

Chapter 5B

Visual C# Program: Hyperbolic Ball

In this chapter, you will learn how to use the following Visual C# Application functions to World Class standards:

- **Opening Visual C# Editor**
- **Beginning a New Visual C# Project**
- **Laying Out a User Input Form in Visual C#**
- **Adding Comments in Visual C# to Communicate to Others**
- **Declaring Variables in a Program with the Int Statement**
- **Setting Variables in a Program**
- **Draw a Blue Ball on the Form**
- **Adding a Timer to the Program**
- **Programming for the Timer**
- **Running the Program**

Open the Visual C# Editor

In this lesson, we will continue to move objects in a Visual C# window so we can entertain or train a person using the application. More specifically, we will begin this project by bouncing a ball up and down in a natural way in the window. We expanded on the idea from an article on a Bouncing Object¹ from the EHow website.

To open a new project, we select File on the Menu Bar and New Project.

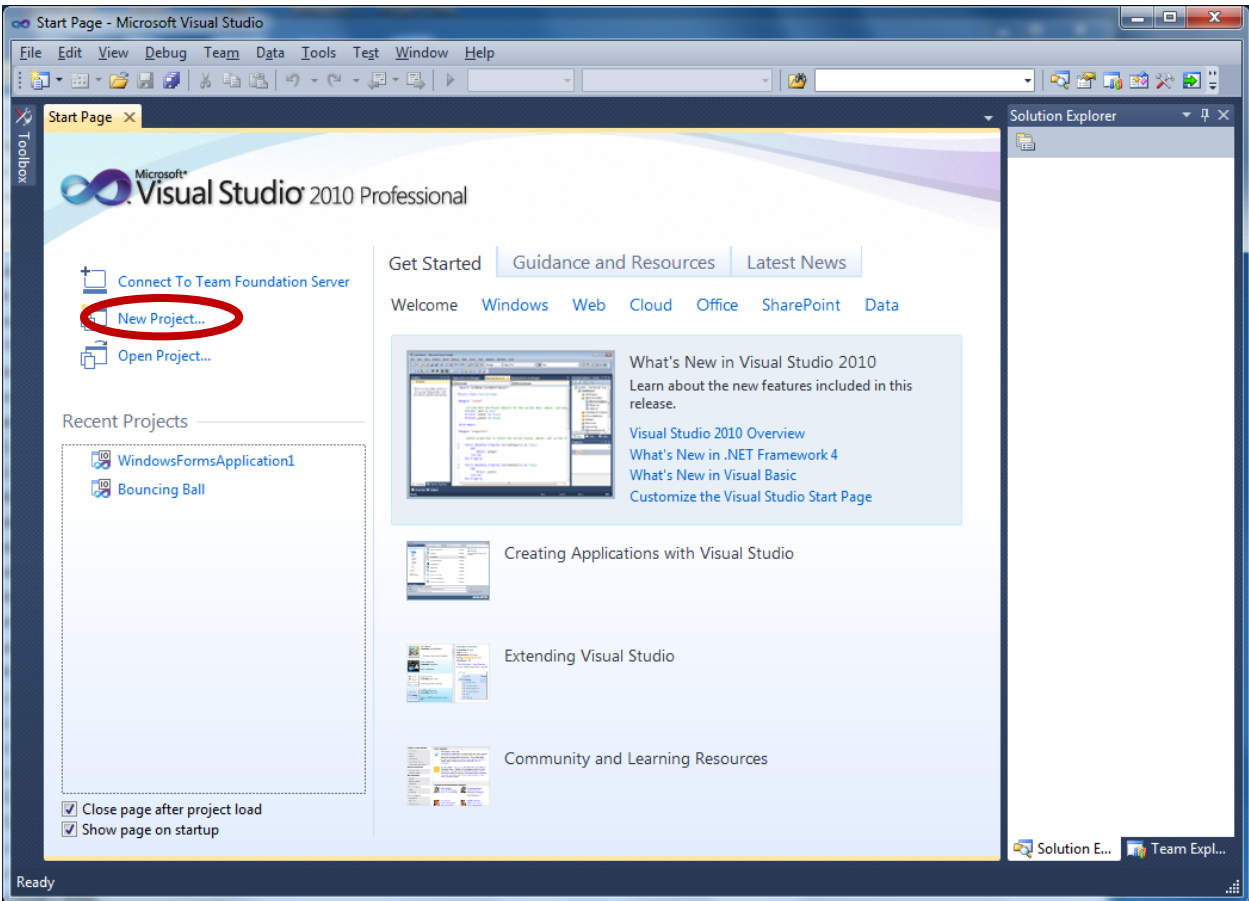


Figure 5.1 – The Start Page

To open a new project, we select New Project on the left side of the Start Page.

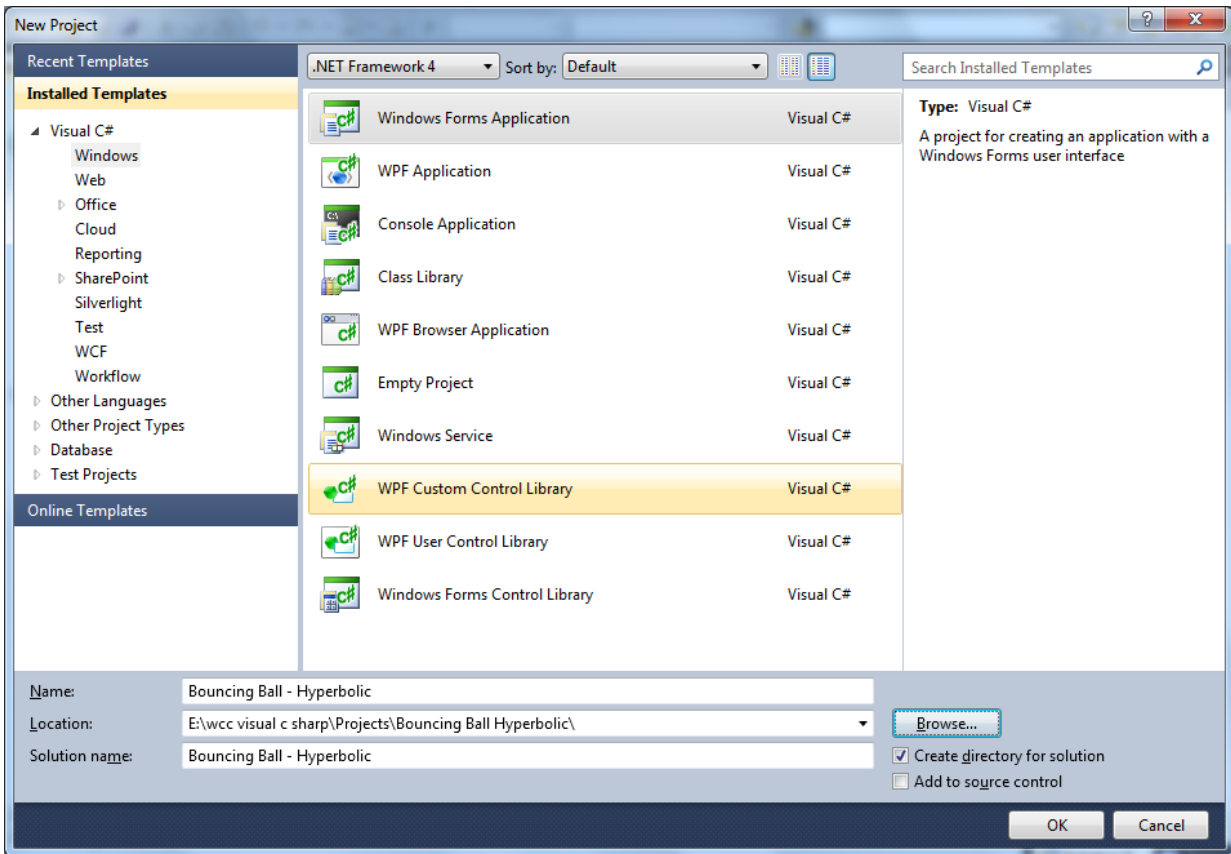


Figure 5.2 – New Project

We start a new Windows Application Project by picking the Windows under Visual C # in the left pan of the New Project window. Then we pick Windows Form Application in the center pane.

At the bottom of the Window, we name the project, Bouncing Ball - Hyperbolic. We make a folder for our projects called Visual C Sharp on the desktop, on our flash drive or in the Documents folder. We make another folder inside the first called Bouncing Ball - Hyperbolic. On the New Project window, we browse to the Bouncing Ball - Hyperbolic location. The solution name is the same as the project name.

Beginning a New Visual C# Application

Remember, that all programming projects begin with one or more sketches. The sketch will show labels, textboxes, and command buttons. In this project, we will name the input form, **Bouncing Ball - Hyperbolic**. In this application, we will just have a form.

We will write code that draws a circle and fill the ball with the same color. We will make the blue ball bounce up and down.

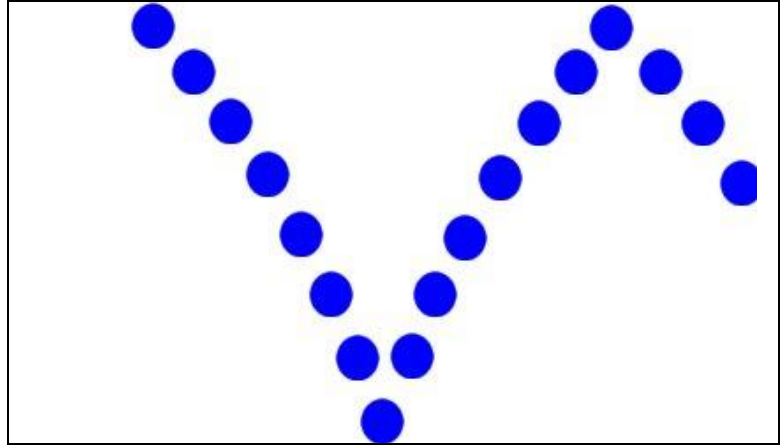


Figure 5.3 – Sketch of the Bouncing Ball Form

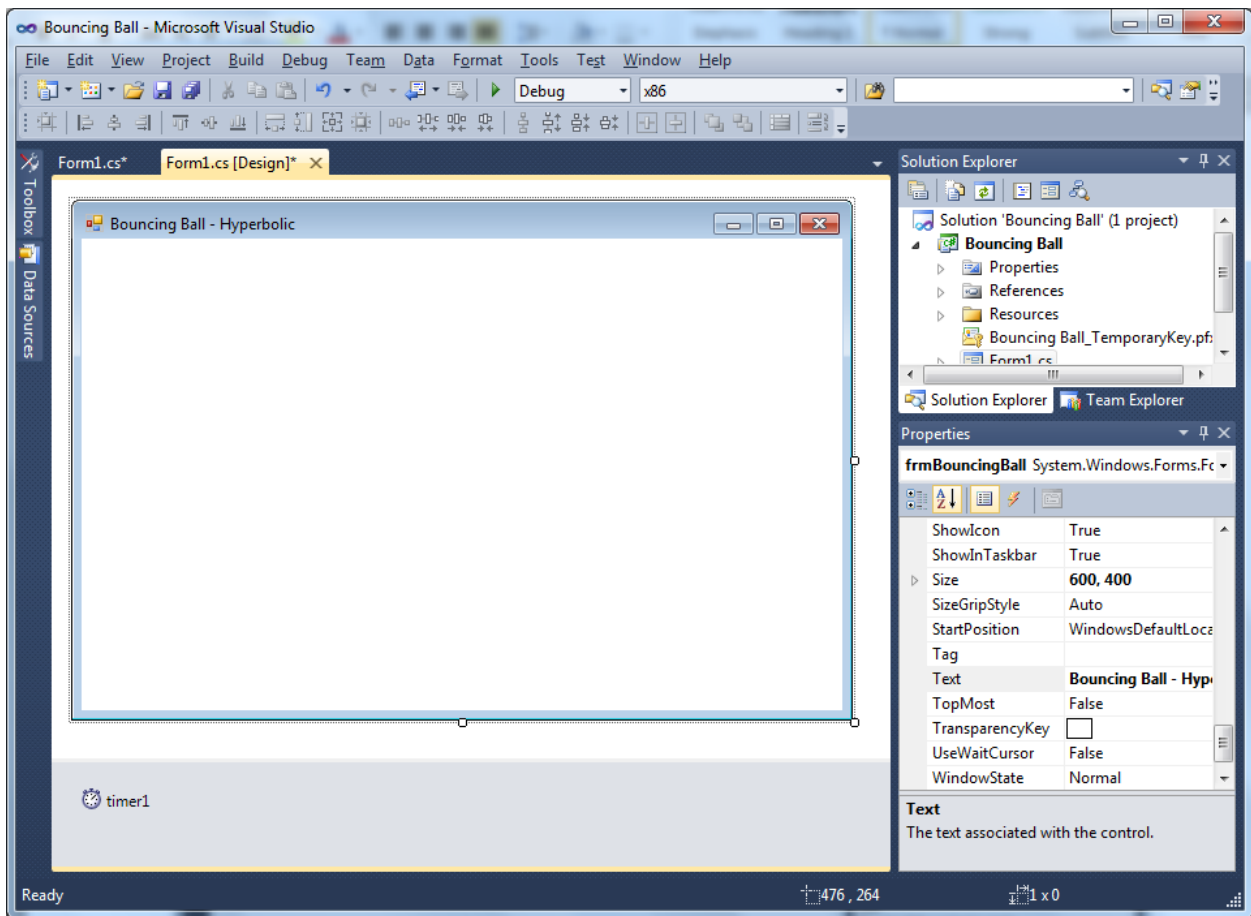


Figure 5.4 – Designing the Bouncing Ball Form in Visual C#

Laying Out a User Input Form in Visual C#

We will change the **Text** in the Properties pane to **Bouncing Ball – Hyperbolic** to agree with the sketch in Figure 5.3. Go ahead and change the form in two other aspects, **BackColor** and **Size**.

Alphabetic	
BackColor	White
Size	600, 400

The first number is the width and the second number is the height. The form will change in shape to the size measurement.

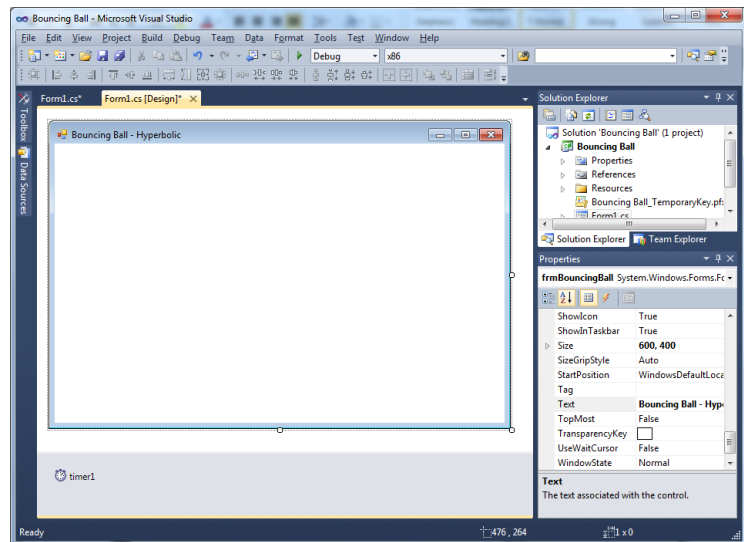


Figure 5.5 – Setting the Form Properties

The background color will change to a white. There are many more attributes in the Properties pane that we will use on future projects.

Adding Comments in Visual C# to Communicate to Others

The comments we placed in the first three lines of the program will inform the individual opening and reading the code of the ownership. This is for those user that may run the application without checking the label on the bottom of the form with the copyright information. It is a great tool to alert the client to the rules of the program and tell them what the application will do.

To begin the actual coding of the program, double click on the form. At the top of the program and after the line of code with `Namespace Bouncing Ball {`, place the following comments with two slash (`//`) characters. Remember, the two slashes (`//`) will precede a comment and when the code is compiled, comments are ignored.

Type the following line of code:

```
//Bouncing Ball - Hyperbolic  
//This program will open a dialogue box and the ball with bounce up and down  
//in a natural curve until the user presses the escape button
```

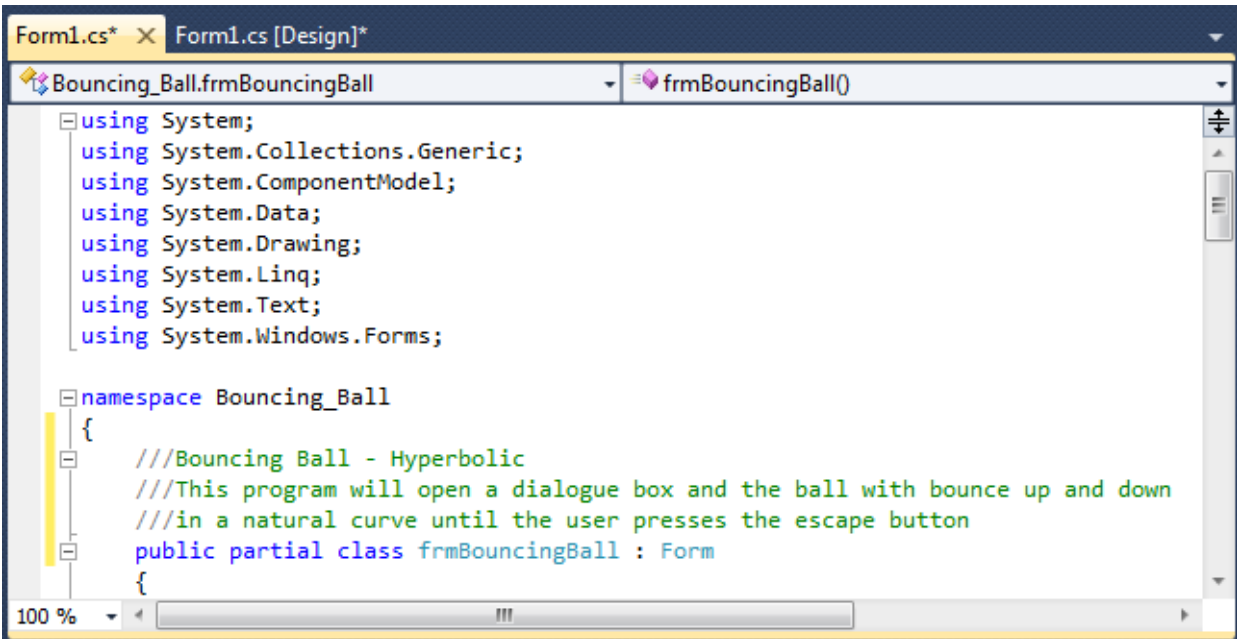


Figure 5.6 – Adding an Introducing Statement

Declaring Variables in a Program with the Int Statement

When we are going to use a number, text string or object that may change throughout the life of the code, we create a variable to hold the value of that changing entity. In Visual C#, the integer statement is one of the ways to declare a variable at the procedure level.

In this program, we will create data from mathematical computations. We will place the values in integer variables called x, y, dx (change of x), c (counter) and bc (bounce counter). These variables will hold whole numbers for movement on the form in terms of pixels, so we will declare all five as integers.

Type the following code under the public partial class frmBouncingBall : Form subroutine of the program.

```
public partial class frmBouncingBall : Form
{
    int x;    //x is the position on the x axis from the upper left corner
    int y;    //y is the position on the y axis from the upper left corner
    int dx;   //dx is the change of x axis
    int c;    //x movement in a single cycle
    int bc;   //bounce counter
}
```

The integers x and y represent the actual position of the bouncing ball. The dx is the change in the x position for movement of the timer. We will move to the right one pixel at a time. In our hyperbolic curve, we will use the formula $y = x^{1.1}$. However as x will count from 0 to 600 as

the ball bounces to the right, we introduce the variable c to cycle the x coordinate in one frequency. When we use the $y = c^{1.1}$, the y coordinate will increase and decrease because of the value of c . To make that happen, we use the integer bc , which counts up when the ball is falling on the form and counts down (negative) when the ball is rising.

We created a frequency chart that shows the values of x , c , y and bc in 20 pixels of movement to the right.

x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
c	1	2	4	7	11	16	22	29	37	46	56	46	37	29	22	16	11	7	4	2
y	1	2	8	18	36	64	103	156	225	311	419	311	225	156	103	64	36	18	8	2
bc	1	2	3	4	5	6	7	8	9	10	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

Therefore, we can see the purpose of each variable we choose in the program.

```

namespace Bouncing_Ball
{
    ///Bouncing Ball - Hyperbolic
    ///This program will open a dialogue box and the ball with bounce up and down
    ///in a natural curve until the user presses the escape button
    public partial class frmBouncingBall : Form
    {
        int x;        ///x is the position on the x axis from the upper left corner
        int y;        ///y is the position on the y axis from the upper left corner
        int dx;       ///dx is the change of x axis
        int c;        ///x movement in a single cycle
        int bc;       ///bounce counter
    }
}

```

Figure 5.7 – Declaring Variables with Int Statements

The variable names should relate to what data they hold. We should not have spaces in the name. Some programmers use the underscore character ($_$) to separate words in phrases. This is acceptable, but a double underscore ($__$) can cause errors if we do not detect the repeated character. We could call the variables $x_coordinate$ and $y_coordinate$.

Setting Variables in a Program

Next, we will set the variables using the equal function. We will set the numbers in the two textboxes to their variable and we compute resistance by division and the power by using multiplication.

Type this code under the private void `frmBouncingBall_Load(object sender, EventArgs e)` subroutine of the program.

```

private void frmBouncingBall_Load(object sender, EventArgs e)
{
    ///Loads the ball on the upper top part of the screen
    x = 0; ///starting point for x
    y = 0; ///starting point for y
    dx = 1; ///change in x position for each moment of time
    c = 1; ///c is the position in the x axis in one cycle
    bc = 0; ///the counter plus or minus as the ball goes up or down
}

```

The variable called speed is the change of vertical position in pixels, so we will start the movement of the ball in one pixel increments. The x and y coordinates are the position of the ball. The form is measured from 0,0 in the upper left corner to width and height of the form in the lower right corner, which is 600 pixels by 400 pixels for our project.

We start the ball at 0,0. We keep the change of x (dx) the same throughout the entire program at 1. The integer starts out at 1 and our counter starts at 0.

The screenshot shows the Visual Studio IDE with the following code in the frmBouncingBall_Load method:

```

namespace Bouncing_Ball
{
    ///Bouncing Ball - Hyperbolic
    ///This program will open a dialogue box and the ball with bounce up and down
    ///in a natural curve until the user presses the escape button
    public partial class frmBouncingBall : Form
    {
        int x;        ///x is the position on the x axis from the upper left corner
        int y;        ///y is the position on the y axis from the upper left corner
        int dx;       ///dx is the change of x axis
        int c;        ///x movement in a single cycle
        int bc;       ///bounce counter

        public frmBouncingBall()
        {
            InitializeComponent();
        }

        private void frmBouncingBall_Load(object sender, EventArgs e)
        {
            ///Loads the ball on the upper top part of the screen
            x = 0; ///starting point for x
            y = 0; ///starting point for y
            dx = 1; ///change in x position for each moment of time
            c = 1; ///c is the position in the x axis in one cycle
            bc = 0; ///the counter plus or minus as the ball goes up or down
        }
    }
}

```

Figure 5.8 – Setting the Variables in the C# Code

Draw a Blue Ball on the Form

Before we write the code for the ball, click on the Form1.cs tab to return to the form. Then we should click on the events icon (the picture of a lightning bolt).

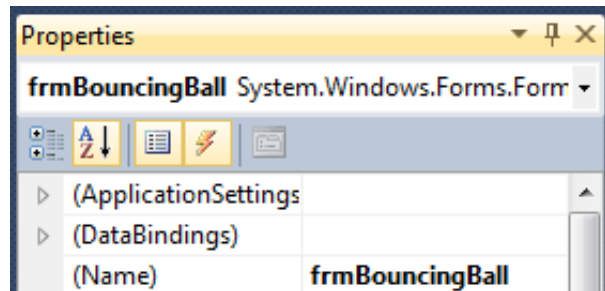


Figure 5A.9 – Event Icon

Then we want to double click on Paint in the Appearance section of the Properties window. The following will appear in our code.

```
private void frmBouncingBall_Paint(object sender, PaintEventArgs e)
{
}
}
```

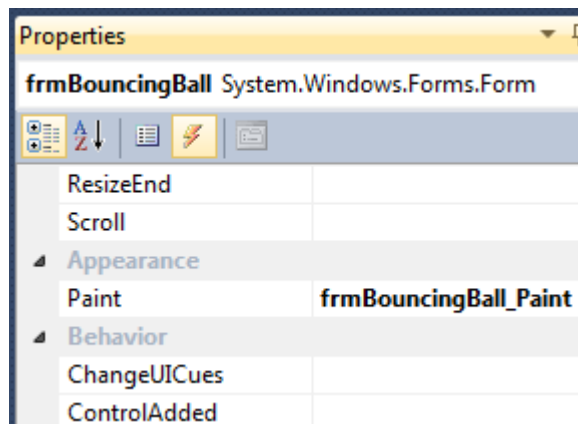
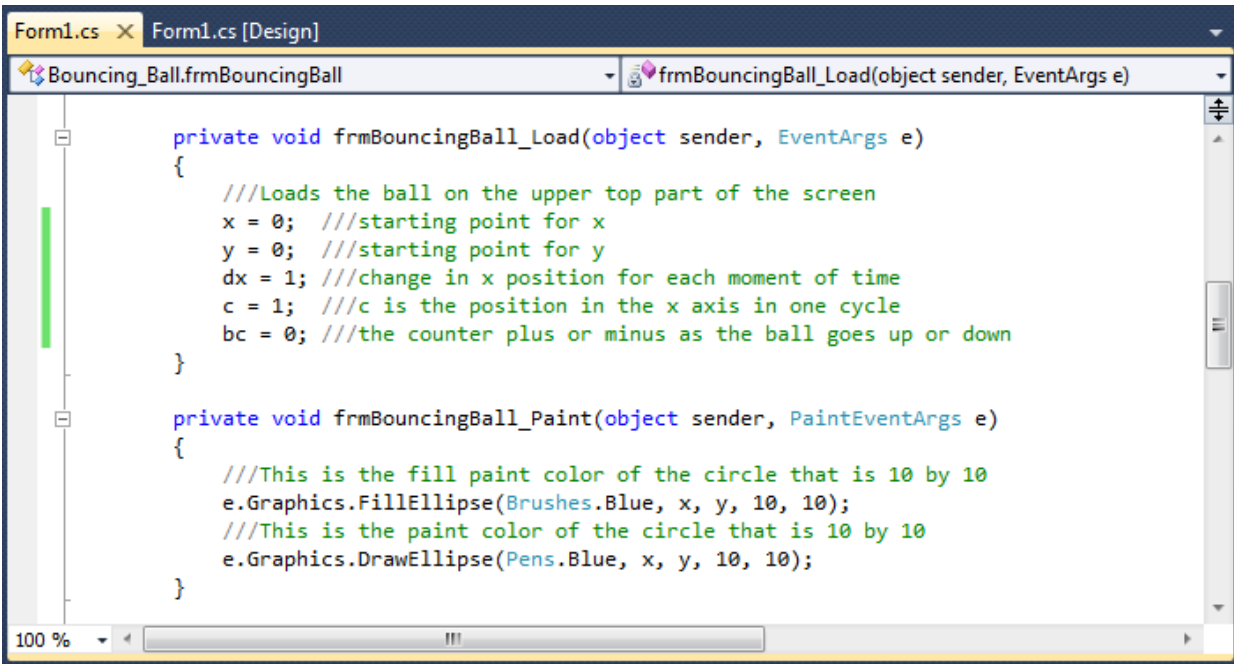


Figure 5A.10 – Paint Appearance

We could add a picture of a ball, but we will draw a blue 10 by 10 pixel ball and fill it with the same blue color.

Type the following code in the private void frmBouncingBall_Paint(object sender, PaintEventArgs e) subroutine.

```
private void frmBouncingBall_Paint(object sender, PaintEventArgs e)
{
    ///This is the fill paint color of the circle that is 10 by 10
    e.Graphics.FillEllipse(Brushes.Blue, x, y, 10, 10);
    ///This is the paint color of the circle that is 10 by 10
    e.Graphics.DrawEllipse(Pens.Blue, x, y, 10, 10);
}
}
```



```
private void frmBouncingBall_Load(object sender, EventArgs e)
{
    ///Loads the ball on the upper top part of the screen
    x = 0; ///starting point for x
    y = 0; ///starting point for y
    dx = 1; ///change in x position for each moment of time
    c = 1; ///c is the position in the x axis in one cycle
    bc = 0; ///the counter plus or minus as the ball goes up or down
}

private void frmBouncingBall_Paint(object sender, PaintEventArgs e)
{
    ///This is the fill paint color of the circle that is 10 by 10
    e.Graphics.FillEllipse(Brushes.Blue, x, y, 10, 10);
    ///This is the paint color of the circle that is 10 by 10
    e.Graphics.DrawEllipse(Pens.Blue, x, y, 10, 10);
}
```

Figure 5.11 – The Code to Draw a Circle and Fill It

Adding a Timer to the Program

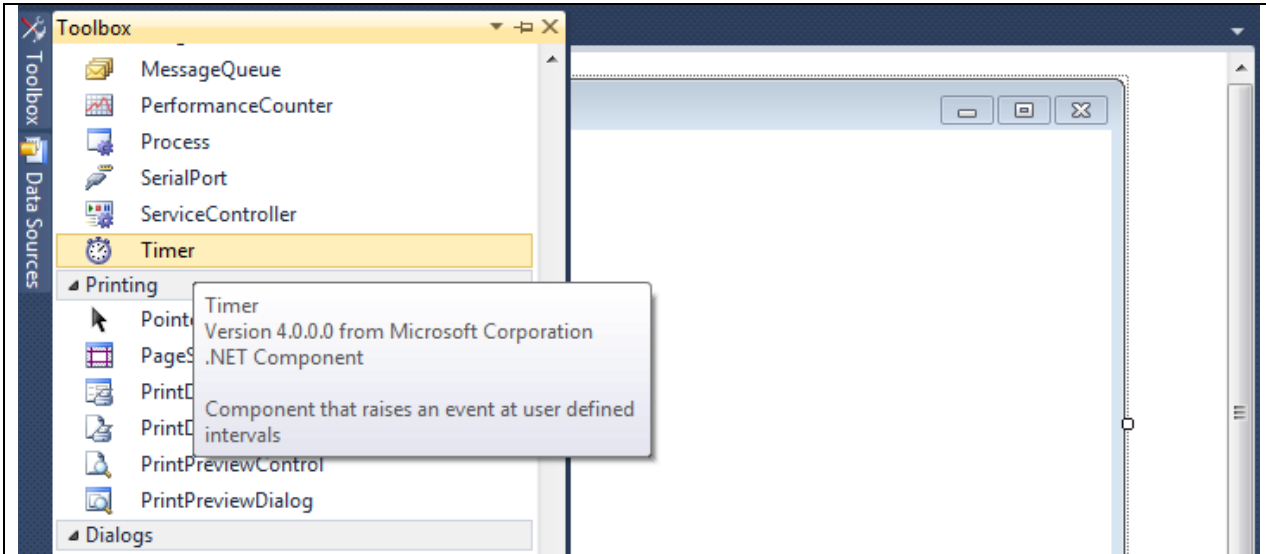


Figure 5.12 – Adding a Timer

We will next add a Timer to the Bouncing Ball application by selecting Timer from the Toolbox and placing it on the form.

We Double click on the timer and name the object timer1. We will set Enabled and GenerateMember as True and the Interval to 1 millisecond. We will set the Modifiers to Private.

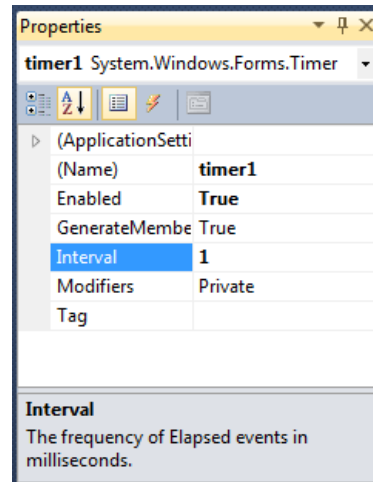


Figure 5.13 – Setting the Timer’s Properties

Programming for the Timer

We will double click on the timer on the form and add code to change the y coordinate to y + speed. Remember speed is set to 1.

We then add an if – then – else statement. We check if y is less than y and if it is true, we change the direction of speed and the ball will change direction. We also check if the y + 10 (10 is the size of the ball) is greater than the form height and if it is, we change the direction of speed and the ball will change direction.

Type the following code in the private void timer1_Tick(object sender, EventArgs e) subroutine.

```
private void timer1_Tick(object sender, EventArgs e)
{
    x = x + dx; // adds 1 to the x axis each timer movement
    c = c + bc; // we add pos or neg bc to the cycle counter
    if (x < 0)
    {
        dx = -dx; // if x is less than 0 then we change direction
    }
    else if (x + 10 > this.ClientSize.Width)
    {
        dx = -dx; // if x + 10, the radius of the circle is greater than the form height then we change direction
    }
    if (c < 1)
    {
        c = 1; // reset c as 1 when the cycle counter is less than 1
        bc = 0; // reset bc as 0 when the cycle counter is less than 1
    }
}
```

```

}
y = Convert.ToInt32(Math.Floor(Math.Pow(c, 1.1))); // c to 1.1 power
bc = bc + 1; // add one to the counter bc
if (y + 10 > this.ClientSize.Height)
{
    bc = -bc; // if y + 10, the radius of the circle is greater than the form height then we change direction
}
this.Invalidate();
}

```

The screenshot shows a Visual Studio window with the following code:

```

Form1.cs* x Form1.cs [Design]*
Bouncing_Ball.frmBouncingBall timer1_Tick(object sender, EventArgs e)
private void frmBouncingBall_Paint(object sender, PaintEventArgs e)
{
    ///This is the fill paint color of the circle that is 10 by 10
    e.Graphics.FillEllipse(Brushes.Blue, x, y, 10, 10);
    ///This is the paint color of the circle that is 10 by 10
    e.Graphics.DrawEllipse(Pens.Blue, x, y, 10, 10);
}

private void timer1_Tick(object sender, EventArgs e)
{
    x = x + dx; /// adds 1 to the x axis each timer movement
    c = c + bc; /// we add pos or neg bc to the cycle counter
    if (x < 0)
    {
        dx = -dx; /// if x is less than 0 then we change direction
    }
    else if (x + 10 > this.ClientSize.Width)
    {
        dx = -dx; /// if x + 10, the radius of the circle is greater than the form height then we change direction
    }
    if (c < 1)
    {
        c = 1; /// reset c as 1 when the cycle counter is less than 1
        bc = 0; /// reset bc as 0 when the cycle counter is less than 1
    }
    y = Convert.ToInt32(Math.Floor(Math.Pow(c, 1.1))); /// c to 1.1 power
    bc = bc + 1; /// add one to the counter bc
    if (y + 10 > this.ClientSize.Height)
    {
        bc = -bc; /// if y + 10, the radius of the circle is greater than the form height then we change direction
    }
    this.Invalidate();
}
}

```

Figure 5.14 – Programming for the Timer

Running the Program

After noting that the program is saved, press the F5 to run the Bouncing Ball application. The Bouncing Ball window will appear on the graphical display as shown in Figure 5.13. The ball will be moving up to the top and then fall back down.

After viewing the bouncing ball, click on the blue X in the upper right corner to close the application.

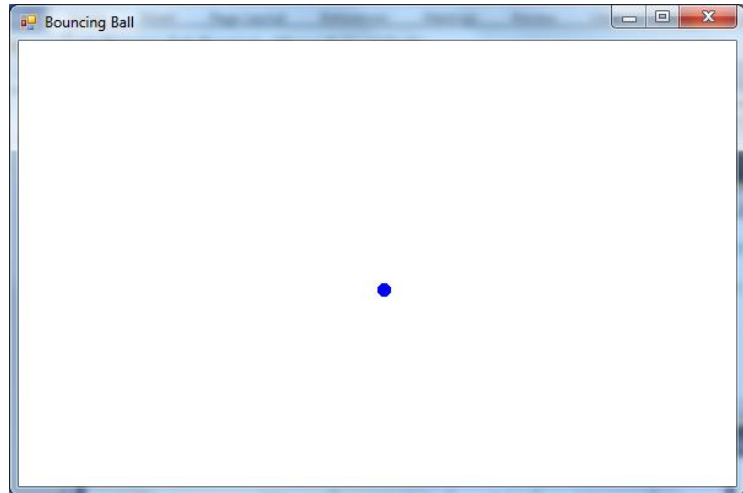


Figure 5.15 – Launching the Program

If our program does not function correctly, go back to the code and check the syntax against the program shown in previous sections. Repeat any processes to check or Beta test the program. When the program is working perfectly, save and close the project.

Here is the entire code to check your syntax.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Bouncing_Ball
{
    //Bouncing Ball - Hyperbolic
    //This program will open a dialogue box and the ball with bounce up and down
    //in a natural curve until the user presses the escape button
    public partial class frmBouncingBall : Form
    {
        int x;    //x is the position on the x axis from the upper left corner
        int y;    //y is the position on the y axis from the upper left corner
        int dx;   //dx is the change of x axis
        int c;    //x movement in a single cycle
        int bc;  //bounce counter
    }
}
```

```

public frmBouncingBall()
{
    InitializeComponent();
}

private void frmBouncingBall_Load(object sender, EventArgs e)
{
    //Loads the ball on the upper top part of the screen
    x = 0; //starting point for x
    y = 0; //starting point for y
    dx = 1; //change in x position for each moment of time
    c = 1; //c is the position in the x axis in one cycle
    bc = 0; //the counter plus or minus as the ball goes up or down
}

private void frmBouncingBall_Paint(object sender, PaintEventArgs e)
{
    //This is the fill paint color of the circle that is 10 by 10
    e.Graphics.FillEllipse(Brushes.Blue, x, y, 10, 10);
    //This is the paint color of the circle that is 10 by 10
    e.Graphics.DrawEllipse(Pens.Blue, x, y, 10, 10);
}

private void timer1_Tick(object sender, EventArgs e)
{
    x = x + dx; // adds 1 to the x axis each timer movement
    c = c + bc; // we add pos or neg bc to the cycle counter
    if (x < 0)
    {
        dx = -dx; // if x is less than 0 then we change direction
    }
    else if (x + 10 > this.ClientSize.Width)
    {
        dx = -dx; // if x + 10, the radius of the circle is greater than the form height then we change direction
    }
    if (c < 1)
    {
        c = 1; // reset c as 1 when the cycle counter is less than 1
        bc = 0; // reset bc as 0 when the cycle counter is less than 1
    }
    y = Convert.ToInt32(Math.Floor(Math.Pow(c, 1.1))); /// c to 1.1 power
    bc = bc + 1; // add one to the counter bc
    if (y + 10 > this.ClientSize.Height)
    {
        bc = -bc; // if y + 10, the radius of the circle is greater than the form height then we change direction
    }
    this.Invalidate();
}

```

```
}  
}  
}
```

There are many variations of this Visual C# Application we can practice and obtain information from a personal computer. While we are practicing with forms, we can learn how to use variables, strings and comments. These are skills that we want to commit to memory.

*** World Class CAD Challenge 90-7 * - Write a Visual C# Application that displays a single form and when executed, the program will show a hyperbolic bouncing ball from the computer using mathematical computations.**

Continue this drill four times using some other form designs, each time completing the Visual C# Project in less than 1 hour to maintain your World Class ranking.

ⁱ How to Make an Object Bounce in C#, Jaime Avelar, eHow Contributor, 2012, Demand Media, October 18, 2012 < http://www.ehow.com/how_12030598_make-object-bounce-c.html>