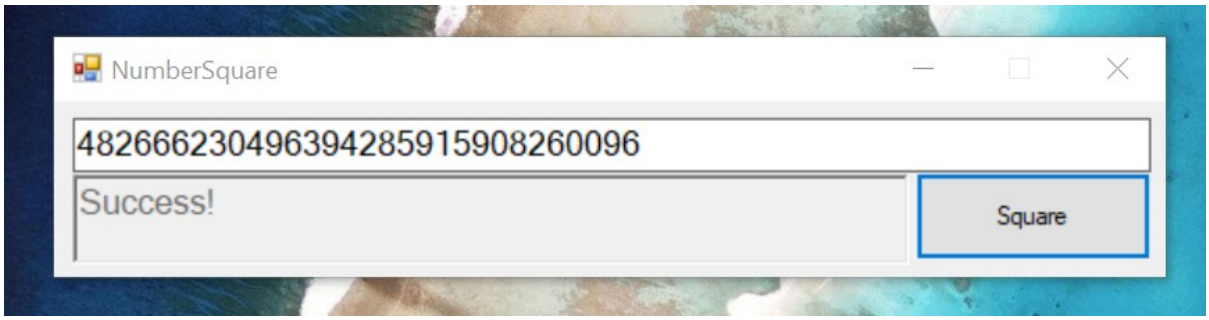
NumberSquare

Simple GUI program to square integers

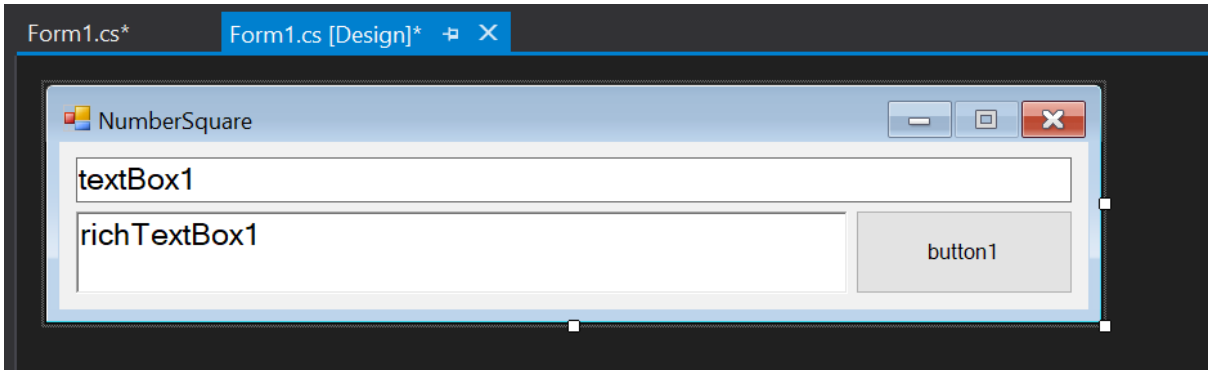
To demonstrate the different paradigms of computing common in this kind of development, namely Object Oriented, Procedural and Event-Driven, I created a simple GUI applet that takes an input, squares it, then outputs it. .NET GUI programs rely on events for user interactions, with each GUI element having corresponding event calls. Similarly, C# is an object-oriented language, with classes and methods and variables relying on an object structure. C# is also procedural, featuring if statements, for loops and while loops. This applet utilises all three paradigms described, with an event on the button, object references, and if statements for error checking and stability.



Event Driven Paradigm

This image is of the Form1.cs Design tab, showing the layout of the GUI elements as well as the names of each element. textBox1, for example, is a textBox, with a Text attribute, in this case below showing the name of the variable. textBox1 also has an event (which isn't being used) that is called every time the text is updated. This could be used to copy the text into another box, or check in real time whether or not the input is a valid integer.

Like the textBox, there's also button1, which is an example of a button class. This has an attribute of text which cannot be changed by the user, in this case showing the variable name, but in the actual solution shows the text "Square". button1 also has an event that is called when the user clicks on the button, presses space with the tab index equal to the button, or potentially hits enter. This event is where all of the code for this implementation resides.



An image showing the three events that these GUI elements interact with, the first of which being the only one containing code.

```
1 reference
private void button1_Click(object sender, EventArgs e)...

1 reference
private void textBox1_TextChanged(object sender, EventArgs e)...

1 reference
private void richTextBox1_TextChanged(object sender, EventArgs e)...
```

Object Oriented Paradigm and Procedural Paradigm

This is the contents of the button1_Click event, showing all the code required for this program to operate. This section uses frequent references to the textBox1.Text and the richTextBox1.Text attributes, contained within other objects in the namespace. As well as this, BigInteger requires a reference to System.Numerics, which was required to be added in the Refernences tab in Visual Studio. The way that these object references are important is that it's a simple and fast way to get user input from the GUI, far improved compared to a command line input that stops the program until an input is received. In other use-cases, these events could work as a subroutine, gathering information while a main loop is working, without stopping the whole program.

As well as this, BigInteger was required to be used as it allows for an undefined amount of characters for an integer, rather than Int64 which is 64 bytes and has a numeric limit. BigInteger, in this case, is stopped from going more 10,000 characters long, as calling the button function too many times causes system slowdowns and memory problems.

```
private void button1_Click(object sender, EventArgs e)
{
    BigInteger inputInt;
    BigInteger outputInt;
    if (!BigInteger.TryParse(textBox1.Text, out inputInt))
    {
        richTextBox1.Text = "Must input an integer (failed BigInteger.TryParse()).";
    }
    else
    {
        outputInt = inputInt;
        if (outputInt.ToString().Length > 10000)
        {
            richTextBox1.Text = "Output more than 10,000 characters long.";
        }
        else
        {
            outputInt = outputInt * outputInt;
            textBox1.Text = Convert.ToString(outputInt);
            richTextBox1.Text = "Success!";
        }
    }
}
```