CS 414 Homework 1

Overview

The purpose of this homework is to gain some initial experience with the Visual Studio programming environment. You have our permission to work in any of the supported languages (C, C++, Java, J++, Visual Basic, C#). However, our recommendation is to use C#, which is almost identical to Java but uses different libraries (the "System" class). We may not be able to help much if you decide to use some other language and then can't figure out how to do the assignment!

Skills we want you to learn

Creating a Windows form with some controls on it. Handling a "button click" event. Defining a new object class, with one thread for each object instance. Changing the position of a Windows control and redrawing it as needed. Thread sleep primitive.

Grading

Assignment 1 will be graded on a 100 point scale. The value of this assignment is approximately $1/8^{th}$ of your overall homework score for CS414

Due date

Assignment 1 is due in 2 weeks from today, on September 21. Please use CMS to hand in your solution. In fact this project can be done in about 15 minutes, but we're giving you ample time to learn Visual Studio and C# -- time you'll need, so don't waste it!

Learning C#

You can teach yourself C# (if you already know Java or C++) from either printed books or online materials. If you want to purchase a textbook with detailed "how to" information, we recommend the "C# HOW TO PROGRAM" textbook, from Deitel. This offers very simple to follow, step by step instructions for learning C# and using Visual Studio. But there are a great many books available on C# and you can go with any of them.

The assignment

The goal of this assignment is to create a small Windows application that has a box and two colored rectangles that bounce around within the boundaries of the box. *They do not need to bounce off one-another for this stage of the assignment. We'll add that sort of thing later.* Each colored box will have a direction and a speed (basically, for each unit of time, the box will move (d_y, d_x)). Striking a wall will cause the box to rebound at the same speed in the opposite direction. For example, if a box strikes the left wall, its new speed will be (d_y, $-d_x$).
Feel free to ask for help from us, or from a friend who knows a lot. Our talented undergraduate TAs are going to try and work in the CSUG lab and will have some form of sign up during their

office hours (posted on our web page): "Ask me about CS414 problems." But even with all this help available, *YOUR WORK MUST STILL BE YOUR OWN*. Help does not mean copying someone else's program!

How to build it.

First, obtain an account on the CS undergraduate lab, or obtain a (free) copy of Visual Studio for C# from the CS department. There is a web page about obtaining free software from the department. It is possible that we may not be able to provide Visual Studio for your machine right away, or even at all. In that case, just use the undergraduate lab for this work.

Next:

1. Log in and launch Visual Studio for C#
2. Ask it to create a new project. Pick C# and "Windows Application. You'll be placed into the "Forms Designer" screen. Drag the corner of the form to make it nice and big.
3. Drag a button from the left hand "toolbox" menu onto your form. You'll see that it is named "button1". By editing the properties of the button (shown on the bottom right), find the "Text" property and rename it "Start". This will be a "start" and "stop" button (that is, we want you to have one button that switches from "start" to "stop" and back, when clicked. You can change what the button has printed in it by changing its "Text" property and then invoking its ReDraw method)
4. If you click on this button, Visual Studio will create an event hander for "click" events. Code a simple handler that changes the text from Start to Stop and back.
   You'll end up with code that looks like this:

```
private void button1_Click(object sender, System.EventArgs e)
{
      if(button1.Text == "Start")
            button1.Text = "Stop";
      else
            button1.Text = "Start";
      button1.Refresh();
}
```

   We won't be showing you more code. This fragment is to help you get started!

5. From the Debug menu, click "Run", or press alt-F5. Your program will be launched. Test the start/top button functionality functionality.
6. Add a public static Boolean variable to Form1 called "Running". Make it false if Start hasn't been pressed and true if Start has been pressed. (E.g. it switches back and forth). (We make this "static" so that it can be accessed easily within Form1).
7. Now lick back to the "Form1.cs [Design]" screen (you can get back to your code later, don't worry). Drag a Picturebox from the Toolbox onto the bottom of the form. Make it have a 3D border. If you want to be fancy, you can provide some form of background image.
8. Add two more windows "button" controls inside the Picturebox; any size you like should be fine. Change the BackColor color of one of them to green (you can just type "Green"

into the BackColor property box). Make the other one red. Change both Text properties to a null string. Make note of their names (probably Button2 and Button3).

9. Now switch to the Form1.cs code window. In the list of "using" statements at the top, add "Using System.Threading"
10. Now inside the Form1 class, right at the top, define a new public class called "HasAThread". It should have:
    a. A private integer variable "dx" (the x component of the speed), and another called dy.
    b. A variable of type System.Windows.Forms.Button called "mybutton".
    c. A constructor method called "public HasAThread(). It should have three arguments: a System.Windows.Forms.Button used to initialize mybutton, an initial value for dy, and an initial value for dx. Then include these two lines of code at the end of the constructor:

```
Thread myThread = new Thread(new ThreadStart(Run));
myThread.Start();
```

    d. A public void method "Run()". Put this code into Run():

```
while(true)
{
        System.Threading.Thread.Sleep(100);
}
```

    Explanation: each instance of HasAThread will be created with a single thread running inside it. The code executed by this thread is whatever you put into the body of Run(). Initially, we're just looping, having the thread sleep for 100 milliseconds and then wake up, but it doesn't do anything when it wakes up. You'll add code to this loop.

11. Now, find the place where C# says "TODO: Add any constructor code after InitializeComponent " Add code to instantiate your class twice (you need to add some extra variables to Form1 so that it will "remember" these new object instances, or they will be automatically garbage collected when the constructor exits!). Initialize the first object's private variable be the green button you created in step 4, and the second object's private variable be the red one. Have the green object start with dx and dy both equal to 10, and the red one with dx and dy both equal to -10.
12. Now modify the code for "Run". Check the Running flag each time the thread wakes up. If the "Start" button was pressed, add the speed to the Location.X and Location.Y properties of the control, then call the control's Refresh() method. If the control hits a border, make it "bounce off" in the obvious way (e.g. by reversing the **x** or **y** speed, and adjusting the position appropriately). Query the built-in methods of your controls to find the coordinates and size of the border, and the coordinates and size of your "bouncing boxes".

    Note: modifying a control's position is a little tricky. You need to first create a new "point" representing the new position, and then set the Location. For example, if B is a button, you could use code like this:

```
using System.Drawing; // with the other “using” stmts
. . .
Point P = new Point(B.Location.X+dx, BV.Location.Y+dy);
B.Location = P;
B.Refresh();
```

The reason for this code is that the properties of a Location are read-only. Thus if you were to try a short cut and just write:

```
B.Location.X += dx;
B.Location.Y += dy;
```

C# will complain that you can’t change the X and Y properties. But you can change the entire Location object, and this is what our code sample does.

13. Modify your program so that the controls have random initial start positions within your picturebox, and random initial speeds in the range between -25 and 25.

Hand in:

1. A printout of your program, with your name and netid at the top.
2. A 2 or 3 line explanation of what happens when your controls happen to overlap one-another.
3. A paragraph or two explaining how you could extend your program to make the boxes bounce off one-another. However, don’t do this yet. We’ll worry about being fancier next week!