Modifying a Process’s Memory in WPMTK

By Luke Kali, co-owner of sistemiinferno

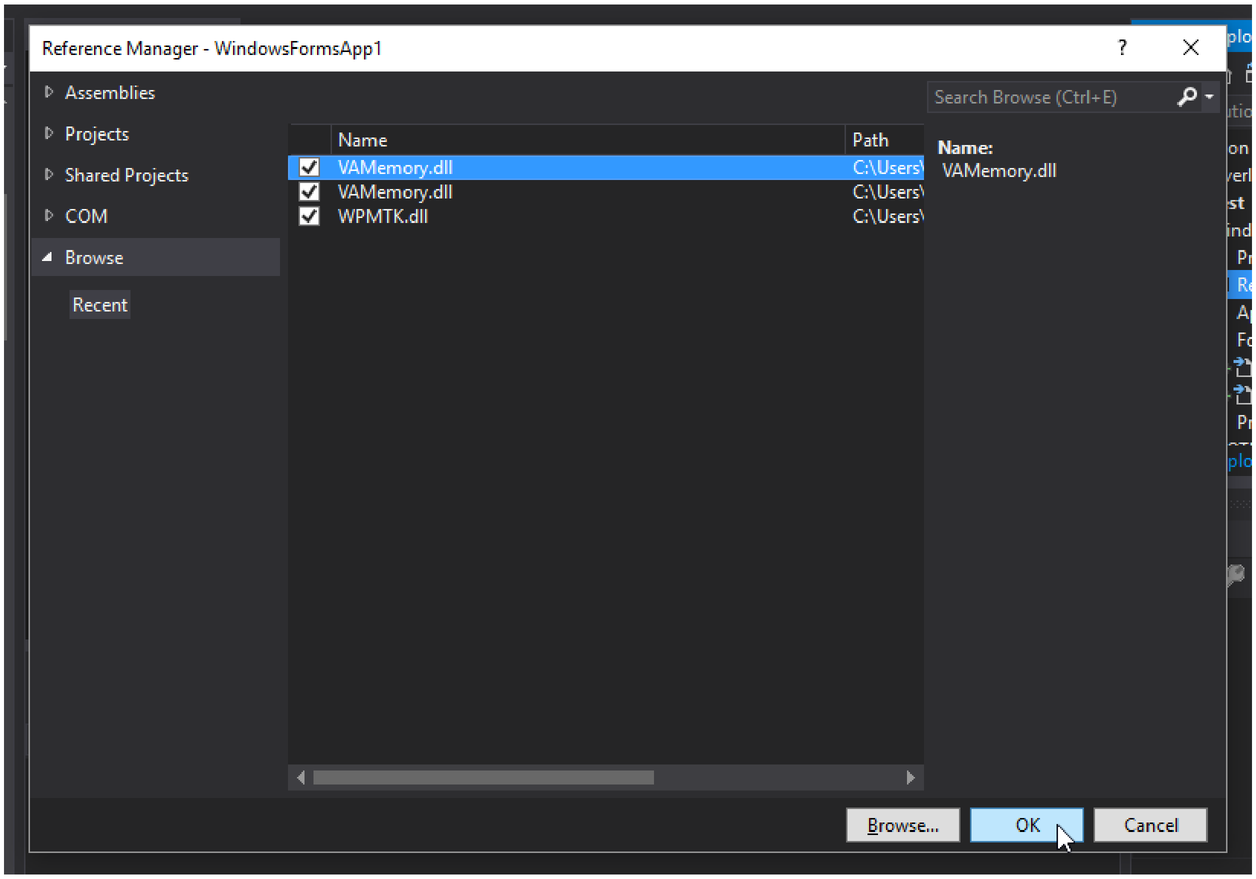
Setting up a C# project for WPMTK

Create any form of Visual C# project. The best would be a Windows Forms project, since we will be making one.

Getting your project to find the libraries

Configuring your Visual Studio project to find the libraries is a very easy process:

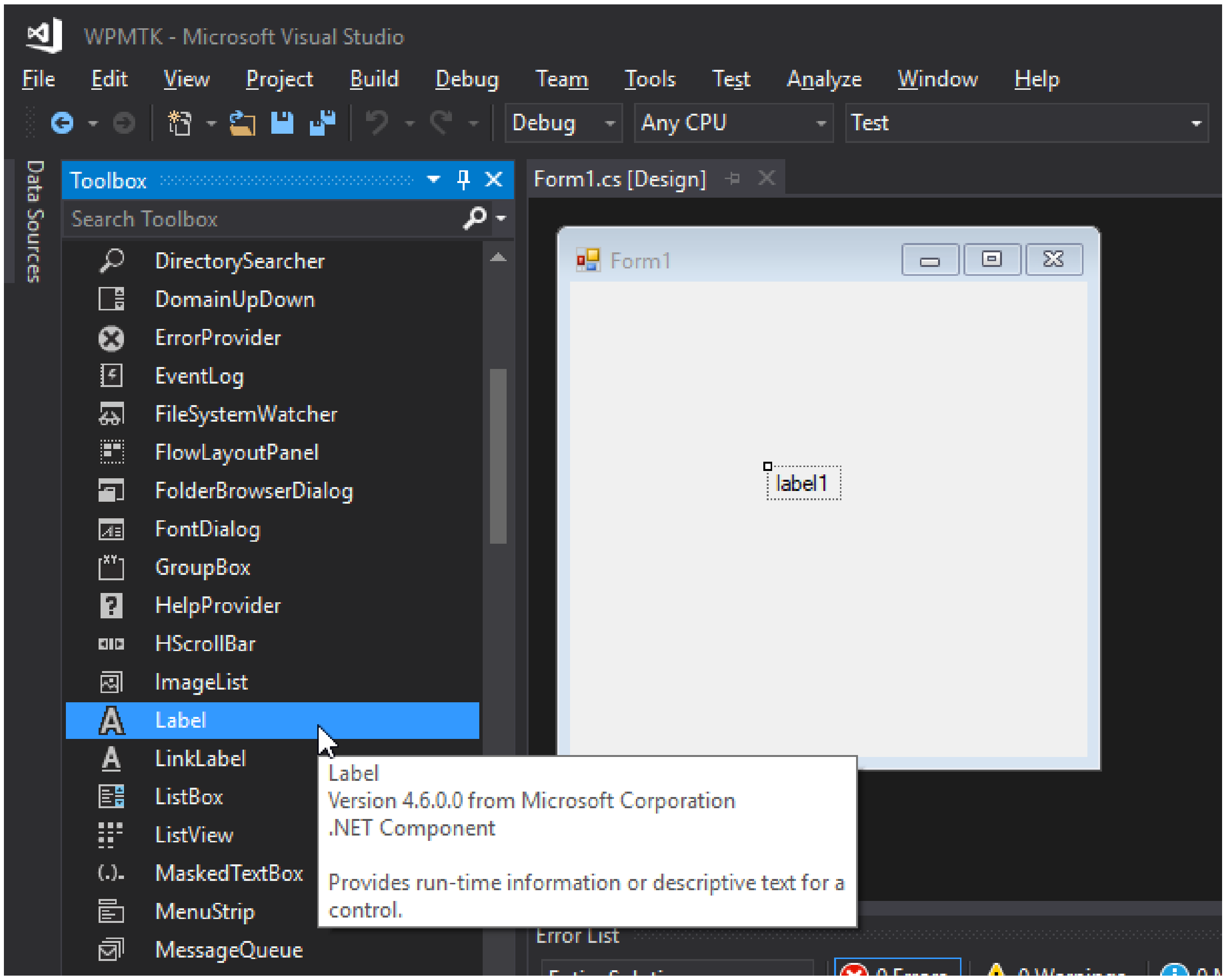
1. Right click "References" in the **Solution Explorer** and click "Add Reference".
2. At the bottom of the window, click "Browse…" next to the "OK" and "Cancel" buttons.
3. Locate the three dll files included with the WPMTK library: "wpmtk.dll", "wotk.dll", and "VAMemory.dll". (Tip: Hold Ctrl and select each of them before clicking the "Add" button.)
4. Click the "OK" button to set in the changes.



	Name	Path
✓	VAMemory.dll	C:\Users\
✓	VAMemory.dll	C:\Users\
✓	WPMTK.dll	C:\Users\

Name: VAMemory.dll

2. Drag and drop a **Label** component from the **Toolbox** window on the left, to anywhere in the window.



Label
Version 4.6.0.0 from Microsoft Corporation
.NET Component

Provides run-time information or descriptive text for a control.

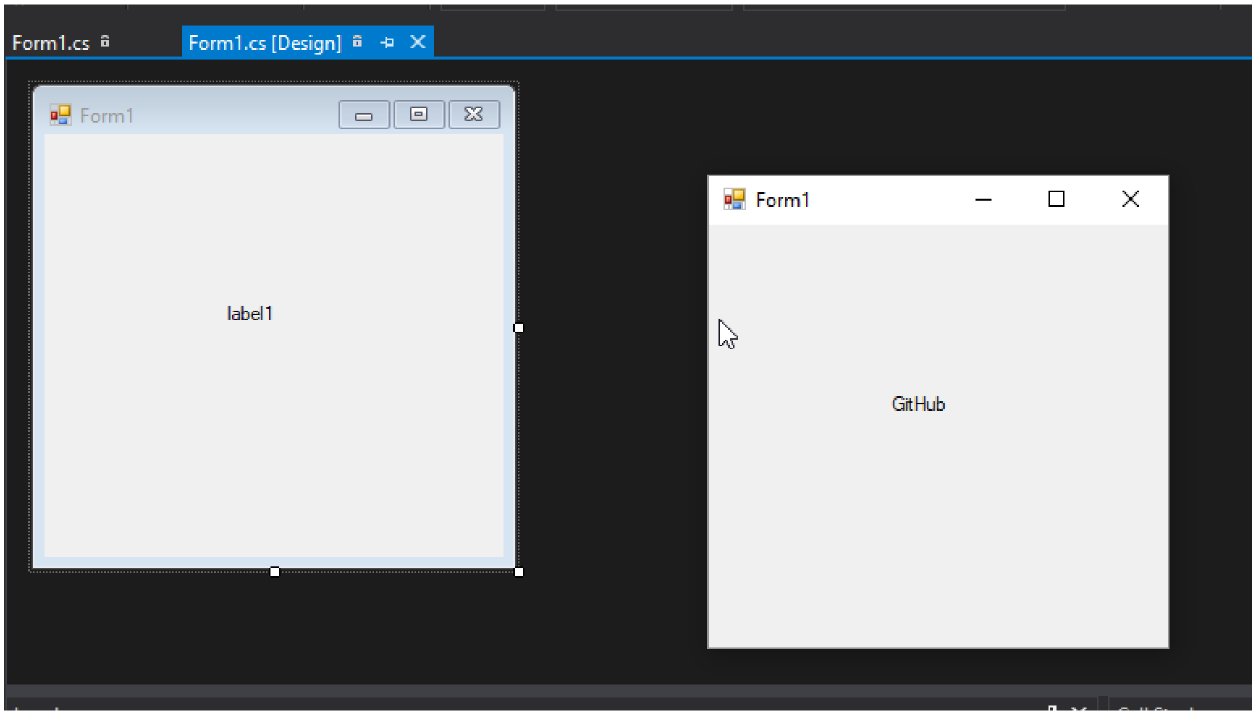
3. Double click the back of the **Form1** window. It should jump to the new event called “Form1_Load”.

4. Inside **Form1_Load**, place the following code:

```
private void Form1_Load(object sender, EventArgs e)
{
    WPMTK.Process process = new WPMTK.Process("GitHub");
    label1.Text = process.WindowTitle;
}
```

Please keep in mind, that you should rename “GitHub” to whatever game or process title you’re trying to hack. **It MUST be the name of the window’s title only, and not the actual process name.**

You should get a result similar to this:



Great job! Now what you can do from here is either learn how to use Cheat Engine to obtain memory addresses of a game's variables, and then use:

```
process.memory
```

To do all your useful memory address manipulation actions.

Next we'll learn about drawing an undetectable graphical overlay above our little **Form1**.

Creating an Undetectable Overlay in WOTK

By Luke Kali, co-owner of sistemiinferno

Initializing the overlay

The WOTK.dll library is an external library from WPMTK. It was developed by my partner in crime (co-owner of sistemiinferno); MasterUmbra.

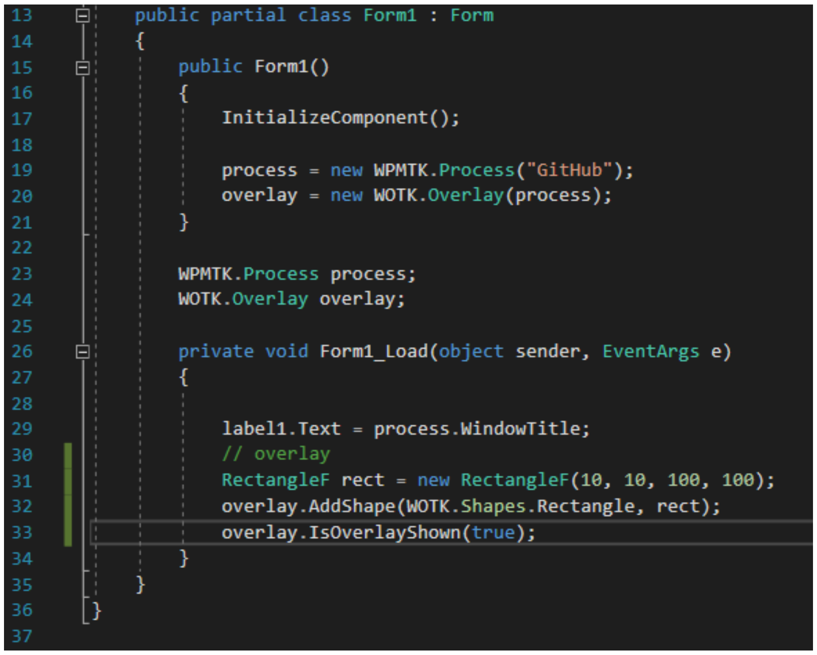
It was developed with the intention to be a small add-on for WPMTK. Its purpose is to create an invisible **Windows Form** control, and then handle the process of drawing graphics onto it.

This is a sublime way to create an undetectable overlay for games. What kind of anti-cheat system bans you for having other windowed processes open?

1. Go to the **Form1_Load** event again.
2. After the code we wrote in the last section, add the following:

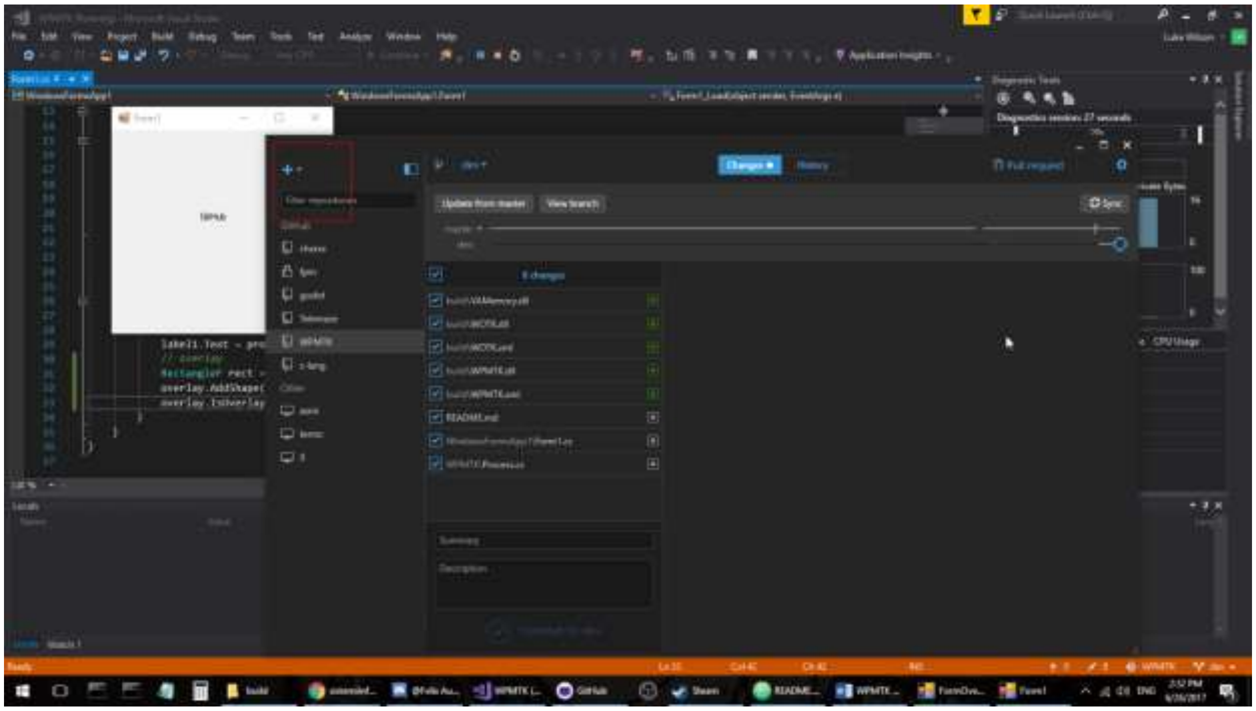
```
// overlay
WOTK.Overlay overlay = new WOTK.Overlay(process, false);
Rectangle rect = new Rectangle(10, 10, 100, 100);
overlay.AddShape(WOTK.Shapes.Rectangle, rect);
```

With a little bit of alteration (optimization and readabilty), you can get a source file like the following:



```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();

        process = new WPMTK.Process("GitHub");
        overlay = new WOTK.Overlay(process);
    }

    WPMTK.Process process;
    WOTK.Overlay overlay;

    private void Form1_Load(object sender, EventArgs e)
    {

        label1.Text = process.WindowTitle;
        // overlay
        RectangleF rect = new RectangleF(10, 10, 100, 100);
        overlay.AddShape(WOTK.Shapes.Rectangle, rect);
        overlay.IsOverlayShown(true);
    }
}
}
```

And after you run the project, it comes out like the following:



If you look closely at the top left of GitHub, you can see the red box we made from our project. It’s simple, right? :)

This is an undetectable overlay. We’re still developing WPMTK, but we can only really make a great library with your help. Please consider sending us feedback on what we should make next, or how we could make the library any better.

Thanks for reading!

I know that this simple tutorial may not work anymore after a few more updates, but after you get the general idea of creating a project and making use of the library, it’s really not that hard.

Just keep experimenting and trying to break stuff. :) And if you do find something not working as it should, feel free to message us at sistemiinferno@gmail.com. It really helps us when you do.

For help with documentation:

- Hover over our method names and there should be some useful information about what it does and what it requires.
- To browse our documentation, we release the DLL files with each their own (besides VAMemory.dll) an .xml file. The .xml file contains all of the documentation of the DLL.