# Particle Explosion Worksheet

A useful technique for explosions, fireworks, water fountains etc.

The project is going to be created using OOP principles but to save time I won’t be using getters and setters, all attributes will be public and addressed with dot notation, no classes will be inherited. I’ve also used var a lot when defining variable because it’s easier. No other reason.

A screenshot of a computer

Description automatically generatedThis is the effect we are looking for. Multiple explosions in random colours, each decaying and fading out over time.

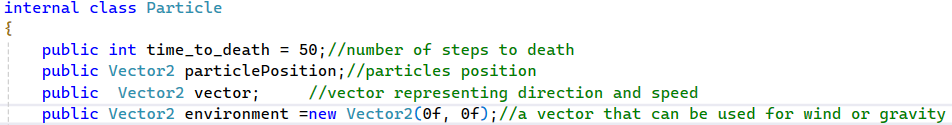
## Step 1

Create a new Windows Forms project using the dot net framework

## The Particle Class

Go to the project menu and Add Class. Call this new class Particle.cs (note the capital P)

These are the class attributes. Add them to the class code.



Note: the class is internal. This means it can only be used in the Namespace where it was created.

Now we need to make the constructor for the class.

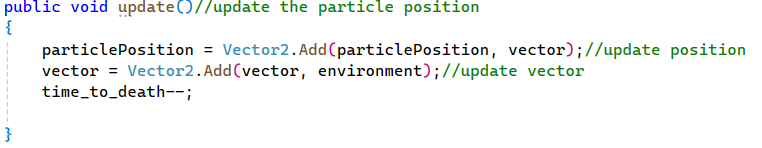
A computer code with blue text

Description automatically generated with medium confidence

The constructor receives the start position of the particle as X, Y co-ordinates and the direction and speed as vectorX and vectorY.

A Vector2 class in C# stores to float values. The polarity of each number decides the direction and the value of each number decides the speed. The two values are assessed together. We use the first Vector2, ParticlePosition, as an X, Y location on the screen but the vector attribute is used to determine the direction and speed. When I use the Vector2 Add method on the two it results in a new position for the particle. I can add in an environmental Vector as well to simulate wind or gravity. Set -2.0 on the X value of this Vector and the wind will blow, quite quickly, from right to left.

Finally we need to update the position of the particle using a new method Update( )

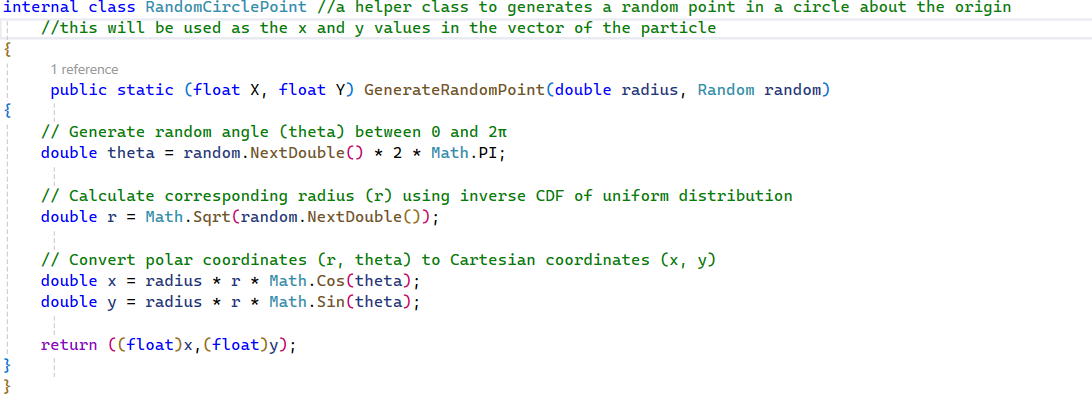


Note: the attribute Vector will be randomly generated. For that I’m going to create a helper class. This is a class that has one job, random numbers within a circle generation. We’ll do that now.

# The RandomCirclePoint Class

Create a new class as before. This one is called RandomCirclePoint.cs

It has one method. GenerateRandomPoint which receives the radius of the circle in which the points will be and a random seed. It returns two values. The X and Y location of that point relative to the centre of the circle. That’s important because these are going to be values in our vector.



Note how the function returns TWO values. A useful technique in C#.

Next we want to group a whole load of Particles together to make an explosion. For this we’ll create an Explosion class.

# The Explosion Class

Here are the attributes of the Explosion class. I’ll need a collection of Particles, I chose to use an array. A timer to trigger the regular updates of particle positions, a random seed to help generate random colours and variables to store the Red, Green and Blue values generated.

A close-up of a text

Description automatically generated

Next we create the constructor for this class.

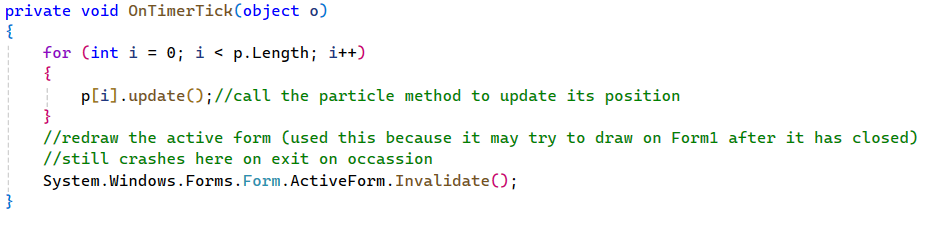
A screen shot of a computer code

Description automatically generated

First, I generate the R,G,B values for the colour. Not necessary, but fun.

Then, I’m iterating through the Particle array, generating a very random seed, calling the RandomCirclePoint method to get a new point to use as a vector for each particle, then I instantiate each particle into the array. Note: this method receives the start point of the explosion which is where we click on the screen. Once we’ve done all that we can create the new timer with a 100ms period.

The final method is for the timer tick event,



So, on each 100ms tick of the timer I update the position of each particle using the Particle.Update( ) method.

Then I use the Form.Invalidate( ) method to redraw the form. This still crashes sometimes when I shut the form and it tries to redraw a closed form. Work in progress.

Finally we can get to the Form1 class. This will have been generated automatically when you created the project. Go to the form designer. You’ll see it as Form1.cs[design] on the tabs at the top.

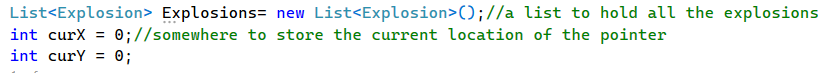
We need to set up a few things with the form. Make sure you have the form selected and the properties window open then do the following:

* Set the form BackColor to black
* Set DoubleBuffered to true. //this smooths the animations
* Select the Events button. It looks like a lightning bolt, and…
  + Double click the Click event
  + Double click the MouseMove event
  + Double click the Paint event.
  + Check each handler has been generated.

Now go to the Form1.cs tab and we’ll set up the class.

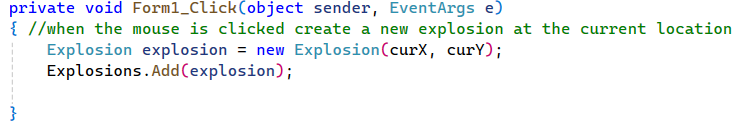
## The Form1 Class

The attributes for this class are as follows.

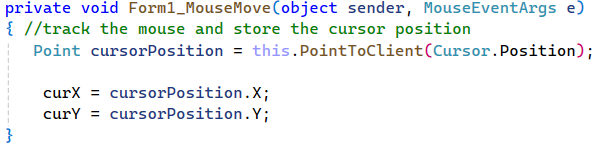


A list to hold all our explosions and two attributes to store the X, Y location of the mouse pointer

Here is the Form1\_click method. The header will have been generated for you. You can’t just type it in.

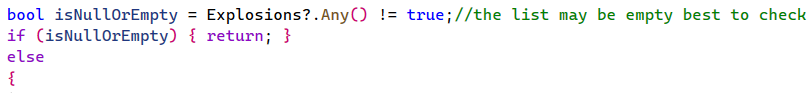
When the form is clicked we create a new explosion at the cursor location

Here is the Form1\_MouseMove method. This tracks the mouse position and stores its coordinates.



Finally, Here is the Paint Method. This one is going to be complicated because I need to iterate through each explosion THEN iterate through each particle in that explosion and draw its updated position. There will be loops within loops.

First, I’m going to check to see if the list is empty or if its null. Either will crash my program. I’m doing this a bit backwards because it was a fix to overcome constant crashes with null references.



If there is nothing there then return. Now for the else part…



Then the particle array

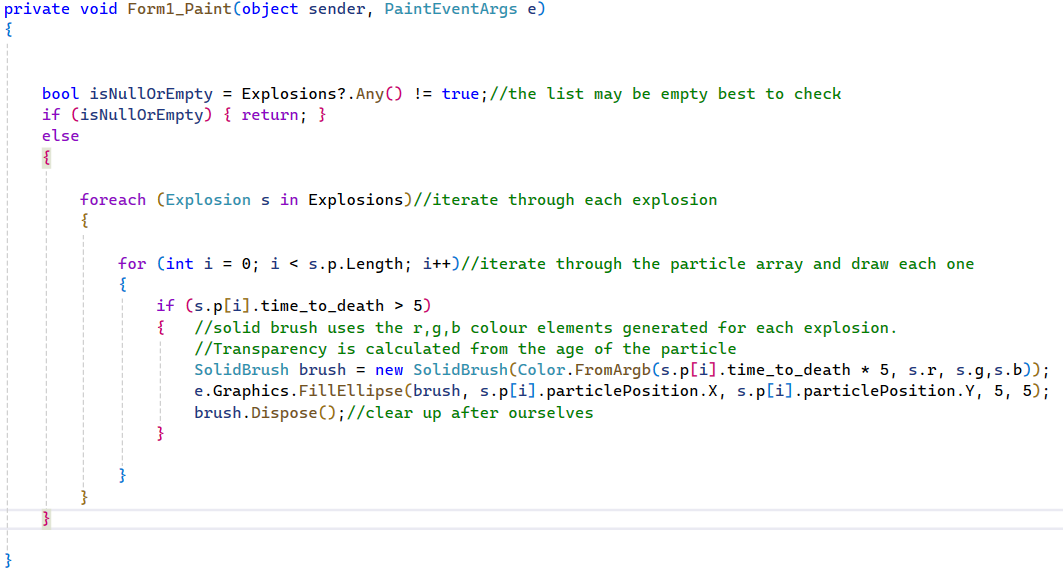


Finally we check to make sure the time to death isn’t too small. If that’s OK then draw the particle

A screen shot of a computer code

Description automatically generated

Here’s the whole thing



## Tasks

1. Find the environment attribute in the Particle class and apply a gravity effect. Explosions should move slowly to the “ground”. Show the code here.
2. Now try to change the environment to the explosion goes in a fast streak towards the top left of the screen as if a fast moving space ship just exploded.

Show the code here

1. Now try to change how long a particle lives for. What is the best value? Paste your code here.
2. Try changing the number of particles in an explosion. What is the optimum number? What value works best? Show the code here.
3. Try changing the size of a particle. You’ll have to hunt down this variable. Should this value be an attribute of the class Particle? Try moving this to the particle class, add it to the constructor and call this value when you paint the particle. There is no right way to do this but there might be a better way than mine.

Paste your code here.

1. Let’s get complicated. Change the particle from a solid ellipse to an unfilled circle. You’ll need to use a pen instead of a brush and will have to modify the Paint method on the form. Google is your friend here.

Show your code here.