In this lecture, we're going to work through our first example of making a genetic algorithm, and

we're going to develop the little colored person game that you would have seen in the previous lecture

to begin with.

You'll need the resources, the unity package that is attached to this lecture called Camo GA Training.

Bring that into a new unity project.

And you can see I've opened that up here.

I've got the game window a little bit bigger than I usually do because most of the stuff's going to

be happening in here and we have to click on the little people as they come up on the screen.

Now, inside of the asset package that you get, there'll be a scene called Camo.

There's going to be a prefab called person.

And this person is a Sprite selected in the assets and just have a look in the inspector.

What you've got, you've got a transform, you've got a Sprite renderer and you've got a box collider

2D, that Box Collider is so that we can click on it to destroy it on the screen.

So it's really simple setup and the code is possibly even simpler than that.

We'll begin by creating a new C sharp file.

Call it DNA.

And that DNA is going to go onto a person so you can drag and drop it onto the person prefab and then

just make sure that it is attached by selecting it and having a look in the inspector.

And then we'll open that up and add in the code.

That will be the DNA for a person.

Now, if you remember from the previous lecture or possibly your biology classes in high school, then

you'll remember that we have DNA and DNA is contained within a chromosome.

So probably rightly, we probably should have called this class chromosome, but I've called it DNA.

And inside that we have genes.

And these genes are the things that we can pass along to our offspring.

In this case, we want our people to pass values off to the offspring.

And what they're going to be passing along is their color.

And that's the only thing that they need to impart on the children or the next generation.

So that means we need to store them somewhere.

And really, there's nothing fancy about what we're going to store this particular gene in.

It's going to be stored in a three float value.

You could put this into an array if you want.

You don't need to.

There's only three values.

And so we can put above here, let's say this is the gene for color.

Now, although it might be called Gene, which does one particular thing or has one sort of hereditary

value in it, it doesn't matter how it's represented because it can be three floats, an array, it

could be a structure of different values, it can be it can be really, really complex or just really,

really simple.

And in this case it is.

So you've got those three values.

Now, you also want to store for each person, whether they've been clicked on or not.

Now, when they get clicked on, we'll say that they have basically died.

So let's go Bull.

Dead equals false.

To start with.

Now, consider this to be like a shooting game or something.

And every time you click on a little person, you're basically shooting them and they're becoming dead.

And therefore, the ones that live the longest on the screen are going to be the fittest in this case.

And therefore the fittest population want to then breed with each other.

So this value will set to true after we have clicked on them.

The next value that we want to store will be public so that we can access it from outside and it will

be a float and it will be time to.

Die like that.

Let's set it to zero.

And when they do die, we'll record how long they lived for so that we can then later on sort them according

to how long they lived and keep the longest lived as the fittest and then breed them.

Now after this, we just need two more, I guess.

Just little helpers that we'll use in our code.

And this is access to the sprite renderer and the collider 2D.

And that's just so that we can turn them off when they've been clicked on in the start.

We then want to initialize our renderer and our collider values to point to the sprite renderer and

the collider 2D that is on this particular person, and that's so that we can use them when we click

on them.

So the code for clicking on something, let's put it up here.

We're just going to use an on mouse down, so no fancy raycasts or anything, just really simple way

to access when we've clicked on a collider and it looks like this.

So void on mouse down.

If you're not familiar with this, this is a unity owned method that you can use on a game object for

detecting when something with a collider has been clicked, when the user clicks on, the person dead

will become true.

And Time to Die is going to be set to the time elapsed in our population manager.

Now, we haven't created our population manager yet, so you will get an error when you try and save

this code, but just ignore that for now.

So what we're going to do is run the population for 10s and during that 10s you're going to be clicking

on the little people.

And when they die, it's going to record how long they lived for in that particular cycle.

And then we go to the next cycle or the next generation when we start the time again at zero and do

it again.

Now I've got a little debug in here.

This is just to check that you're clicking is working.

I probably just going to comment that out because I know it will work, but you can put that in there

and then we set our renderer and our collider to be disabled.

So just set them to both to be false so that you can't see that person because they're dead and you

can't click on them.

So get rid of their collider.

We still want the game object itself to exist in the hierarchy because we have to grab these guys and

breed them and we need access to their genes basically.

So that's important is turning them off, not destroying them.

All right.

So that you can save and don't forget, you will get an error.

Just ignore it, because that is all that you need in the DNA, in the hierarchy.

We're now going to add an empty game object.

This game object is going to manage our population for us.

So let's call it the population.

Manager.

And then we're going to create a script that goes on there and it will be called The Population Manager.

So C-sharp script.

Population manager.

Then drag and drop that onto the population manager in the hierarchy.

Okay, It's not going to let us because we've still got that error hanging over us because of the previous

code.

So we'll just have to go in and edit this.

So open it up and what we're going to do is add in a link to our game object prefab that we've got,

which is our little person.

So we want to public game object person prefab.

Like that.

And then we're going to set a population size because you can do this exercise with five, ten, 50,

100,000 little people if you want to.

Of course, anything like this with evolutionary computing, genetic algorithms and even neural network,

the bigger your population you're working with, the better outcomes you're going to get eventually.

Of course, this is a clicking game.

So if you're going to click a thousand people on the screen, it's not going to happen.

So we're just going to keep it small anyway, but we'll set it up to be a variable so you can change

it.

INT population size equals ten for now.

We also need to keep track of all of the people that we create, so I'll put a list up for that.

So let's call this a list of game objects.

And call it population and initialize it here.

New list.

Game object.

And we also need a timer in the environment that works out the elapsed time between them existing and

when we click on them.

So let's just create a public static float called Elapsed.

And this is the thing that's causing the error currently.

In the editor.

Once we've got that in there, this error we've got will go away and we will able to test it out.

Now, currently all we can do is really assign the person and once we've done that, we need to initialize

our game by creating the first population.

So there'll be ten of them and they'll all be person prefab.

We'll do that inside of Start.

So you want something that looks like this.

So this is a for loop.

It's going to loop around for population size number of times.

Okay.

So we're going to get ten loops out of this.

Each time we loop, we're going to work out a random position on the screen.

Now the screen is in X and Y because we're in 2D.

So the Z component in this case is just going to be zero.

And the range on the screen is from -9 to 9 in the X direction across here.

And from -4.5 to 4.5 in the Y direction, which is going up and down.

Now, I figured that out by putting something on the screen like my little person.

Let me show you that.

So if we just grab this little person, put them on the screen, there they are.

And then if we can find those over here.

Hit the W key and just move it like this.

And when you get to there, have a look at what its X position is.

So in this case -9.34 and I've put nine -nine inches the code, so that's all right.

And then go all the way to the other side to find the extreme in the x over there which you'll see as

positive nine and then you can do that for the Y direction.

Now lost my little person.

There it is.

And we can just go up there and then go down there and get those values for that range and then we'll

just delete.

All right.

So that's where these range values are coming from.

If you've got a different sized area on your screen that you're playing around with in the game view,

then by all means change this.

If you've got your camera at a different resolution, change it.

You might want to put 100 things on the screen.

This works for now with my particular settings.

Okay.

So once you've got this random for the person that you're creating at this time, we're going to create

the person.

So here we go.

We create a game object and we instantiate it from the prefab at the position and then we use a quaternion

dot identity.

This just gives us a zero rotational value default.

We do not want any rotations in this case.

We're just changing the position of that sprite on the screen.

Now, after we create our little person, we go ahead and set their DNA.

And remember their DNA has a red, green and a blue channel in it.

So we give them a random color value, basically.

And so RGB is between 0 and 1 for the colors where zero is none of that color.

And one indicates all of that color.

When it's 111, you've got white and when it's 000, you have black and everything else is in between.

Then once we've done that, we then add that new game object we've created to that list the population

so that we've got hold of it.

Okay, so you've done that, save it.

And then we're going to go back into unity and hopefully our error that we had before is going to disappear.

There it goes.

Select your population manager in the hierarchy.

Drag and drop that new script you just created onto the inspector.

And then for the person prefab that is exposed in the inspector, you want to drag your little person.

So grab your person prefab and drag and drop that into that exposed value there.

Now when we press play, we'll get ten people at random locations on the screen.

Now they're currently all white because we forgot to set the color up inside of the DNA script.

So just open up your DNA script again and go down to the start method here.

And that's where we need to use those color values that we've just set up through the population manager

and put them onto the renderer.

So go s renderer here.

Dot color equals new color.

Made up of R, G, and B, save that and we'll switch back to unity and we'll give it another whirl.

And this time we should get ten randomly located, multiple colored people, as you can see there.

Okay, so it's working well.

We should be able to click on them to actually, because we've already put that script in.

Okay, let's go back to our population manager and start adding in some code for breeding.

So the first thing we'll need at the top here is int trial time.

And this is how long we want each I guess little game trial to go for and let's go for 10s, which is

ample enough time to go around and click ten people and we're going to keep track of which generation

we're on as well.

So we'll store generation and we'll set it to one.

So the first population that we create that you see on the screen is generation number one.

Now the other thing that's really nice when you're doing this is to print out some stuff on the screen

so you can see what's going on.

Now I'm going to use on Gooey, which is our method I always use because it saves you setting up all

of the canvas stuff for something you're not going to continually use.

And here's just a simple on gooey.

It's going to first of all, I'm just setting up the font size and color so that you can see it in this

particular video using a gooey style which goes in the end of your gooey commands here.

So for these two labels, we're printing out generation plus whatever the generation value is, and

then the trial time followed by whatever elapsed is and elapsed is going to change based on the actual

time counter that we will put in.

So that's all done.

Let's now go down to the update where we will update our timer.

So add in this and what it's doing is it's increasing our elapsed value plus time delta time.

So it's just going to keep adding the time between frames which will show us on the screen the seconds

as they elapse.

Then if the elapsed time has become greater than the trial time, the trial is over.

So we've reached 10s.

Now what we're going to do is then run a new method called breed new population.

And this is where we will find out which of our members of the little people, tribe that have just

been created are the most fittest and we will breed them together.

Then we set elapsed time back to zero so we can start counting up to ten again.

So this is just going to keep cycling through ten second games essentially and continually breed the

population.

So we'll leave the code there for this lecture and we'll get stuck into the breed new population method

in the next part.