public float g;

public float b;

public float s;

// DNA script

void Start () {

sRenderer = GetComponent<SpriteRenderer>();

sCollider = GetComponent<Collider2D>();

sRenderer.color = new Color(r,b,g);

this.transform.localScale = new Vector3(s,s,s);

}

// PopulationManager script

GameObject Breed(GameObject parent1, GameObject parent2)

{

Vector3 pos = new Vector3(Random.Range(-9,9),Random.Range(-4.5f,4.5f),0);

GameObject offspring = Instantiate(personPrefab, pos, Quaternion.identity);

DNA dna1 = parent1.GetComponent<DNA>();

DNA dna2 = parent2.GetComponent<DNA>();

if(Random.Range(0,1000) > 5)

{

offspring.GetComponent<DNA>().r = Random.Range(0,10) < 5 ? dna1.r : dna2.r;

offspring.GetComponent<DNA>().g = Random.Range(0,10) < 5 ? dna1.g : dna2.g;

offspring.GetComponent<DNA>().b = Random.Range(0,10) < 5 ? dna1.b : dna2.b;

offspring.GetComponent<DNA>().s = Random.Range(0,10) < 5 ? dna1.s : dna2.s;

}

else

{

offspring.GetComponent<DNA>().r = Random.Range(0.0f,1.0f);

offspring.GetComponent<DNA>().g = Random.Range(0.0f,1.0f);

offspring.GetComponent<DNA>().b = Random.Range(0.0f,1.0f);

offspring.GetComponent<DNA>().s = Random.Range(0.1f,0.3f);

}

return offspring;

}