### IN628 Practical 1.2: Components Review

1. Make a Button that grows 10 pixels wider each time it is clicked. When the right hand edge of the button reaches the right hand edge of the Form, change the Text of the button to “Too Big”.
2. Make a Form with a TextBox and a Button. When the button is clicked, its Text changes to equal the contents of the text box.
3. Make a Form with a Button and a ListBox. Each time the button is clicked, add a new line to the ListBox. The line should read “This is line *n*”, where n is the line number.
4. Make a Form with a Button which, when clicked, toggles its Text between “On” and “Off”.
5. Make a Form with a Button, a TextBox and a PictureBox. Using the Property Inspector, set the BackColor of the PictureBox to something other than grey, so that it is clearly visible. The user enters an integer value into the TextBox. When the Button is clicked, the PictureBox moves n steps to the right, where n is the number entered in the TextBox. Each step should be 10 pixels. After each step, the program should pause for 100 msec (otherwise the PictureBox moves so fast you can’t see it). To pause for n milliseconds, use the command System::Threading::Thread::Sleep(n).
6. In this problem, you will dynamically change the image contents of a PictureBox control. To do this, you modify the PictureBox->Image property. You set it to an Image object, which is created by calling Image::FromFile(“filename”). For example, you can display the picture file bob.jpg as follows:

pictureBox1->Image = Image::FromFile(“bob.jpg”);

On the I: drive you will find a set of 9 images called Dragon1.bmp, Dragon2.bmp, etc. Make a Form with a Button and a PictureBox. When the button is clicked use technique described above to load the PictureBox with each of the dragon pictures in order from 1 to 9 to produce a simple animation. Use Sleep(n) as above to control the timing, and use Application::DoEvents() to refresh the screen after sleeping. (Hint: This problem can be solved by brute force, or more elegantly by using a for loop and the string concatenation operator “+”).

1. C++/CLI applications do not automatically provide a canvas. Instead, you must create a *Graphics* object to paint on. To create such an object use the following code:

Graphics^ mainCanvas = CreateGraphics();

You must be using System::Drawing to have access to the Graphics class.

Note that only some C++/CLI classes expose the CreateGraphics() method. The Form is one of them. So you can create a Graphics object as shown above in any Form method (e.g. in the Form\_Load or in any button click handler).

After you have created a Graphics object you have many methods for drawing on it. You can draw shapes, images, text, lines, etc. Today we will use only one of the many Graphics methods: FillEllipse.

The syntax for FillEllipse is: FillEllipse(*Brush, x, y, width, height*)

The parameters *x* and *y* are the coordinates of the upper left corner of the ellipse, and the parameters *width* and *height* are its horizontal and vertical dimensions. The Brush is another member of the System::Drawing namespace. Brushes determine the colour *inside* a drawn shape. (There is another object *Pen* that determines the colour of the outline of a shape.) Before calling FillEllipse, you must create a Brush. The full syntax to create a SolidBrush (there are other varieties we will see during the term) is:

Brush^ greenBrush = gcnew SolidBrush(Color:Green)

Given the above discussion, what do you think happens when the button shown below is clicked? Test it and see.

private: System::Void button7\_Click(System::Object^ sender, System::EventArgs^ e)

{

Graphics^ mainCanvas = CreateGraphics();

Brush^ greenBrush = gcnew SolidBrush(Color::Green);

mainCanvas->FillEllipse(greenBrush, 100,100,10, 10);

}

Using the techniques just described, make the following: Add a Timer to your form. At each Timer tick, draw an ellipse at a random location on the screen. The ellipse should be of random width between 0 and 100 pixels, and random height between 0 and 100 pixels. Arrange to have approximately equal proportions of red, green, blue and yellow ellipses. Include a button that turns the timer on and off.