

Chapter 6B

Visual C# Program: Simple Game 2

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#**
- **Inserting a Label into a Form**
- **Adding a PictureBox 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**
- **Adding a Timer to the Program**
- **Programming for the Timer**
- **Running the Program**

Open the Visual C# Editor

In this lesson, we will write our second game, which allows two players to try to get the ball by their paddle. We will take much of the code from the previous lesson, but we will practice writing new condition statements for the extra objects. At the beginning of the game, the player will launch a ball from their paddle and the other contestant will try to keep the ball from passing the imaginary line just equal to their paddle. If it does, the sending player will gain a point. When one player gets a total of five points, they win and the game over text will appear and the game will conclude.

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

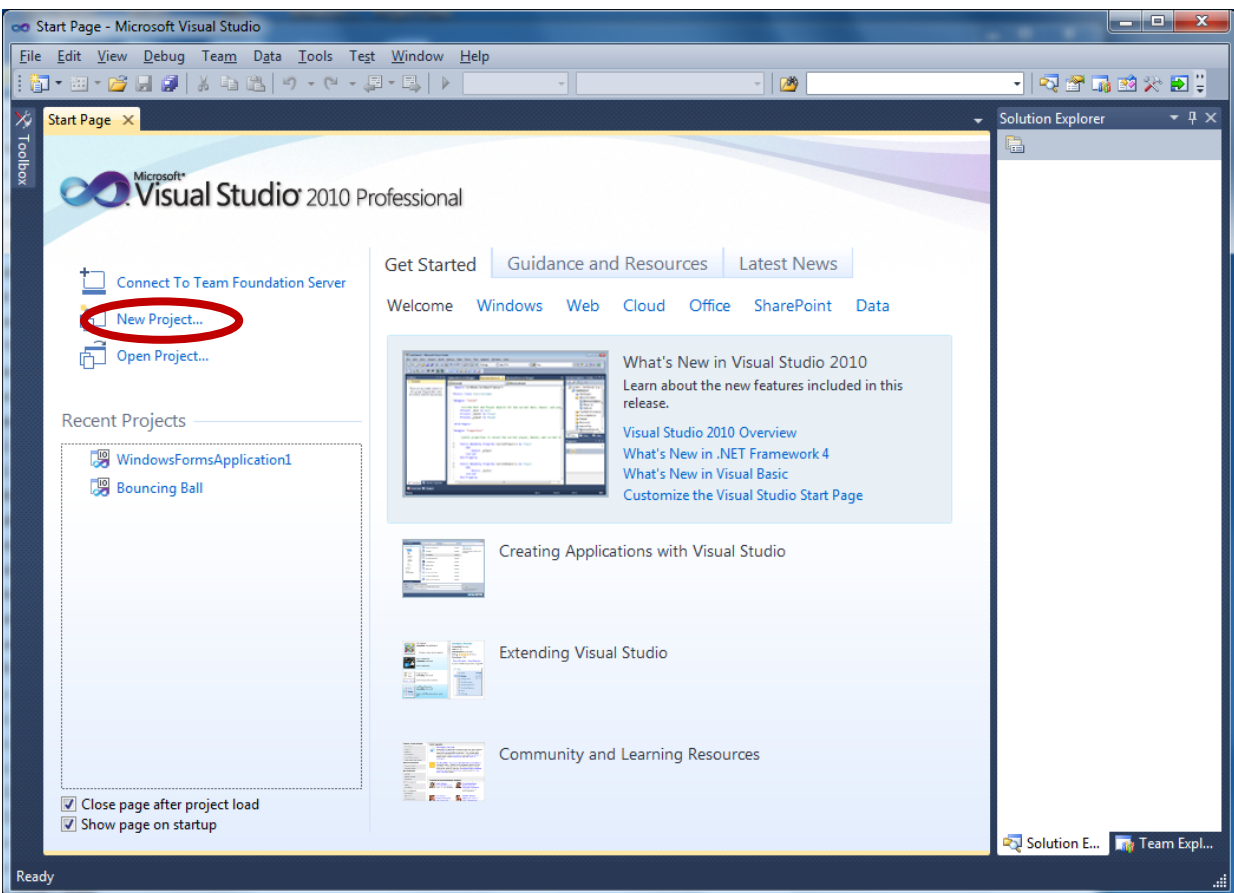


Figure 6B.1 – The Start Page

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

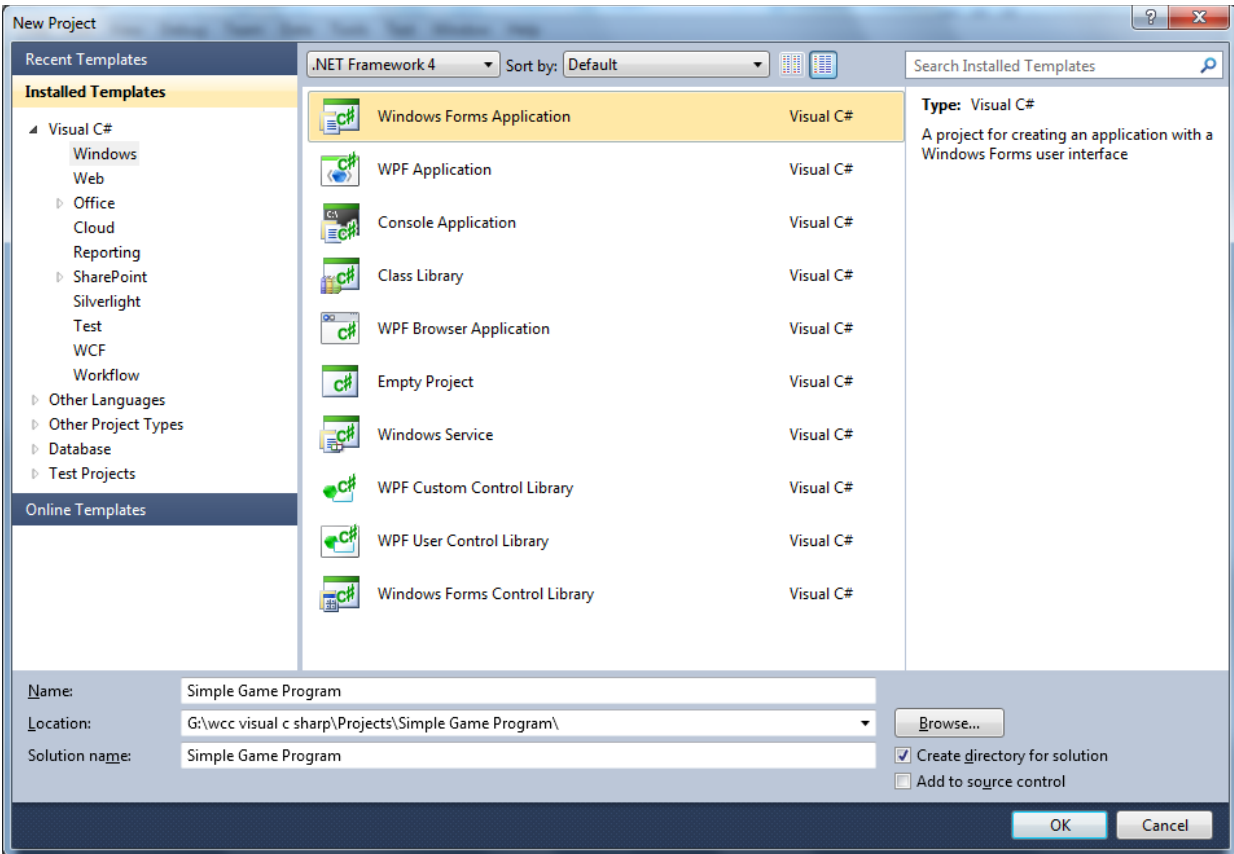


Figure 6B.2 – New Project

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, Simple Game Program 2. 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 Simple Game Program 2. On the New Project window, we browse to the Simple Game Program 2 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 and pictures. In this project, we will name the input, **Simple Game**. In this application, we will just have a form.

We will write code that moves a maroon ball from the middle right starting at the 500,200 location. We will displace the ball 1 pixel to the right and 1 pixel up for each tick of time on the timer. We will have a vertical image for a paddle to stop the ball from going by us. We will add a label for keeping the score. We will gain a point for every time the ball goes by our opponent's paddle. When that occurs, the player that won the point will serve another ball. The name of the game and the title of the score boxes are in three labels.

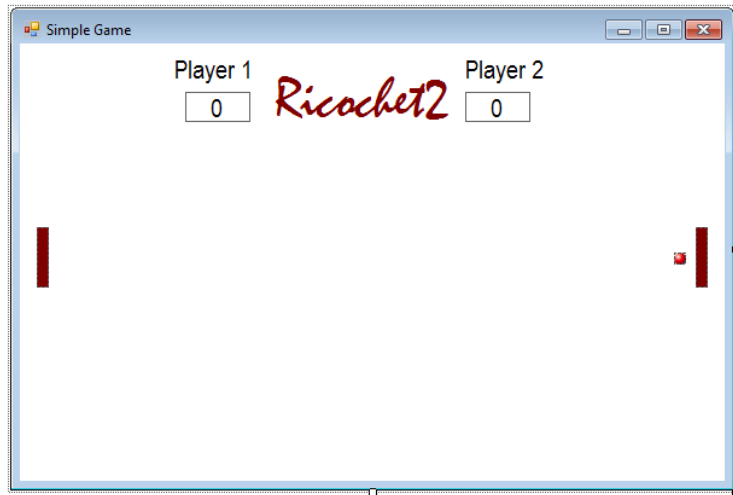


Figure 6B.3 – Sketch of the Simple Game Form

When one player gets a total of five points, they will the win and the game over text will appear on the bottom of the window. The game will conclude.

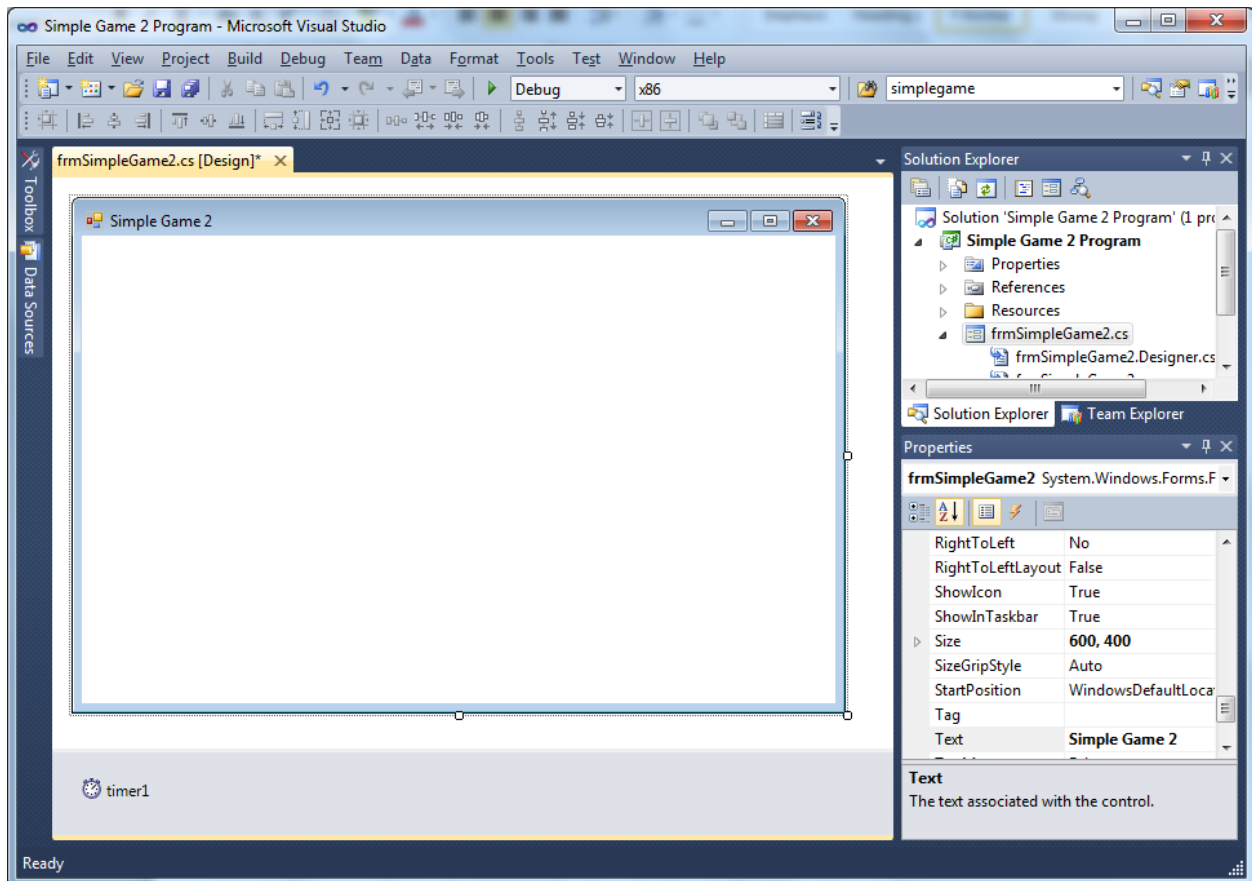


Figure 6B.4 – Designing the Simple Game Form in Visual C#

Laying Out a User Input Form in Visual C#

We will change the **Text** in the Properties pane to Simple Game 2 to agree with the sketch in Figure 6B.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.

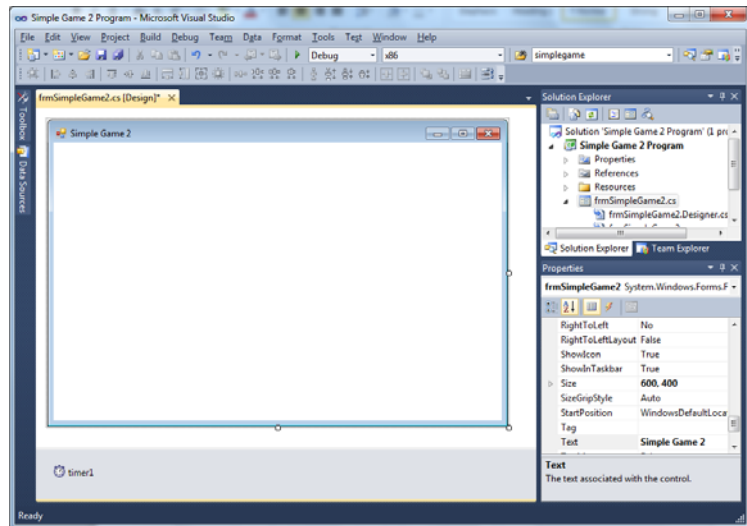


Figure 6B.5 – Setting the Form Properties

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

In the Solution Explorer pane, right click on Form1.cs and rename it to frmSimpleGame2.cs.

Inserting a Label into a Form

A good form is easy to figure out by the user, so when we are attempting to provide information on the window that will run in Windows; we add labels to textboxes to explain our intent. Press the Label (A) button on the Control Toolbar to add a label. To size the label area, click on the upper left area of the form and hold down on the left mouse button, draw the dotted label box.

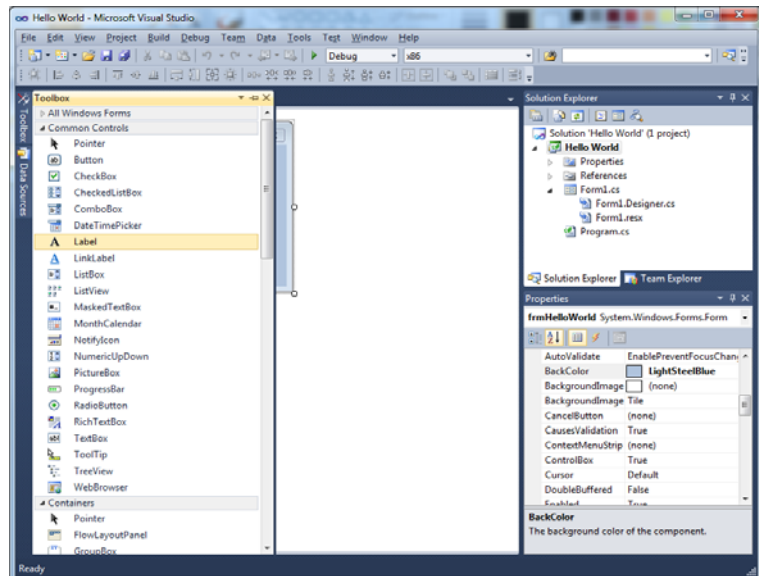


Figure 6B.6 – Placing a Label on the Form

We will name the Label using a common Visual C# naming convention where the programming object is a three letter prefix followed by the name or phrase of the tool. For our first label, the name is **lblPlayer1Label**.

Alphabetic	
(Name)	lblPlayer1Label
BackColor	White
Font	Arial Narrow, 16 pt
Text	Player 1

On the sketch, the label’s caption is “**Player 1**” The font on the sketch is 16 point, Arial Narrow. When highlighting the row for Font, a small command button with three small dots appears to the right of the default font name of Microsoft San Serif. Click on the three dotted button to open the Visual C# Font window.

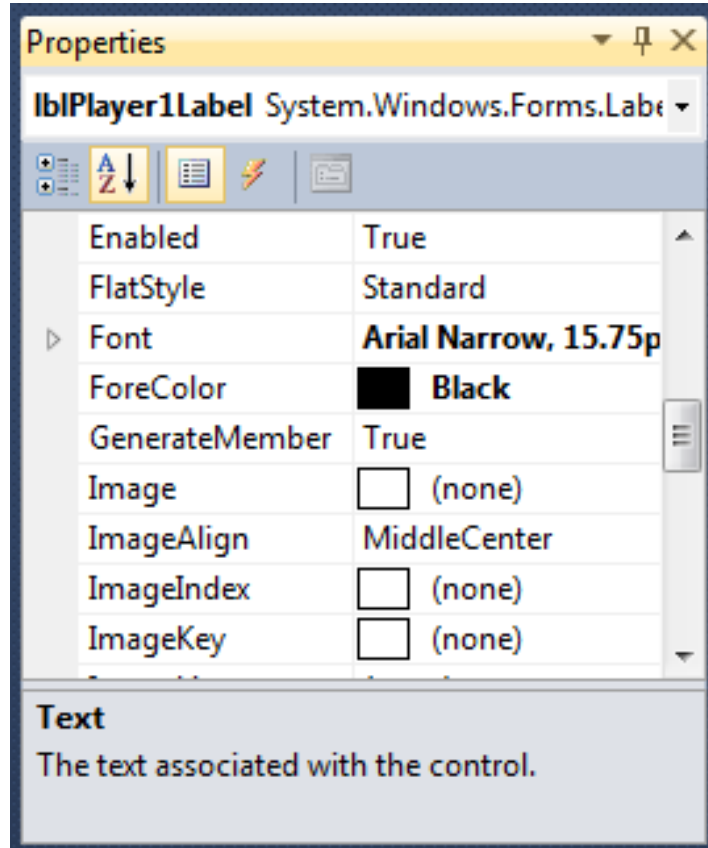


Figure 6B.7 – Changing the Font Property

We will select the Arial font, Regular font style and 16 size for this project to agree with the initial sketch if the user input form. When we adjust the attributes for the label, these changes do not alter globally for the other objects on the form. If we wish to underline the text or phrase in the label, add a check to the Underline checkbox in the Effects section of the Font window. When we finish making changes to the font property, select the OK command button to return to the work area.

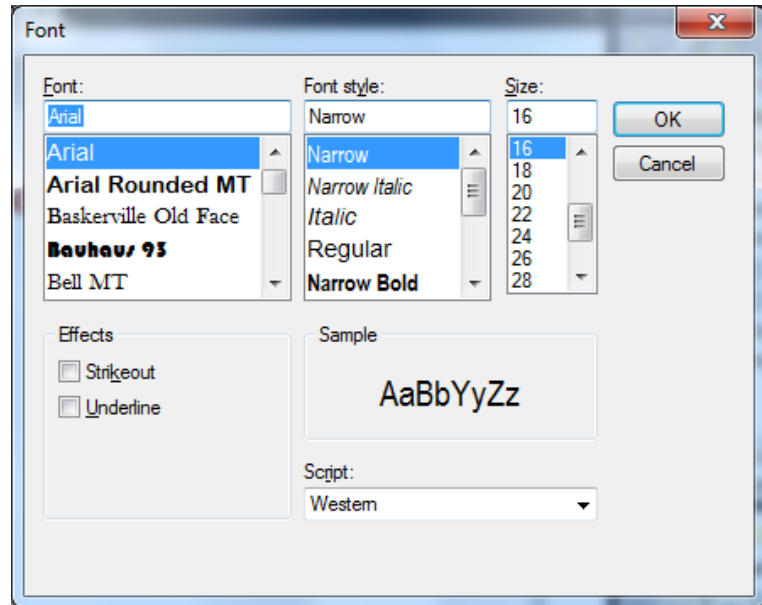


Figure 6B.8 – The Font Window in Visual C#

When the first label is done, the background color of the label matches the background color of the form. In many cases that effect is visually pleasing to the eye, versus introducing another color. Both color and shape will direct the user in completing the form along with the explanation we place on the window to guide the designer in using the automated programs. Use colors and shape strategically to communicate well.

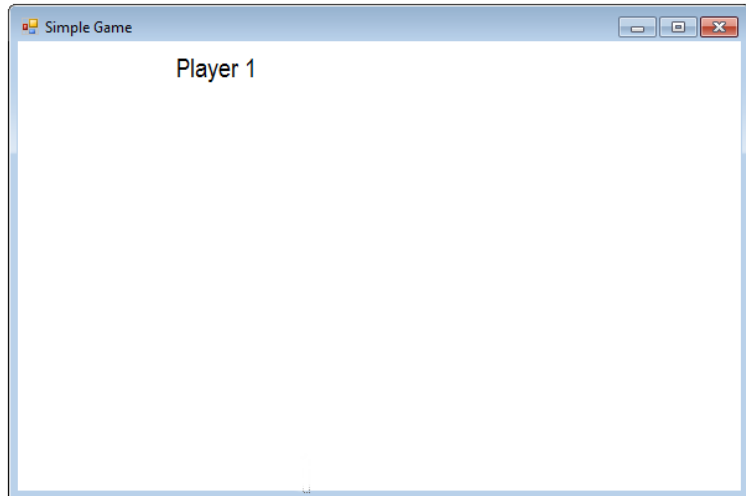


Figure 6B.9 – The Finished Label on the Form

We will call the label for the second player `lblPlayer2Label` and we will make the text state “Player 2”.

Alphabetic	
(Name)	<code>lblPlayer2Label</code>
BackColor	White
Font	Arial Narrow, 16 pt
Text	Player 2

The next label is for the name Ricochet2 which is `lblName`.

Alphabetic	
(Name)	<code>lblName</code>
BackColor	White
Font	Mistral, 36 pt
ForeColor	Maroon
Text	Ricochet2

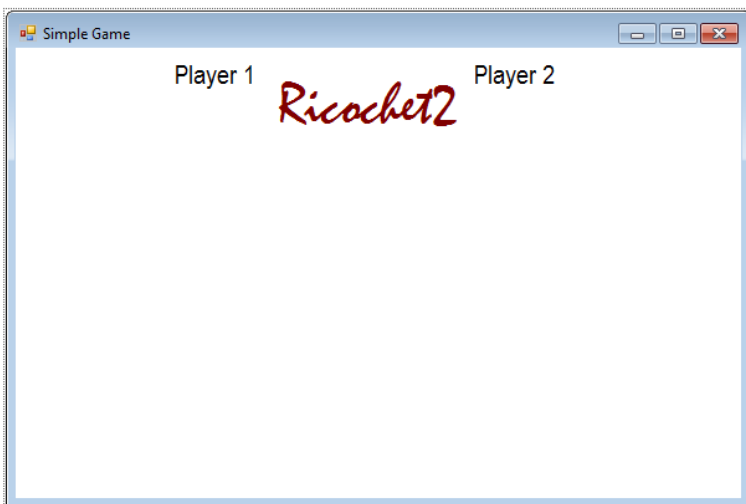


Figure 6B.10 – The Finished Labels on the Form

We will add two labels that will hold the scores for player 1 and 2. We will position these labels under their identifiers and they will hold their initial number, 0 for their score. We will place the last label in a position below the paddle and in the center.

Alphabetic	
(Name)	lblPlayer1
Border	Fixed Single
BackColor	White
Font	Arial Narrow, 16 pt
Size	54,25
Text	0

Alphabetic	
(Name)	lblPlayer2
Border	Fixed Single
BackColor	White
Font	Arial Narrow, 16 pt
Size	54,25
Text	0

Alphabetic	
(Name)	lblGameOver
BackColor	White
Font	Arial, 16 pt
Text	(blank)



Figure 6B.11 – Placing a Labels on the Form

After typing in Game Over for the last label, we clear out the text so we will insert the text in the object when the game is concluded.

Inserting a Picture into the Form

We will place three pictures on the form, two are for the paddles and the other is the ball. The paddle is 10 pixels wide by 50 pixels tall and the ball is 10 by 10 pixels. We need to open a graphics program and make both of these icons and save them in our Simple Game Program project folder.

We select the toolbox and PictureBox and we draw a box to the right of the labels that will contain the answers. We name the picturebox **imgPaddle1**. We scroll down on the properties window and select the three dots button at the Image property and a Select Resource window will appear.

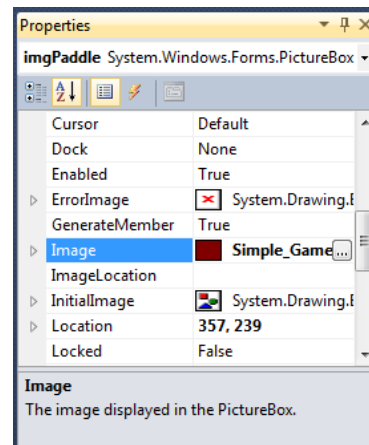


Figure 6B.12 – Adding an Image

We then will import the graphic of our paddle which we made in Microsoft Paint and saved as a bitmap image. We then press the OK button and the image will appear in the picturebox.

Alphabetic	
(Name)	imgPaddle1
Image	Verticle_paddle.bmp
Location	14,152
Size	10,50

Next, we will add a second paddle on the right hand side of the window using the same procedure.

Alphabetic	
(Name)	imgPaddle2
Image	Verticle_paddle.bmp
Location	560,152
Size	10,50

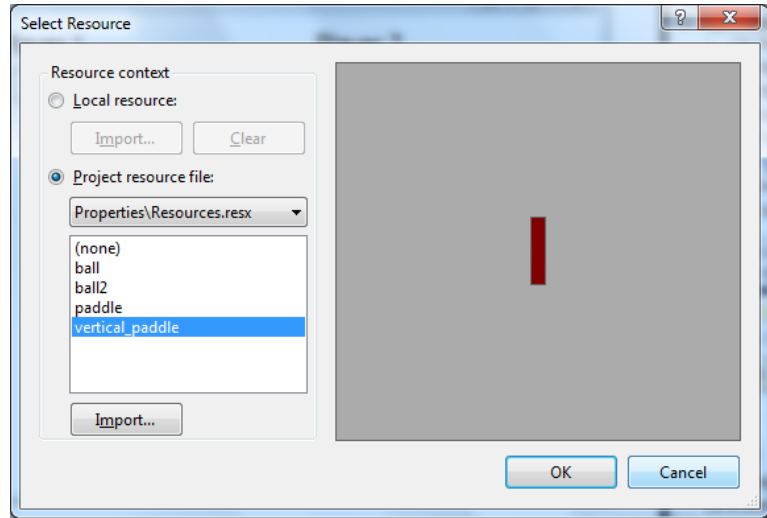


Figure 6B.13 – Import the Paddle Image

We then will import the graphic of our ball which we made in Microsoft Paint and saved as a JPEG image. We then press the OK button and the image will appear in the picturebox.

Alphabetic	
(Name)	imgBall
Image	Ball.jpg
Location	542,173
Size	10,10

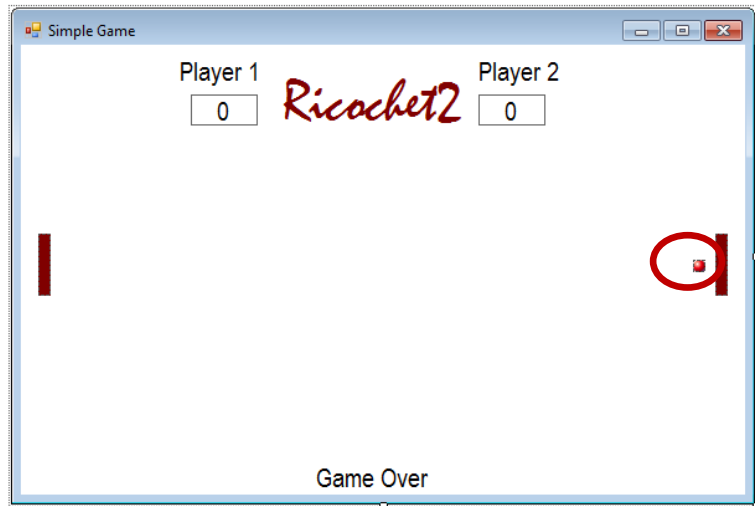


Figure 6B.14 – Import the Ball Image

Adding Comments in Visual C# to Communicate to Others

The comments we placed in the first four 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 Simple Game {`, 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:

```
//Simple Game 2
//This program will allow two players to return a ball and continue to earn points
//If the ball passes the paddle of the other player, the sending player receives the point
//and immediately serves another ball. When one player earns five points, they win.
```

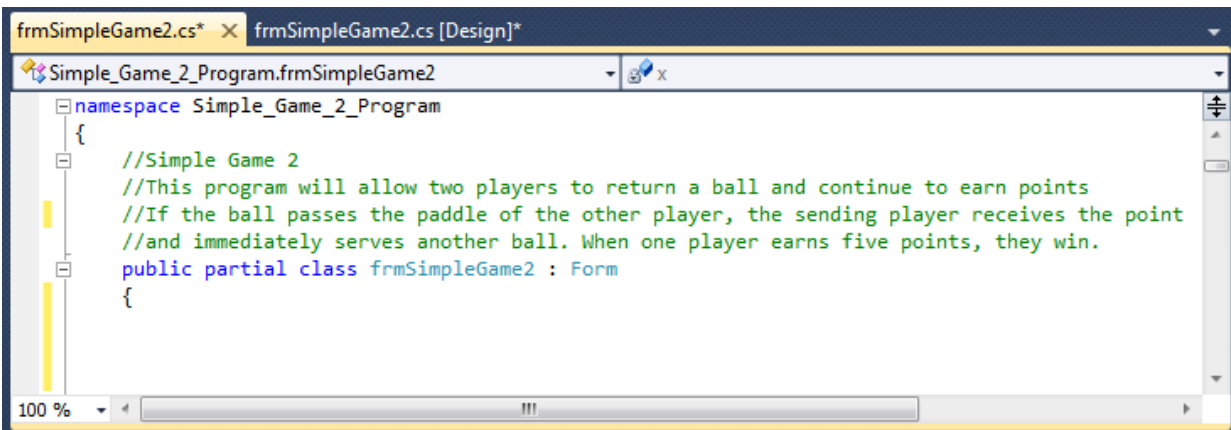


Figure 6B.15 – Adding an Introduction

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`), `dy` (change of `y`), `ptx1`, `pty1` for the paddle1 location, `ptx2`, `pty2` for the paddle2 location, `score1` for player 1 score and `score2` for player 2 score. These variables will hold whole numbers for movement on the form in terms of pixels, so we will declare all ten as integers.

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

```
public partial class frmSimpleGame2 : 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;     //speed is the change of x
int dy;     //speed is the change of y
int ptx1;   //x coord of the paddle 1
int pty1;   //y coord of the paddle 1
int ptx2;   //x coord of the paddle 2
int pty2;   //y coord of the paddle 2
int score1; //player 1 score
int score2; //player 2 score

```

The integers x and y represent the actual position of the ball in the Simple Game Program. The dx and dy are the change in the x position for movement of the timer. The ptx1 (ptx2) and pty1 (pty2) are the location of the paddle so we can calculate the where the 50 pixel long line is to determine whether the ball hit that region. The score1 (score2) is originally set to zero and increases by one each time the ball goes by an opponent's paddle, so we need a variable for it. When one of the integers, score1 or score 2 reaches zero, we will end the game.

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

```

frmSimpleGame2.cs* X frmSimpleGame2.cs [Design]*
Simple_Game_2_Program.frmSimpleGame2 frmSimpleGame2()
namespace Simple_Game_2_Program
{
    //Simple Game 2
    //This program will allow two players to return a ball and continue to earn points
    //If the ball passes the paddle of the other player, the sending player receives the point
    //and immediately serves another ball. When one player earns five points, they win.
    public partial class frmSimpleGame2 : 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;         //speed is the change of x
        int dy;         //speed is the change of y
        int ptx1;       //x coord of the paddle 1
        int pty1;       //y coord of the paddle 1
        int ptx2;       //x coord of the paddle 2
        int pty2;       //y coord of the paddle 2
        int score1;     //player 1 score
        int score2;     //player 2 score

        public frmSimpleGame2()
        {
            InitializeComponent();
        }
    }
}

```

Figure 6B.16 – 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

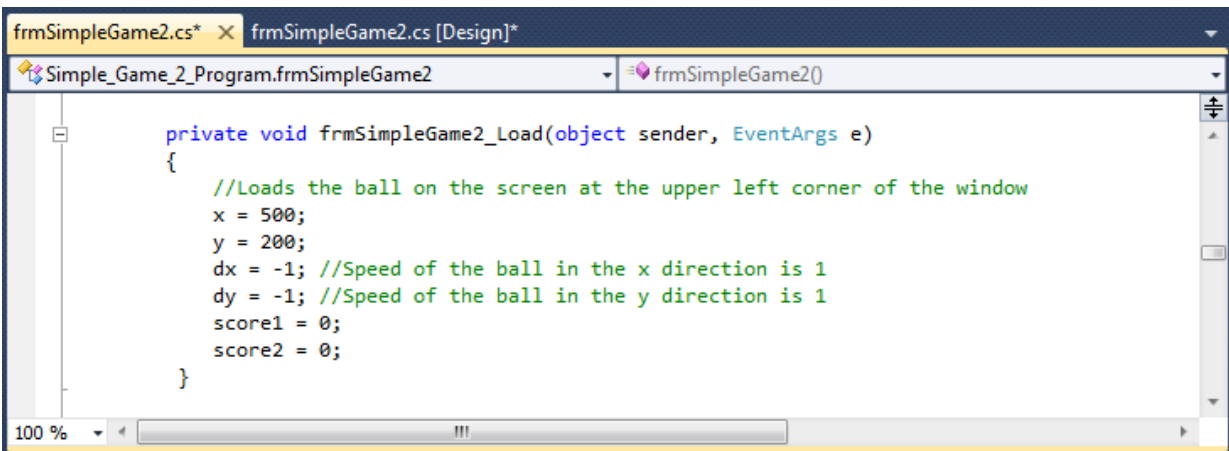
Next, we will set the variables using the equal function. We will set the numbers in the two textboxes to their variable and we compute resistance by division and the power by using multiplication.

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

```
private void frmSimpleGame2_Load(object sender, EventArgs e)
{
    //Loads the ball on the screen at the upper left corner of the window
    x = 500;
    y = 200;
    dx = -1; //Speed of the ball in the x direction is 1
    dy = -1; //Speed of the ball in the y direction is 1
    score1 = 0;
    score2 = 0;
}
```

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 x equals 500 and y equals 200. We keep the change of x (dx) and the change of y (dy) the same throughout the entire program at 1 or -1. The score1 and score 2 variable are set at 0.

The image is a screenshot of a Visual Studio code editor window. The title bar shows 'frmSimpleGame2.cs*' and 'frmSimpleGame2.cs [Design]*'. The editor content shows the following C# code:

```
private void frmSimpleGame2_Load(object sender, EventArgs e)
{
    //Loads the ball on the screen at the upper left corner of the window
    x = 500;
    y = 200;
    dx = -1; //Speed of the ball in the x direction is 1
    dy = -1; //Speed of the ball in the y direction is 1
    score1 = 0;
    score2 = 0;
}
```

The code is color-coded: keywords are blue, comments are green, and identifiers are black. The editor interface includes a scrollbar on the right and a status bar at the bottom showing '100 %'.

Figure 6B.17 – Setting the Variables in the C# Code

Adding a Timer to the Program

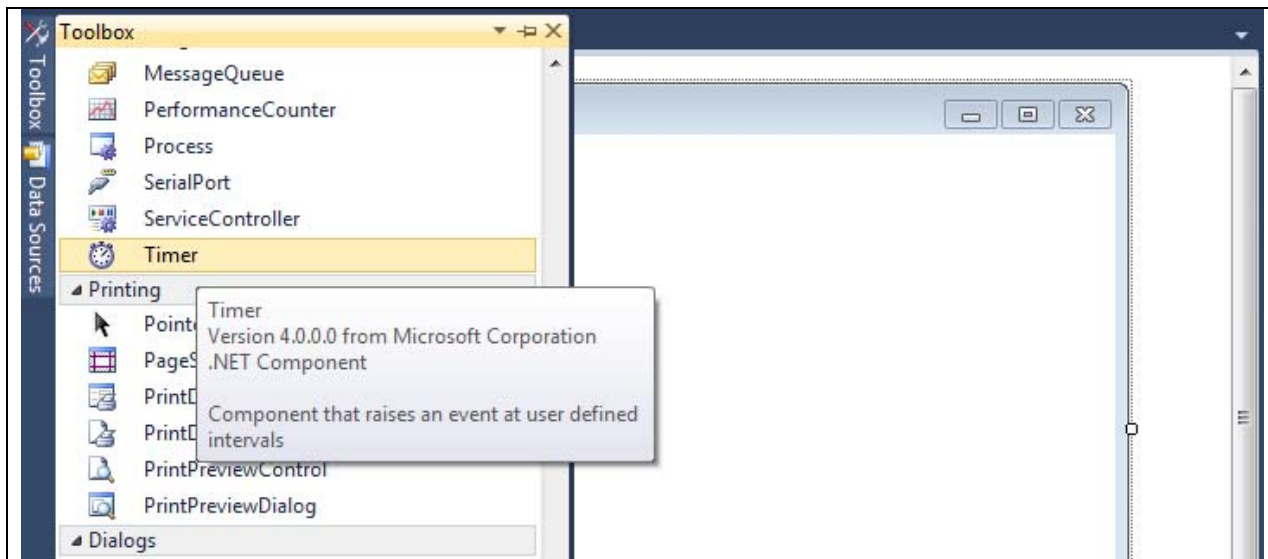


Figure 6B.18 – Adding a Timer

We will next add a Timer to the Simple Game 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.

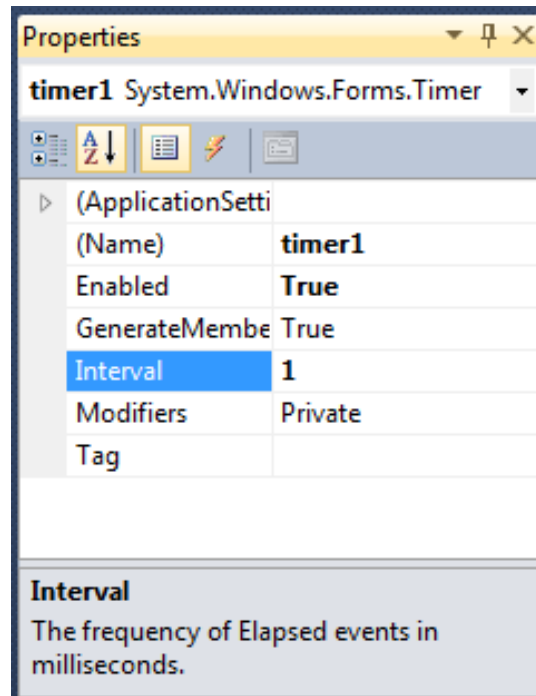


Figure 6B.19 – Setting the Timer's Properties

Programming for the Timer

We will double click on the timer at the bottom of the form and we add code to change the x and y coordinates, the direction of the ball using dx and dy, We also will add points to the score and deduct points from the number of lives left.

The first thing we do in the timer subroutine is set the ball location. Then we change the x coordinate by the value of dx. If the ball is at either the left vertical boundary ($x < 0$) or the right vertical boundary ($x + 10 > \text{this.ClientSize.Width}$), then we change the dx direction. `This.ClientSize.Width` is the number of pixels of the Window's width.

The next order of coding is to add dy to the y coordinate. If the ball is at the top horizontal boundary ($y < 0$), then we change the dy direction. The next check is to examine three points of logic. First we retrieve the location of the paddle. With an If statement, we will see if the ball's x coordinate is equal or greater than the paddle's location and less or equal than the width of the paddle. Finally, we check if the y + 10 of the ball is greater than the y coordinate of the paddle and if all conditions are met, we change the direction of the ball, add one to the score and post the new score on the game window. The next If statement will detect that the ball has gone by the paddle. Then we have an If statement to check to see if the game is over.

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

```
private void timer1_Tick(object sender, EventArgs e)
{
    imgBall.Location = new Point(x, y);
    x = x + dx;

    y = y + dy;
    if (y < 0)
    {
        dy = -dy; /// if y is less than 0 then we change direction
    }
    else if (y + 10 > this.ClientSize.Height)
    {
        dy = -dy; /// if y + 10, the radius of the circle is greater than the form height then we change direction
    }

    //Paddle 1
    Point point1 = this.imgPaddle1.Location;
    ptx1 = (point1.X);
    pty1 = (point1.Y);
    if (y >= pty1)
    {
        if (y <= (pty1 + 50))
        {
            if (x < (ptx1 + 11))
```

```

        {
            dx = -dx;
        }
    }
}
if (x < ptx1 + 9)
{
    score2 = score2 + 1;
    lblPlayer2.Text = Convert.ToString(score2);
    dx = -1;
    dy = -1;
    x = 500;
    y = pty2;
}

```

//Paddle 2

```

Point point2 = this.imgPaddle2.Location;
ptx2 = (point2.X);
pty2 = (point2.Y);
if (y >= pty2)
{
    if (y <= (pty2 + 50))
    {
        if ((x + 10) > ptx2)
        {
            dx = -dx;
        }
    }
}
if ((x + 9) > ptx2)
{
    score1 = score1 + 1;
    lblPlayer1.Text = Convert.ToString(score1);
    dx = 1;
    dy = 1;
    x = 100;
    y = pty1;
}

```

//Game Over code

```

if (score1 == 5)
{
    lblGameOver.Text = "Game Over";
    dx = 0;
    dy = 0;
}
if (score2 == 5)

```

```

    {
        lblGameOver.Text = "Game Over";
        dx = 0;
        dy = 0;
    }

    this.Invalidate();
}

```

Programming the Paddle

Brand new in this lesson is making left and right arrow keys move the paddle to the left and right in the game. We need to add the following to the `public frmSimpleGame2()` portion of the code and right below `InitializeComponent()`:

```
KeyDown += new KeyEventHandler(frmSimpleGame2_KeyDown);
```

Then we add the `private void frmSimpleGame_KeyDown(object sender, KeyEventArgs e)` subroutine. The first two lines of code obtain the location of the paddle1 and assign it to `px1` and `py1`. The next two lines of code obtain the location of the paddle2 and assign it to `px2` and `py2`. If the Z key is pressed, paddle1 moves 5 pixels downward. If the A key is pressed, paddle1 moves 5 pixels upward. If the down arrow is pressed, paddle2 moves 5 pixels downward. If the up arrow is pressed, paddle2 moves 5 pixels upward.

The last if statement keeps the paddles in the window boundary. So if the key is pressed when the paddle is on the boundary, the location stays the same.

Type the following code:

```

public frmSimpleGame2()
{
    InitializeComponent();
    KeyDown += new KeyEventHandler(frmSimpleGame2_KeyDown);
}

private void frmSimpleGame2_KeyDown(object sender, KeyEventArgs e)
{
    int px1 = imgPaddle1.Location.X;
    int py1 = imgPaddle1.Location.Y;
    int px2 = imgPaddle2.Location.X;
    int py2 = imgPaddle2.Location.Y;

    //Paddle 1
    if (e.KeyCode == Keys.Z) py1 += 5;
    else if (e.KeyCode == Keys.A) py1 -= 5;
}

```



```

if (py1 < 0)
{
    py1 = 0;
}
if (py1 > this.ClientSize.Height - 50)
{
    py1 = this.ClientSize.Height - 50;
}

// Paddle 2
if (e.KeyCode == Keys.Down) py2 += 5;
else if (e.KeyCode == Keys.Up) py2 -= 5;

if (py2 < 0)
{
    py2 = 0;
}
if (py2 > this.ClientSize.Height - 50)
{
    py2 = this.ClientSize.Height - 50;
}

imgPaddle1.Location = new Point(px1, py1);
imgPaddle2.Location = new Point(px2, py2);
}

```

Running the Program

After noting that the program is saved, press the F5 to run the Simple Game 2 application. The Simple Game 2 window will appear on the graphical display. The ball will be moving down from the top and we have the ability to use the arrow keys to stop the ball from getting by the paddle.

After playing the Simple Game 2 program, click on the red X in the upper right corner to close the application.

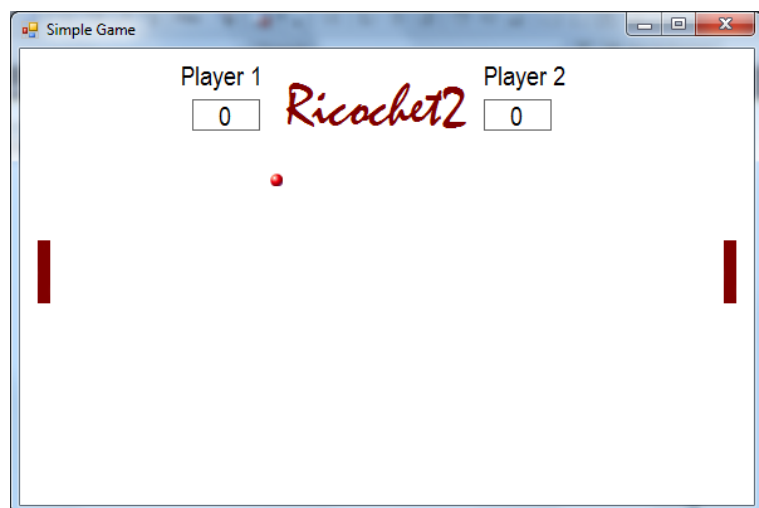


Figure 6B.20 – Launching the Program

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 Simple_Game_2_Program
{
    //Simple Game 2
    //This program will allow two players to return a ball and continue to earn points
    //If the ball passes the paddle of the other player, the sending player recieves the point
    //and immediately serves another ball. When one player earns five points, they win.
    public partial class frmSimpleGame2 : 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;   //speed is the change of x
        int dy;   //speed is the change of y
        int ptx1; //x coord of the paddle 1
        int pty1; //y coord of the paddle 1
        int ptx2; //x coord of the paddle 2
        int pty2; //y coord of the paddle 2
        int score1; //player 1 score
        int score2; //player 2 score

        public frmSimpleGame2()
        {
            InitializeComponent();
            KeyDown += new KeyEventHandler(frmSimpleGame2_KeyDown);
        }

        private void frmSimpleGame2_KeyDown(object sender, KeyEventArgs e)
        {
            int px1 = imgPaddle1.Location.X;
            int py1 = imgPaddle1.Location.Y;
            int px2 = imgPaddle2.Location.X;
            int py2 = imgPaddle2.Location.Y;
        }
    }
}
```

```

//Paddle 1
if (e.KeyCode == Keys.Z) py1 += 5;
else if (e.KeyCode == Keys.A) py1 -= 5;

if (py1 < 0)
{
    py1 = 0;
}
if (py1 > this.ClientSize.Height - 50)
{
    py1 = this.ClientSize.Height - 50;
}

// Paddle 2
if (e.KeyCode == Keys.Down) py2 += 5;
else if (e.KeyCode == Keys.Up) py2 -= 5;

if (py2 < 0)
{
    py2 = 0;
}
if (py2 > this.ClientSize.Height - 50)
{
    py2 = this.ClientSize.Height - 50;
}

imgPaddle1.Location = new Point(px1, py1);
imgPaddle2.Location = new Point(px2, py2);
}

private void frmSimpleGame2_Load(object sender, EventArgs e)
{
    //Loads the ball on the screen at the upper left corner of the window
    x = 500;
    y = 200;
    dx = -1; //Speed of the ball in the x direction is 1
    dy = -1; //Speed of the ball in the y direction is 1
    score1 = 0;
    score2 = 0;
}

private void timer1_Tick(object sender, EventArgs e)
{
    imgBall.Location = new Point(x, y);
    x = x + dx;

    y = y + dy;
}

```

```

if (y < 0)
{
    dy = -dy; /// if y is less than 0 then we change direction
}
else if (y + 10 > this.ClientSize.Height)
{
    dy = -dy; /// if y + 10, the radius of the circle is greater than the form height then we change direction
}

```

//Paddle 1

```

Point point1 = this.imgPaddle1.Location;
ptx1 = (point1.X);
pty1 = (point1.Y);
if (y >= pty1)
{
    if (y <= (pty1 + 50))
    {
        if (x < (ptx1 + 11))
        {
            dx = -dx;
        }
    }
}
if (x < ptx1 + 9)
{
    score2 = score2 + 1;
    lblPlayer2.Text = Convert.ToString(score2);
    dx = -1;
    dy = -1;
    x = 500;
    y = pty2;
}

```

//Paddle 2

```

Point point2 = this.imgPaddle2.Location;
ptx2 = (point2.X);
pty2 = (point2.Y);
if (y >= pty2)
{
    if (y <= (pty2 + 50))
    {
        if ((x + 10) > ptx2)
        {
            dx = -dx;
        }
    }
}
}

```

```

if ((x + 9) > ptx2)
{
    score1 = score1 + 1;
    lblPlayer1.Text = Convert.ToString(score1);
    dx = 1;
    dy = 1;
    x = 100;
    y = pty1;
}

//Game Over code
if (score1 == 5)
{
    lblGameOver.Text = "Game Over";
    dx = 0;
    dy = 0;
}
if (score2 == 5)
{
    lblGameOver.Text = "Game Over";
    dx = 0;
    dy = 0;
}

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 90-10 * - Write a Visual C# Application that displays a single form and when executed, the program will show a Simple Game 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.