C# Types

References refer to the actual data

Form f1 ; // Allocate the reference | f1 = new Form(); //Allocate the object

When a value in C# is assigned, it is copied. And a value cannot have the type null.

A immutable type appears in some ways as a value type, but is actually a reference type, once an instance of this type has been constructed, it cant be changed and allows a reference type to act similarly to a value type in some ways, a common immutable type is a string.

Assigning 1 to 2 references the same string, but changing 2 creates a new string, and if the original string that was referenced changes, 1 would also see the change.

Value types are numeric types, bools and user defined structs and enums and contain values not null

Reference types are are classes, interfaces and delegates and hold a reference to an object.

Parameters

Value paramaters = (int,bool,etc)

Reference Parameters = ref int, ref bool

Output parameters = out int, out bool

Parameter arrays = params int[] numbers

Implicit conversion require no syntax, and is usually for smaller to larger number conversion. Explict conversion = a= (value type)var name; | var = statically typed – type decided and errors caught at compile. Must be initalised at declaration.

Properties

Get/set functions are as easy as

Get {//return variable} Set{variable = value }; value = value we are assigning.

Delegates

Delegates are a type with store a reference to a method. They are similar in behavior top function pointers in C++, 0but are safer and cleaner to use. Can easily point to member functions and if it points to a virtual function, it will use the overriding version if available.

Example

delegate void MyDelegate(int,int);

static public void DOSOMETHINGSTATE(int I, int j)

static void Main(string[] args)

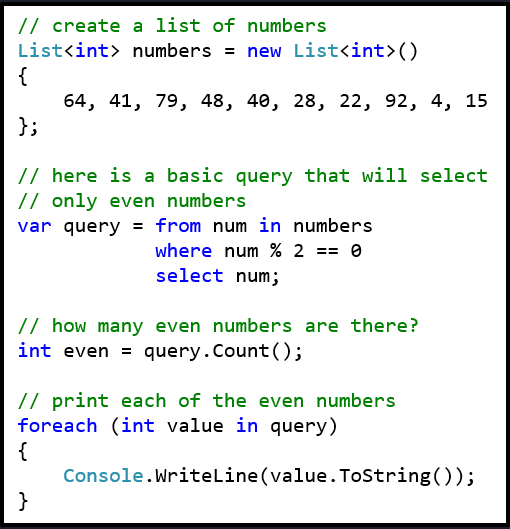
{

MyDelegate d;

D = DOSOMETHINGSTATIC

D(10,20);

}

Delegates can use += to have more then one function, but functions will get called in the order they are added.

LINQ

LINQ is very similar to SQL. It is used to make querys in C#.

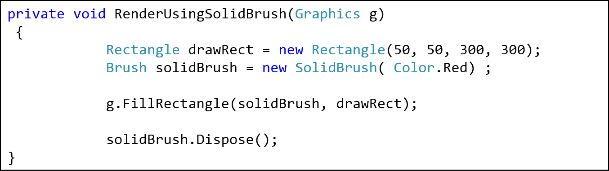
GDI

All GDI draw functions are called on a Graphics Object owned by the form/control that is to be modified

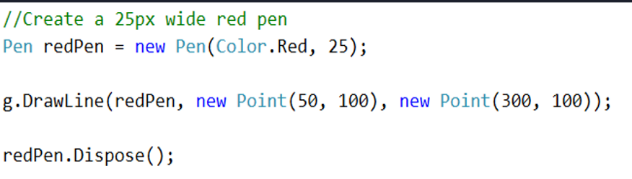
Pens and Brushes are commonly used in GDI

Pen redPen = new Pen(Color.Red, 25) creates a 25 pixel wide red pen.

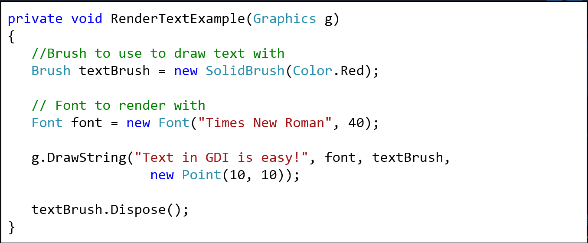
Always call a .dispose for pens and brushes at the end.

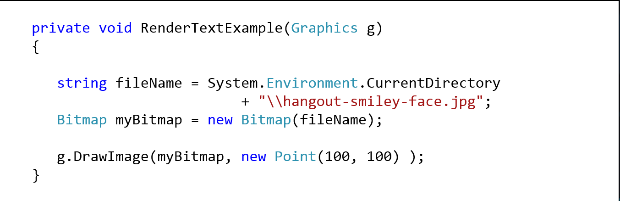
Brushes are used for all Fill<type> functions. (fillrectangle etc)

There is solidbrush, hatchbrush and lineargraident brush.



Pens are used by all Draw<type> functions (drawRectangle etc)

You can render text with the use of a brush and a font and then just calling DrawString.

You can also render an image with a bitmap file and calling the drawImage function.

Call Invalidate() to force a form to redraw.

Agents are AI in a game. They have 3 common steps, Sense, Think and Act.

Seek is V = normalize(P2-P1) x maxVelocity Force = V – CurrentVelocity and Flee, P1 and P2 swap places. Pursue is V = Target Pos + Target Velocity – Agent Pos Force = normalize(V) x maxVelocity – Agent’s Velocity, Evade is the opposite.

Agents that can see questions and respond are “experts”. An arbiter decides the best answer. Blackboards can centralize data, cache calculations, be used as a scratchpad area and to communicate between agents.

A\* = F = G + H where F is final score, G is the cost to travel from start node to the next node and h score is a heuristic, how far away we think it is from the goal.