Welcome back to the second part of creating this camouflage training application with genetic algorithms.

Just before we add the breed population method, I'll just get you to go all the way up to the top and

we're going to add in the link library up here.

So we want to use system dot link because it's got some really nice techniques built into it for sorting

lists, which is what we're about to do.

And without it and having to do it the long way around is a bit cumbersome.

So put that at the top.

Now let's go down and add this in.

I'll put it above update.

Of course, it doesn't matter where you put it and then I'll go through it for you.

So here it is.

It's called breed New population.

What we're doing is, first of all, creating a list to hold our new population.

So these are the offspring that we're going to generate.

Then I'm creating a sorted list.

So I'm getting hold of our population that we've already been clicking on.

And let's just go back up here.

At the end of the time trial, we're going to then order by a value that's in each individual's DNA,

which is the time to die.

So we're ordering it by time to die.

Now Time to Die gets set When you click on it and it's set to whatever elapsed time is.

So if you click on one of the people after one second, then their time to die is one second.

Anyone that's on the screen the longest is going to be like eight, nine or 10s And so they will be

at the end of the list.

So the individuals who died first are at the top and the ones that died last are down the bottom.

So it's now all nice and sorted.

So from that we can say that the fittest individuals in this population are in the bottom of that sorted

list.

And now with genetic algorithms, it doesn't matter really what you consider to be the most fittest.

In this case, I'm just going by the bottom half that are sitting in that particular array.

I'm going to say that they're the fittest, but there's no reason you couldn't pick the bottom third

or you could pick like the bottom 75%, three quarters if you wanted even more of your population.

The important thing really is to make sure that you breed enough so mix up the DNA of the individuals,

enough so that you end up with the same sized population again.

So cutting your list in half and then using the bottom or the top fittest people and mixing them with

every other fittest top person and getting them to produce two offspring each will give you the same

size array at the end.

Did you follow that?

I hope so.

Anyway, to do that.

First of all, I'm clearing the entire population list because remember, this sorted list has got our

sorted population and we're going to use this population list to repopulate with our new population.

So this next bit of code, which is the new part that I've just pasted in here because the previous

one had a typo in it.

What it's doing is it's looping around the whole list starting halfway down.

So you're going to get the top part of the list, the fittest people that are going to be bred together.

And then it uses I and I plus one.

So initially we're breeding the person halfway along and then the next one up from that and we continue

along the list, breeding one with the person above it in the list.

And so by the time we get to the end of looping around this loop, we've bred all the people in the

top half of that list.

Now to ensure the population stays exactly the same size.

You have to do it twice.

And you can see here I'm breeding AI with AI plus one and then turning around and breeding AI plus one

with AI.

And this will give you enough of a population size that you can continue going with different evolutions

and stay at the same population size so that you don't end up with an explosion of population issue.

So this is just breeding everybody with everybody else.

And then after we've done that, we're going to go through the sorted list and destroy everyone that's

in there.

So this is the old population.

We want to use a destroy to get rid of their game objects out of the system completely.

And then we update the generation count, which is showing on the screen to say that we're now in generation

number two.

Okay, so the breed, what's in that breed method?

Just above that method.

Let's add in this new one.

So the breed method is returning a game object because that game object is then going into the population

which is down here.

So we're adding whatever is returned into our population.

We're giving it.

Parent one and parent two.

Now, it doesn't matter which way around.

These are, as I said before, because we're just randomly swapping the DNA over When we create a new

little person, we need a random location for them to go.

The location isn't really important in this.

It's just providing you a bit of extra variety when you're clicking on them, I guess.

So we're just using that same pose as we generated before with random positions on the screen.

Then we're creating the actual instance of that person.

So we're creating that game object called Offspring.

In this case, with our instantiate exactly the same thing we did before.

But now instead of giving it random red, green and blue values as we did with its parent, we're going

to get the red, green and blue values from the parents.

So in this next line, I'm just getting hold of the DNA script using these two lines just to cut the

code length down a little bit.

So we're getting parent ones, DNA, parent Two's DNA.

This is the DNA script that you've got attached to the people.

And then we perform a swap.

So this is the fundamental basic operation that happens in all genetic algorithms.

You run through each value.

You've got all of the genes that you're storing and you randomly swap them between the parents.

And you can see here, I've just got a random dot range if it's less than five.

So I've got a sort of a 50% chance here.

50% of the time this offspring will get parent one's red channel.

The other 50% of the time it will get Parent Two's red channel.

And the same for the green and the same for the blue.

So we're not combining them as far as adding them together and dividing by two as an average, we're

actually swapping them.

Now I'm just going to say this again for emphasis.

This is what makes genetic algorithms work.

This is the complete, you know, the guts of the system.

This is the swapping of the values in the DNA.

And this is all you do.

Now, obviously, swapping these red, green and blue channels around are going to give you different

results for the color of the offspring.

And that's fine.

And you're about to see the amazing results of that.

If you think about humans and the way that humans breed and how we inherit what our parents have given

us, we if someone's got, say, blue eyes and someone's got brown eyes, then you don't get like a

mixture of brown to blue, You get one or the other.

There's no sort of combining.

And that's exactly what's running these genetic algorithms.

I can't emphasize that enough, actually, because this really is the most important part.

Okay, So you've got that.

Now what we're going to do is save it and we're ready to run.

So go back into Unity, press play, and you'll have 10s to click on whichever ones you want.

Now, this is going to train a color into the generation that you pick last.

So if it's about camouflage, then you're going to want to pick off the ones that are the brightest.

So this is obviously the brightest and so is this guy over here.

And we'd probably leave this guy and this guy here till last, because that will then make them the

fittest and they will be breeding because they'll be down the bottom of that array, that sorted array.

If you don't have enough variety in the colors here, then you won't be able to necessarily control

what's going on.

So let's say just by complete random chance, they all came up as purple people, then you're not going

to have any chance whatsoever of breeding that purple out because of those RGB values are just in there

and they're just going to keep going because the bottom half will still have purple color in them.

Let's anyway try and see if we can train into this group dull colors by keeping the duller ones till

last as we go through.

So you've got 10s once you start playing.

Okay, so the next generation is going to pick up on those duller colors, which it did.

It's still giving you some bright pinks, but we've got a lot more.

Ones that are a lot darker in color.

And now you can see I'm getting those darker.

Earthier colors coming through and I'm sort of forgetting what I'm clicking on here as I'm showing you

what's going on.

But you can see them.

They've now become, um, you know, quite brown, which is, I guess, a real good camouflage color

if you're in the bush.

Um, yeah.

Okay.

So after only like 4 or 5 generations in this case, they do get to be quite dull and they've kind of

achieved what they had to achieve.

So that's, that's pretty good.

It takes a lot more generations than that in the process of real evolution for that sort of thing to

occur.

But we're only working with like three values that we're swapping in and out.

Now, you could do this like with your favorite color as well.

If you go back into the population manager, what we could do is say you wanted to pick your favorite

color and you just want to pick that first.

And rather rather than leaving it till last, and then your fitness value becomes flipped around the

other way.

You want to keep anybody who's the right color.

And they were clicked first, so their time to die is going to be really small.

So you want to pick the top part of your sorted list.

Now you can do that with the link library really easily.

We've already got order by which is going to go from lowest to highest.

But if you want to switch that around, it's descending order by descending and it'll switch the order

rate around the other way so that you end up with the ones clicked on first down the bottom of your

sorted list.

So save that and then go back and see if you can train it to start becoming a population of your favorite

color.

Okay.

So I like green and purple.

So let me pick those two colors first and then I'll just get rid of the others in any particular order

and we'll see how we go.

Oh, I've still got a purple guy.

I'll leave him in there, try and get rid of these orange and reds.

Now they've totally bred out purple.

No hope whatsoever.

Now, this again is an issue with having such a small population is that you don't get the variety.

And in fact, I've managed to train something that's almost completely orange, which is kind of like

the opposite of my favorite color, which is quite interesting.

So another process in genetic algorithms that they build into these systems is.

Not just selecting the fittest and swapping the DNA, but the concept of mutation.

So in nature, animals and plants will be born that have a mutation in them.

Now sometimes that mutation might be bad and they die out really quickly, or it might be an advantage

for that particular species.

And so therefore they become fitter and they stay in the population.

How can we add this mutation into our code?

Let's go back and look at our breed.

So the mutation is going to happen where the DNA for the offspring is set and we could have some kind

of a chance that it's going to be set by the parents or it's going to be a mutation.

And therefore we could go if.

Random dot range between zero and you probably want this to be quite large so it doesn't happen all

of the time.

Let's see if we'll set it low because I do want to see it happen.

Let's do it 50% of the time again.

So if it's less than five, then we will get our information from the parent.

Otherwise.

-- editor.

Okay.

No.

Else, please.

Thank you.

Okay, so now we want to set these values here again, and we're going to.

Just use the random range that's already there.

Let's just delete this.

And we're after color values.

So just like we started with.

Setting them like that.

So set them to give you a random color.

Now, you won't see this at the start because you've already got random colors happening.

But as this continues through the process, we should just see after we've got a nice sort of color

thing going on with the camouflage system that we've been using before that.

I random color will just pop up out of nowhere.

Now, remember that we're working in reverse now, so we've got the ones that we click first are our

favorites and they're the ones that are the fittest.

So with this, we'll go through and I'll go and pick on green ones again.

So let's press play.

And oh, by chance I got a lot of green ones.

Okay, get rid of these ones and then we'll see what we get in the end.

More green, hopefully.

Yep.

Got some green.

Green chances are because we have that mutation set so high is that 50% of these guys are going to be

green and the other 50% are just going to be some kind of random color.

And you can see that here.

So it is doing the mutation, but it's not like you can see that.

1 in 1000 has a funny mutated color and you'd need to really increase the number that are on the screen

to see that single one come through and be mutated for you.

And often you want only a low number of mutations because you want to keep the genetic algorithm itself

in control of swapping the DNA and the fitness function itself.

When you when you put in a mutation, that mutation is totally ignoring the fitness function.

So it's not even caring if they're fit or not.

It's just not using their particular DNA.

So sorry.

Let's go back to the population manager.

If you want to put this mutation in here, in this particular example, I would say that you only want

to mutate a very small number of times.

And so what I would do is set this between 0 and 1000 and then have maybe if it's greater than five,

which it will be an awful lot of the time, then you use the parents and then for that small percentage,

then do a particular mutation and that will give you a much better effect if you get more and more mutations

that you don't want or that you're not ending up getting the genetic algorithm to optimize, which means

that it's finding a solution to your problem, which in this case is becoming your favorite color or

being camouflaged, whichever one you choose, then you can get rid of the mutations or just make sure

that they're only happening a very small number of times.

So.

That is the basics of genetic algorithms.

And from here you can just really go crazy with the different applications that you do and the generations.

And in the following lectures we'll move on and create some more complex scenarios where we've got a

whole bunch of agents learning things from their environment.