# HaptiQ Manual

Simone Ivan Conte
sic2@st-andrews.ac.uk

April 4, 2014

# Contents

# 1 Introduction

This manual explains how to create WPF applications using the HaptiQ API and how to print the HaptiQ devices.
For further information or help, please contact the author of this manual.

# 2 API

This section will explain how to create a basic WPF application using the HaptiQ API.

## 2.1 Create a simple WPF project

1. Create a WPF (or Surface WPF) project as shown in Figure 1.
2. Add a reference of the *HaptiQ_API.dll* to the project. In Visual Studio: $Project \rightarrow AddReference \rightarrow Browse$ (see Figure 2).
3. Use the following template in the *XAML* view to get the input of the HaptiQ correctly (aligned with window):

```
<s:SurfaceWindow x:Class="TestApplication.SurfaceWindow1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/
        presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:s="http://schemas.microsoft.com/surface/2008"
    Title="TestApplication"
                 mc:Ignorable="d"
                xmlns:d="http://schemas.microsoft.com/
                    expression/blend/2008"
                xmlns:mc="http://schemas.openxmlformats.org/
                    markup-compatibility/2006"
                SizeToContent="WidthAndHeight"
    >
    <Grid Name="Container" Height="700" Width="1000">

  </Grid>
</s:SurfaceWindow>
```

   The name and size of the container are arbitrary values.
4. Within the code view add the following line of code to start using the API:

```
using HaptiQ_API;
```

5. To start the HaptiQsManager:

```
HaptiQsManager.Create(this.Title, "SurfaceInput");
```

Change *SurfaceInput* with *SurfaceGlyphsInput* to use glyphs rather than bytetags

6. Always remember to dispose the HaptiQsManager and its resources. This is usually done in the OnClosed method of the WPF Window:

```
protected override void OnClosed(EventArgs e)
        {
            base.OnClosed(e);
            RemoveWindowAvailabilityHandlers();

            HaptiQsManager.Instance.delete();
            Application.Current.Shutdown();
        }
```

7. Add Shapes where and when appropriate as follows:

```
HapticShape shape = CREATE HAPTIC SHAPE
shape.color(Brushes.Brown);
Container.Children.Add(shape);
```

The API contains a multitude of functionality. For more information it is suggested to use Visual Studio, which allows easy access to the documentation for the API. The *Templates* contains an example that can be used as a starting point for a new application.

## 2.2 HapticShapes

The API supports the following HapticShapes:

- Rectangle

```
HapticShape rectangle = new HapticRectangle(x, y, width, height);
```

- Circle

```
HapticShape circle = new HapticCircle(x, y, radius);
```

- Link

```
HapticShape link = new HapticLink(src, dst, linkHasDirection);
```

- Line

```
HapticShape line = new HapticLine(new Point(x, y), new Point(x_1, y_1));
```

- Polyline

```
List<Point> points = new List<Point>();
points.Add(new Point(x, y));
...
HapticShape polyline = new HapticPolyline(points);
```

HapticShapes are created for WPF applications. However, client applications can implement the *IHapticObject* interface to support tactile objects for XNA application as well.

## 2.3 Extending the API

The API provided does not consider all possible types of applications users may want to implement. However, it is possible to extend it to add new functionality. The user can extend the API from the client side without having to modify the API.

Note that it might be necessary to add a reference to the *Input_API.dll* when extending HapticShape.

# 3 Hardware

The models for the 4-HaptiQ and the 8-HaptiQ can be found in the folder *HaptiQ 3D Models*. Use MakerWare to import the STL files and print all the interested parts. Note that the printing can take up to several hours, based on the 3D printer used.

## 3.1 Hardware needed

- 3D Printer with 1.8mm ABS Plastic or equivalent
- 4 Hitech HS-65MG micro servomotors (8 for the 8-HaptiQ)
- 1 Phidget AdvancedServoBoard
- 1 Phidget InterfaceKit 8/8/8 or equivalent (must support analog input)
- 4 Force-Sensing resistors (8 for the 8-HaptiQ)
- Screws of various dimensions
- Super glue
- Female and male crimps (with slots)

# 4 Software Listing

The software used in this project is listed below:

| Software | Description | Available at |
|---|---|---|
| Phidget libraries | Control servos and pressure sensors | `http://www.phidgets.com/` |
| GRATF | Locate and recognize Glyphs | `http://www.aforgenet.com/projects/gratf/` |
| Glyphs Studio | Print Glyphs | `http://www.aforgenet.com/projects/gratf/` |
| Surface SDK 2.0 | Interface with tabletop | `http://www.microsoft.com/en-gb/download/details.aspx?id=26716` |
| NCalc | Evaluate mathematical functions in FunctionsApp | `http://ncalc.codeplex.com/` |
| VS2012 | C# IDE | |
| MakerWare | 3D Printer Software | `https://www.makerbot.com/makerware/` |
| SketchUp | 3D Modelling tool | `http://www.sketchup.com/` |
| SketchUp STL | STL Exporter plugin | `http://extensions.sketchup.com/en/content/sketchup-stl` |

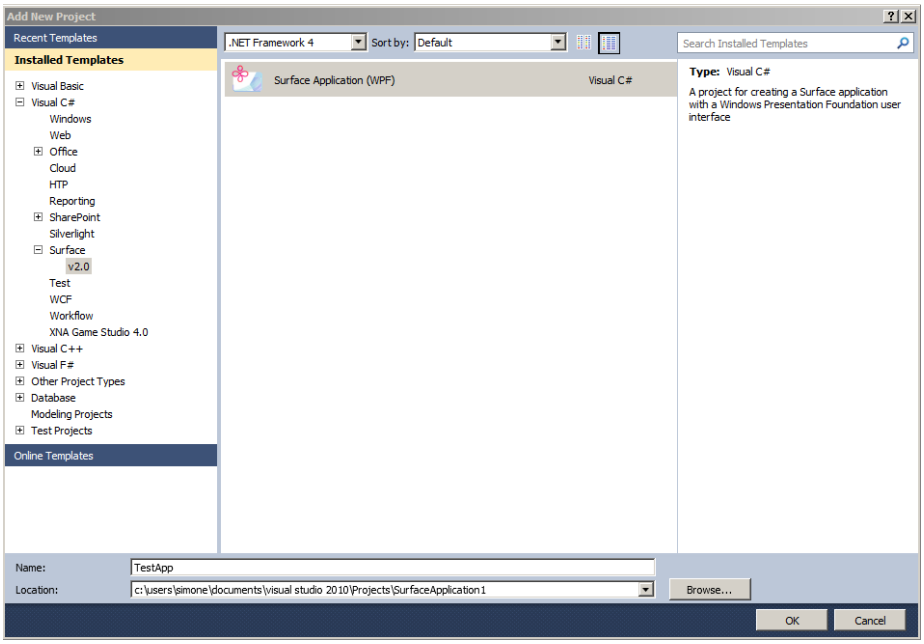


Figure 1: Create a WPF project

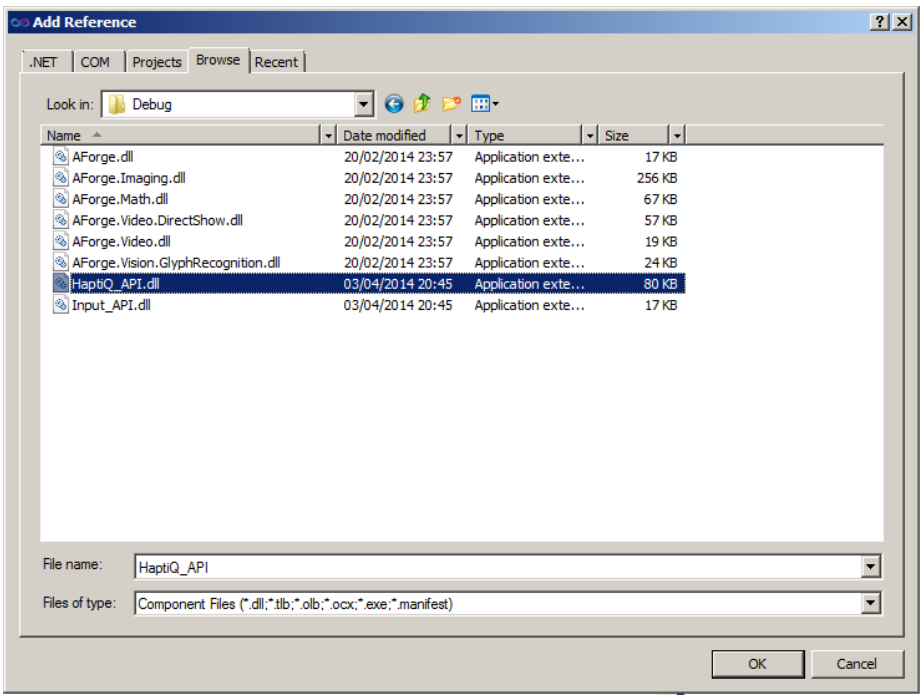


Figure 2: Add reference to HaptiQ API