GDAPS1 – Practice Exercise

Windows UI – The Hard Way

# Objective

You will be creating a window with a large grid of buttons. The buttons should respond to the mouse passing over them by using built-in events. Each button’s event should call the same method (rather than having a separate event handler method for each button).

## Initial Setup

1. Create a **CONSOLE** project (do NOT make a Windows Forms project)
2. Right click on the References folder in the Solution Explorer
3. Choose Add Reference…
4. Go to the “Assemblies” tab (on the left)
5. Add the following references by clicking the check box next to them:
   1. System.Drawing
   2. System.Windows.Forms

## MyForm Class

Create a new class named MyForm.

Make sure your class inherits from the built-in Form class, which is in the **System.Windows.Forms** namespace. Add a *using statement* for that namespace, as well as **System.Drawing**, as you’ll need it later.

After this point, if you close your form class code window, you’ll need to right click on the class in the Solution Explorer and choose “View Code” to open it again. More on this next time!

## Main Method

Test out your form so far in your Main method. Add the System.Windows.Forms namespace at the top of Program.cs and the following lines of code in Main:

Application.EnableVisualStyles();

Application.Run( new MyForm() );

Test your program by running it with F5.

## Finishing up your Form

Create a constructor for your MyForm class. Have it alter the *Size* property of this form by settings it equal to a new Size object, passing in the width and height you want. Also change this form’s *Text* property to display a custom message in the title bar.

**Test** your form again. It should be a different size and have a different title this time.

**Making 100 Buttons**

To create a single, clickable button you can simply make an object of the Button class and add it to the form’s Controls collection. Controls is a property of a form, and it acts essentially like a List.

Make 100 buttons, setting them up in a 10x10 grid on the form. You may need to adjust the size of the form and/or the buttons to get them to fit properly. You’ll also need to move the buttons by changing their *Location* property, which is a Point object.

Points are structs, so you can’t simply say button.Location.X = 10. Instead, you’ll need to make a new point and overwrite Location with that new Point.

**Make the Buttons Interactive**

For this part, you’ll be hooking up each button’s **MouseEnter** event. This event is automatically triggered when the mouse moves onto a button. When it does, you’ll be changing the color of the button (either to a set color or a random color).

There is already a delegate, called **EventHandler**, defined for this kind of event. You just need to make a method that matches that delegate, and have each button’s event use that method. The delegate looks like this:

void EventHandler(object sender, EventArgs e)

Create a method in the Form that matches the EventHandler delegate. Every button will be using this single method, so how will you know which button actually triggered it? That’s what the *sender* parameter tells you. It’s the object on which the event occurred. Since you know it’s a button, you’ll need to cast it to a Button and then alter its BackColor property to change its color.

**Last Step**

The last step is to go back to your button generation loop and hook each button’s MouseEnter event up to the method you just created.

When you run, you should see 100 buttons, which change color when moused over.

# Submission

All of your work must be commented and follow this course’s coding standards. **Read through the Coding Standards document (located in MyCourses) to check over your code before you complete your program. Make sure you follow the coding standards for all code you create.**

1) Submit: Submit your program to the appropriate Assignments dropbox in MyCourses.

2) Check-off: Show your working program to the instructor or TA. If you do not finish before class ends, complete the exercise for homework and show one of us in-class on the next class period. If your program works as expected, you will be “checked off” to earn credit for the exercise.