C# Programming

The Timer Event
Action Code

• Action code is my term for any code that does something, causes something to happen, and has action to it

• Examples of action code:
  – Making an object appear
  – Making an object move
  – Any event
Where to place Action Code

• Action code usually requires interaction, so requires being placed where there is interaction, i.e.
  – Mouse events
    • Mouse movement
    • Mouse button presses
  – Keyboard events
    • Keypresses
Where to place Action Code (cont)

• Action code can also be placed in a collision event
• A collision is when two objects/sprites come into contact with each other
Where to place Action Code (cont)

• Not all action code requires interaction. For this type of event, the code needs to occur based on timing; therefore must be placed in a timer event
What does the timer event do?

• The C# timer event keeps track of time just like a clock would. It is basically an accessible clock.

• The C# timer counts in milliseconds, thousandths of a second. This allows for great detail.
  – Remember to do your math if you are thinking in seconds or minutes.
The First Step in using the Timer

- The first step in using the Timer is to realize that it is a tool, so drag it onto your form.
Setting the time

- You control the amount of time in the timer’s preferences. Remember it counts in milliseconds!
What the timer does automatically

• The timer counts
• When the timer counts down the amount of time set in the preferences, it executes whatever code is in it, then it automatically restarts and counts down again.
A bad idea

• It is a bad idea to allow the timer to run amuck.
• You should have it disabled by default, only to start when the program is triggered to start.
• Again, once it has counted down, you should disable it so that it cannot restart automatically. (unless this is what you want)
What can the timer do?

• Once the timer has counted down the time set in its preferences, it will execute any code that is written in the event.

• Typical events are:
  – Making other objects visible or invisible (display)
  – Playing a sound
  – Start or end other events
private void timer1_Tick(object sender, EventArgs e)
{
    //events occur when time runs out
    message1.Visible = true;
    Feedback.Text = "Incorrect";
    CheckAnswer.Visible = false;
    //timer is disabled until next question is started
    timer1.Enabled = false;
}

Example code inside the timer
A top down approach

• Remember that even in object oriented programming, the events within an object or a method are executed in order from top to bottom.
private void timer1_Tick(object sender, EventArgs e)
{
    //events occur when time runs out
    message1.Visible = true;
    message1.Visible = false;
    message1.Visible = true;
    message1.Visible = false;
}

In the previous code, the message would flash on, off, on, and off
Other examples

```csharp
{  
12: Timer myTimer = new Timer();  
13: myTimer.Elapsed += new ElapsedEventHandler( DisplayTimeEvent );  
14: myTimer.Interval = 1000;  
15: myTimer.Start();  
16:  
17: while ( Console.Read() != 'q' )  
18: {  
19: ; // do nothing...  
20: }  
21: }  
22:  
23: public static void DisplayTimeEvent( object source, ElapsedEventArgs e )  
24: {  
25: Console.Write("\r\{0}\r", DateTime.Now);  
26: }
```
Basic functions

**Timer.AutoReset** MSDN: this indicates "whether the Timer should raise the Elapsed event each time the specified interval elapses or only after the first time it elapses." That means if you want a recurring timer, leave this as true.

**Timer.Enabled** "Whether the Timer should raise the Elapsed event." You must set this to true if you want your timer to do anything.

**Timer.Interval** This indicates "the time, in milliseconds, between raisings of the Elapsed event. The default is 100 milliseconds." You will almost certainly want to make this interval longer than the default. For example, for 30 seconds, use 30000 as the Interval.

**Dispose Disposed** Timers allocate system resources, so if you are creating a lot of them, make sure to Dispose them. This gets complicated fast. I suggest just using a single static timer.

**Timer.Start** This does the same thing as setting Enabled to true. I am not certain why we need this duplicate method.

**Timer.Stop** This does the same thing as setting Enabled to false. See the Timer.Start method shown previously.

**Timer.Elapsed Event** This is the event that is invoked each time the Interval of the Timer has passed. You must specify this function in your code. To add the event, you can press tab twice after typing "_timer.Elapsed +=".
Built in method

• The timer has a built in method of DateTime
  – Allows you to display the date and time
  – This can be found on pages 142-143 of the textbook