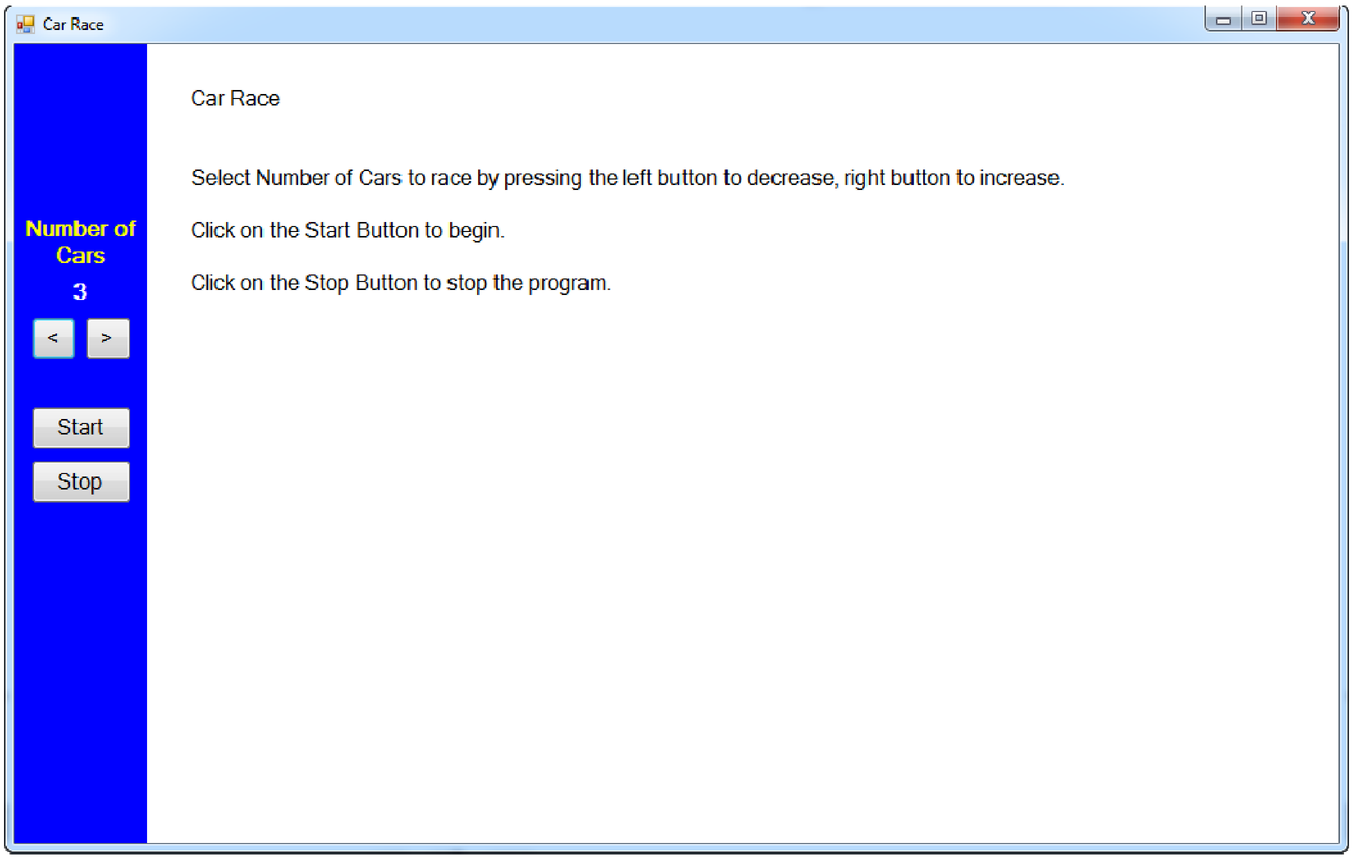
Project 03 – Car Racing

It's a simple game. Up to 10 cars line up to race. You select the car you want to win the race. Clicking a '**Start'** button gets the race going. The cars then move across the screen toward the finish line.

Project Design

Labels are used for instructions and to display the number of cars in the race. Four button controls are included: two to change the number of cars, one to start the game and one to stop. A picture box control holds the car image. A timer control is used to update car motion during the race.

Run the program (press <**F5**>). The game will appear in its 'initial' state. A game description and program instructions are given. At this point, you can click the **Start** button to start the game, click arrow buttons to change the number of cars racing or click **Stop** to stop the program.



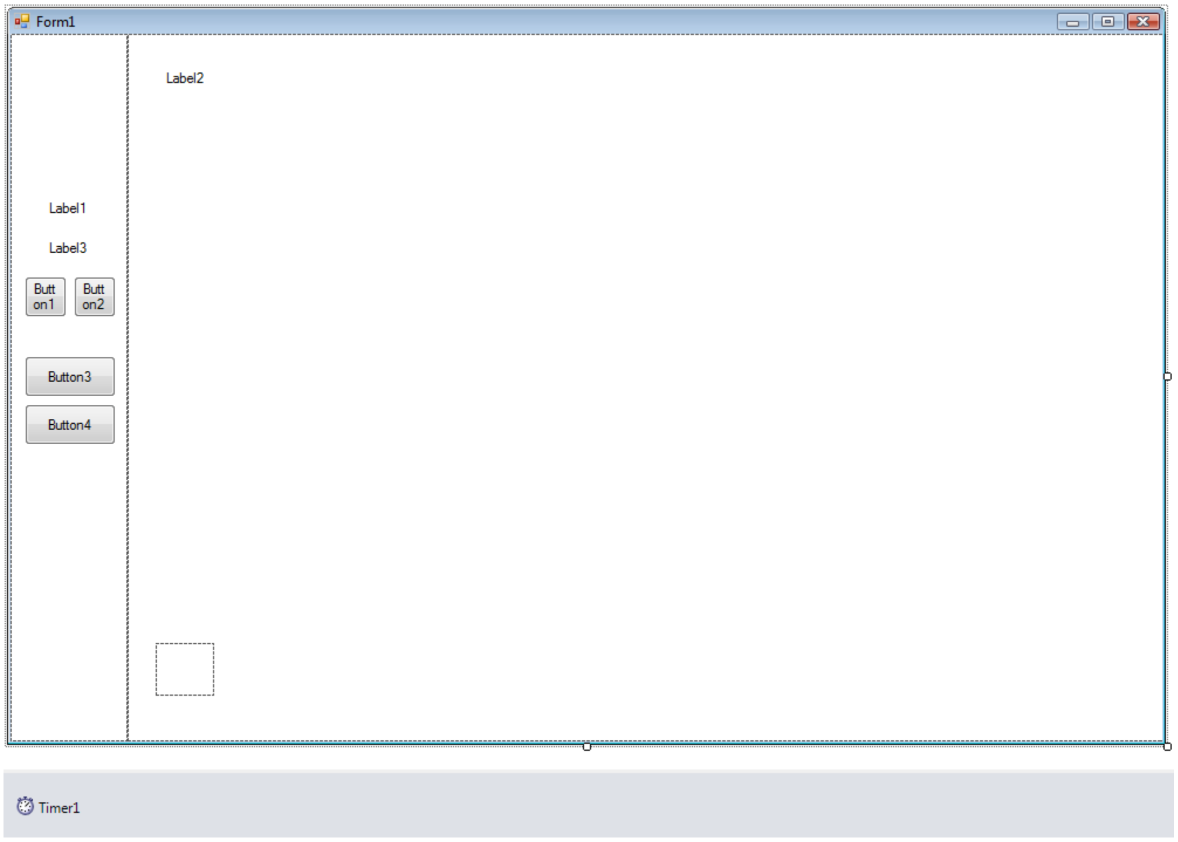
Car Race

Select Number of Cars to race by pressing the left button to decrease, right button to increase.

Click on the Start Button to begin.

Click on the Stop Button to stop the program.

We begin building the **Car Race** game project. Let's build the form. Start a new project in Visual C# Express. Put two panel controls on the form (one 100 pixels wide on the left, one 900 pixels wide on the right). Put two labels and four buttons in the left panel. Put a label control and picture box control on the right panel. Add a timer control. The form should look like this:



The labels are used for displaying information. Two buttons are used to change the number of cars, two to start and stop the game. The timer control establishes the car motion.

Set the control properties using the properties window:

form1 Form:

Property Name	**Property Value**
Name	frmCarRace
Text	Car Race
FormBorderStyle	Fixed Single
StartPosition	CenterScreen

panel1 Panel:

Property Name	**Property Value**
BackColor	Blue

label1 Label:

Property Name	**Property Value**
AutoSize	False
Name	lblCars
FontSize	12
FontStyle	Bold
ForeColor	Yellow
Text	Number of Cars
TextAlign	Middle Center

label2 Label:

Property Name	**Property Value**
AutoSize	False
Name	lblNumOfCars
FontSize	12
FontStyle	Bold
ForeColor	White
Text	0
TextAlign	Middle Center

button1 Button:

Property Name	Property Value
Name	btnDecrease
FontSize	12
Text	<

button2 Button:

Property Name	Property Value
Name	btnIncrease
FontSize	12
Text	>

button3 Button:

Property Name	Property Value
Name	btnStart
FontSize	12
Text	Start

button4 Button:

Property Name	Property Value
Name	btnStop
FontSize	12
Text	Stop

panel2 Panel:

Property Name	Property Value
Name	pnlGame
BackColor	White

label1 Label:

Property Name	Property Value
Name	lblInstructions
FontSize	12

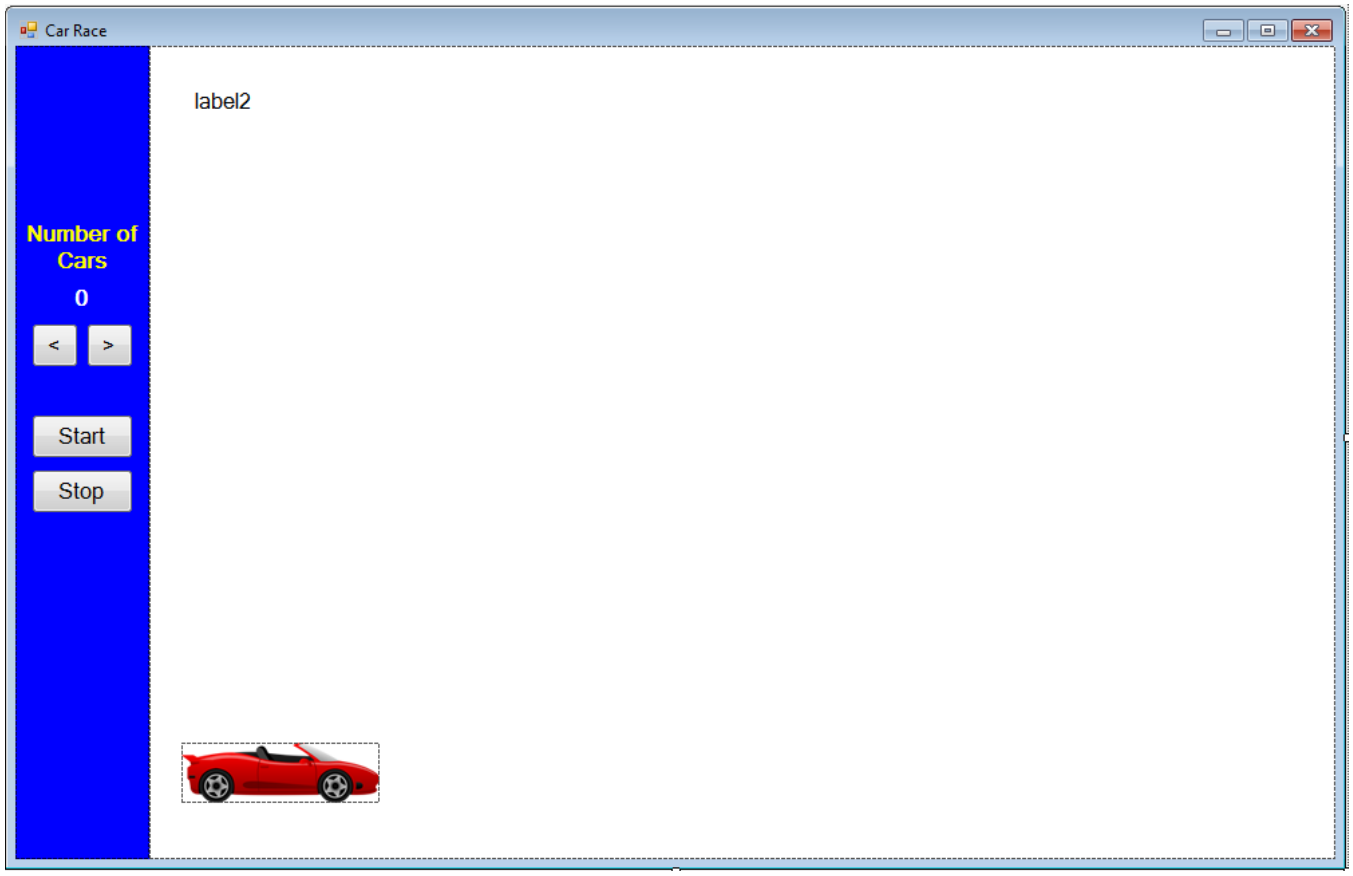
pictureBox1 Picture Box:

Property Name	**Property Value**
Name	picCar
Image	Car.jpg (in **\BVCSE Projects\NoahVCS** folder)
SizeMode	AutoSize
Visible	False

timer1 Timer:

Property Name	**Property Value**
Name	timCar
Interval	100

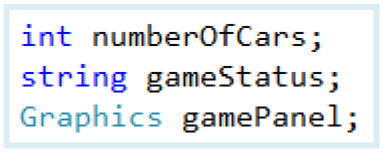
When done, the form should look like this:



Code Design - Initial State (Instructions)

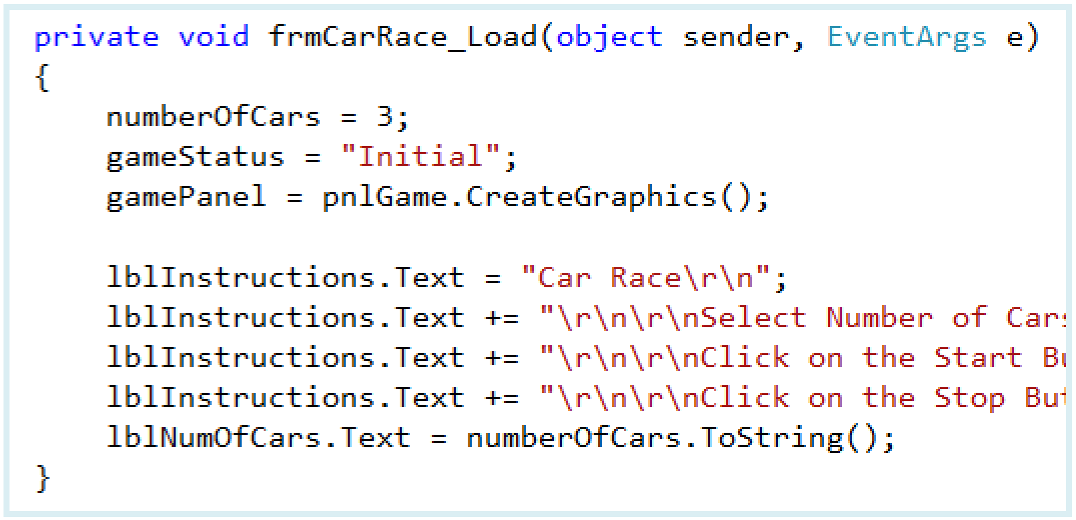
Any time we start a program, there are certain initializations that must take place. When the program first begins, we want to display the game instructions. We also initialize the number of cars (**numberOfCars**) and the game status (**gameStatus**). These initializations are in the form's **Load** procedure.

We declare three variables (**numberOfCars**, **gameStatus** and **gamePanel,** the game graphics object). Do this in the general declarations area:



```
int numberOfCars;
string gameStatus;
Graphics gamePanel;
```

Add this code to the **frmCarRace Load** event procedure:



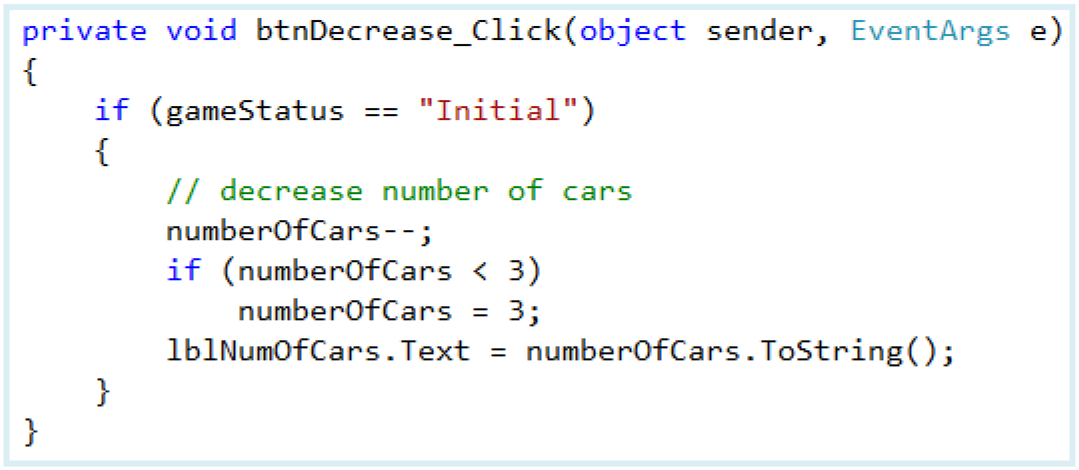
```
private void frmCarRace_Load(object sender, EventArgs e)
{
    numberOfCars = 3;
    gameStatus = "Initial";
    gamePanel = pnlGame.CreateGraphics();

    lblInstructions.Text = "Car Race\r\n";
    lblInstructions.Text += "\r\n\r\nSelect Number of Car
    lblInstructions.Text += "\r\n\r\nClick on the Start B
    lblInstructions.Text += "\r\n\r\nClick on the Stop But
    lblNumOfCars.Text = numberOfCars.ToString();
}
```

As mentioned, we set **numberOfCars** to **3** and the **gameStatus** to "**Initial**". Next, after setting up the graphics object (**gamePanel**), using a series of statements, we write out the game instructions in **lblInstructions**.

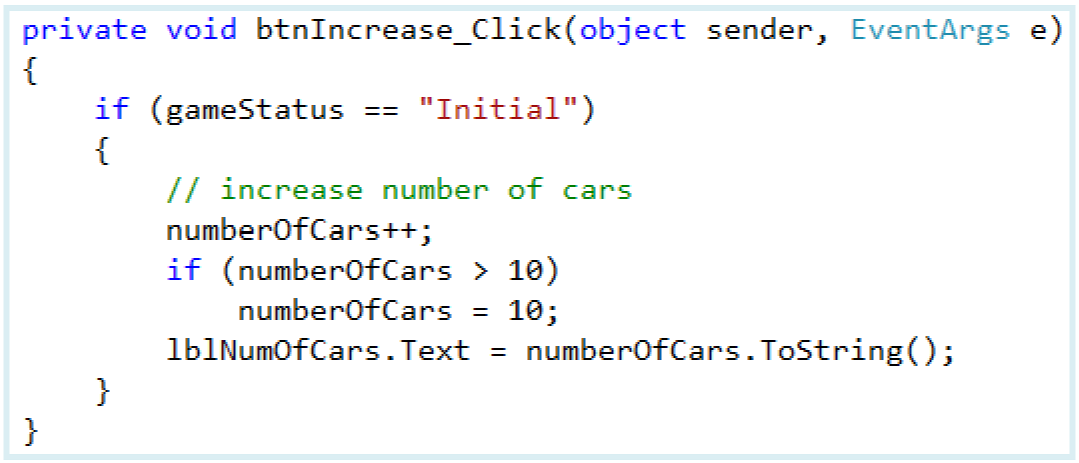
Save and **Run** the program to see:

Code Design – Select Number of Cars

To change the number of cars, a user clicks on either of the two arrow buttons. The code to do this goes in the **btnDecrease** and **btnIncrease Click** event procedures:



```
private void btnDecrease_Click(object sender, EventArgs e)
{
    if (gameStatus == "Initial")
    {
        // decrease number of cars
        numberOfCars--;
        if (numberOfCars < 3)
            numberOfCars = 3;
        lblNumOfCars.Text = numberOfCars.ToString();
    }
}
```



```
private void btnIncrease_Click(object sender, EventArgs e)
{
    if (gameStatus == "Initial")
    {
        // increase number of cars
        numberOfCars++;
        if (numberOfCars > 10)
            numberOfCars = 10;
        lblNumOfCars.Text = numberOfCars.ToString();
    }
}
```

The code is straightforward. The number of cars can only be changed when the game is in "**Initial**" state. If the **<** button is clicked, we decrease the number of cars by one (minimum of 3). If the **>** button is clicked, we increase the number of cars (maximum of 10). The new value for the number of cars is then displayed.

Save and **Run** the program. Make sure the arrow buttons work – make sure the number of cars is always between 3 and 10.

Code Design – Initial to Stopped State

Once the number of cars is finalized, the next step is to click the **Start** button to allow selection of the cars you want to win. Hence, we need code to move from the “**Initial**” state to “**Stopped**” state. When the **Start** button is clicked in “**Initial**” state, we need to take the following steps:

- Change **gameStatus** to “**Stopped**”
- Clear game screen
- Draw the cars and “lane lines” for each
- Draw the finish line
- Display instructions for the user

To draw the cars, we divide the race region into 10 equal size areas (set up for the maximum number of cars). We then just move down the screen adding each car.

The image stored in **picCar** is used to represent the car:

The car is 148 pixels wide and 45 pixels high. (we will shrink it to half its actual size)

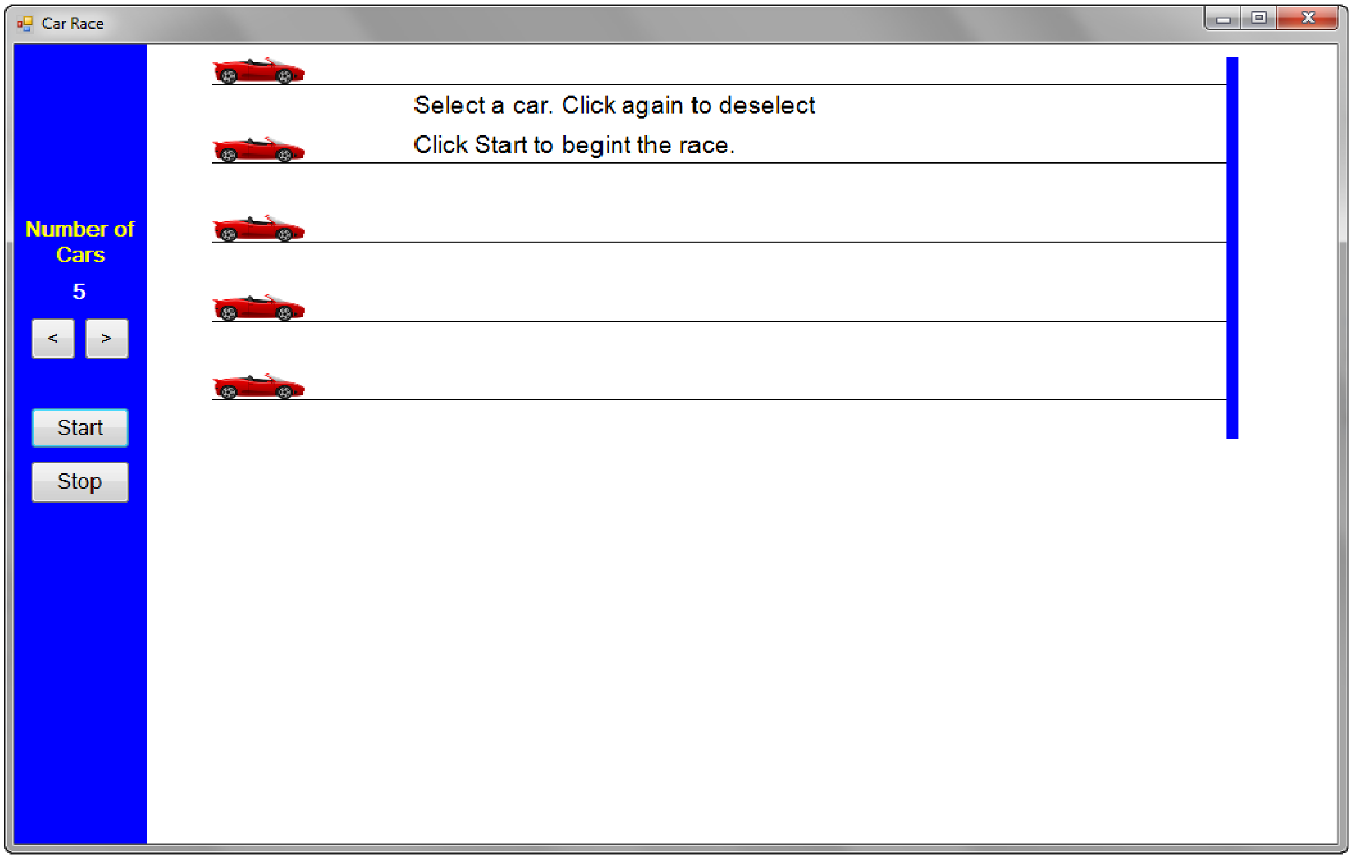
Now, add this code to **btnStart Click** event procedure:

```
private void btnStart_Click(object sender, EventArgs e)
{
    if (gameStatus == "Initial")
    {
        //clicked start
        gameStatus = "Stopped";
        lblInstructions.Visible = false;
        gamePanel.Clear(Color.White);
        //draw cars
        for (int i = 0; i < numberOfCars; i++)
        {
            gamePanel.DrawImage(picCar.Image, 50, i * 60 + 10,
                70, 20);
            gamePanel.DrawLine(Pens.Black, 50, i * 60 + 30,
                820, i * 60 + 30);
        }
        // draw finish line
        gamePanel.FillRectangle(Brushes.Blue, 820, 10, 10,
            numberOfCars * 60 - 10);
        Font myFont = new Font("Arial", 14);
        gamePanel.DrawString("Select a car. Click again to deselect",
            myFont, Brushes.Black, 200, 35);
        gamePanel.DrawString("Click Start to begint the race.",
            myFont, Brushes.Black, 200, 65);
        carsSelected = 0;
        for (int i = 0; i < numberOfCars; i++)
            carPicked[i] = 0;
    }
```

You should be able to identify each step in this new code. Notice in particular how we draw each car. Two new variables are introduced. **carsSelected** (the number of cars picked by the user) is initialized at zero. The array **carPicked** is used to keep track of which cars are selected by the user. Add these lines in the general declarations area:

```
int carsSelected;
int[] carPicked = new int[10];
```

Save and **Run** the program. Choose a number of cars (I picked 5). Click **Start** and you should see:

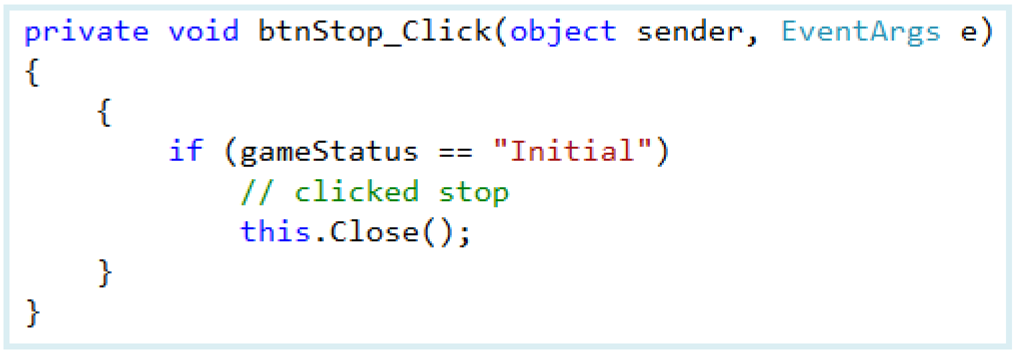


Select a car. Click again to deselect
Click Start to begint the race.

The next step is to select two cars you want in the race. We do that selection code soon. But first, let's take care of one last thing you can do while in "**Initial**" state – stop the program.

Code Design – Stop the Program

One last thing you can do while the game is in “**Initial**” state is to click on the **Stop** button. This will stop the game. The code here is simple. Add the few lines in the **btnStop Click** event procedure:



```
private void btnStop_Click(object sender, EventArgs e)
{
    {
        if (gameStatus == "Initial")
            // clicked stop
            this.Close();
    }
}
```

As desired, if the **Stop** button is clicked, the program ends.

Save and **Run** the program. Make sure the **Stop** button works.

Code Design – Selecting Cars

Once the game is in “**Stopped**” state, the user picks the car they want in the race. To detect clicks on a car, we need to know their location and size. From the code placing the cars on **pnlGame**, you should see that for car i (where **i** goes from **0** to **numberOfCars - 1**):

located at (50, i * 60 +10), width 70, height 20

The array **carPicked** is used to tell us which car has been selected. A red **X** will appear next to a selected car. Once the car is selected (**carsSelected = 1**), you can start the race by clicking **Start**. If you select a car and want to change your mind, click on a selected car and the red **X** will disappear allowing you to pick another.

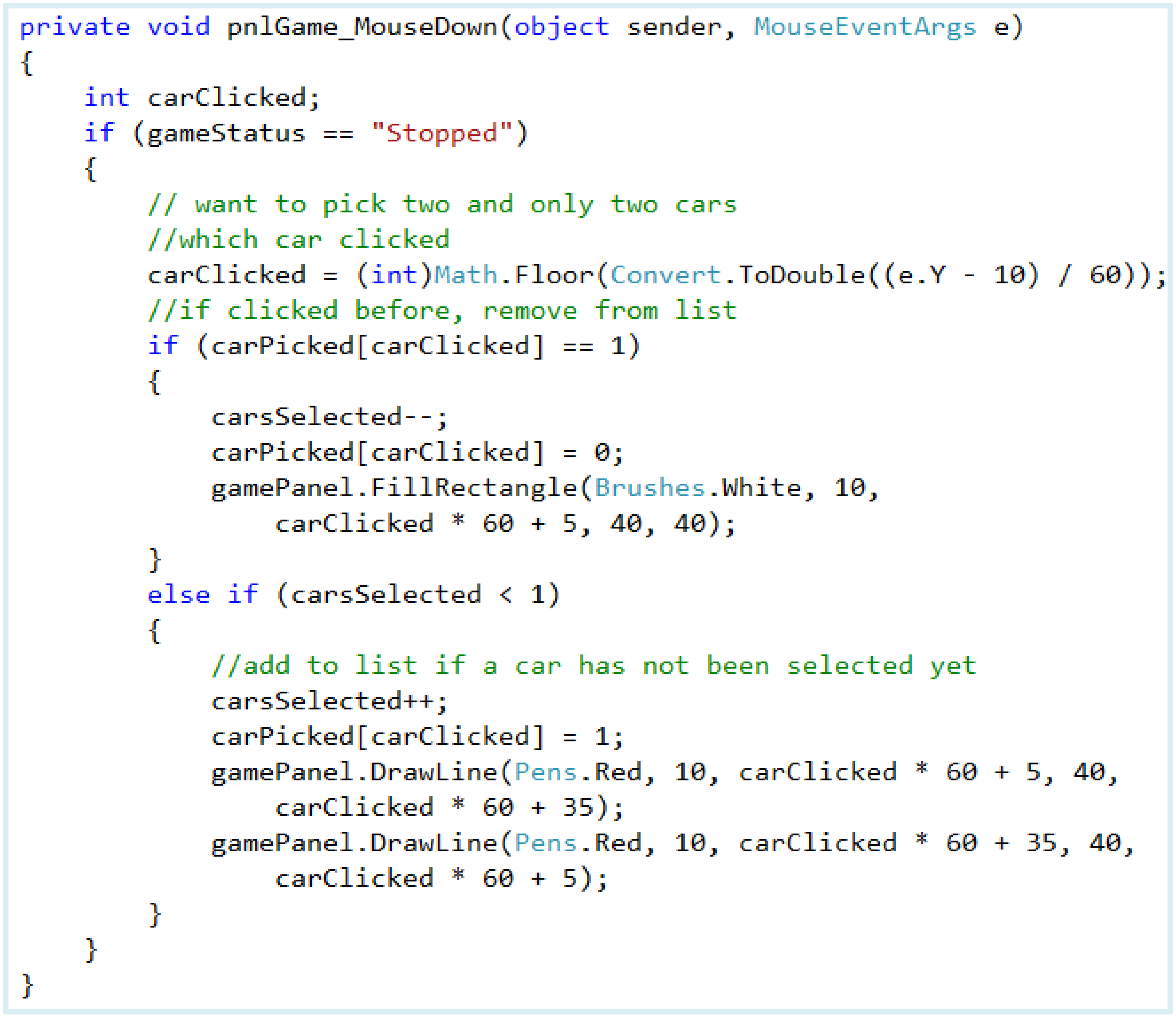
When a car i is clicked, the following steps are taken:

- If car has been selected previously, decrement **carsSelected**, set **carPicked[i]** to **0**. Remove red **X** next to car.
- If car has not been selected and **carsSelected** is less than 1, increment **carsSelected**, set **carPicked[i]** to **1**. Draw red **X** next to car.

We do the steps in this order to make sure we don’t click on a car that has already been selected.

The code for car selection is placed in the **pnlGame MouseDown** event procedure:

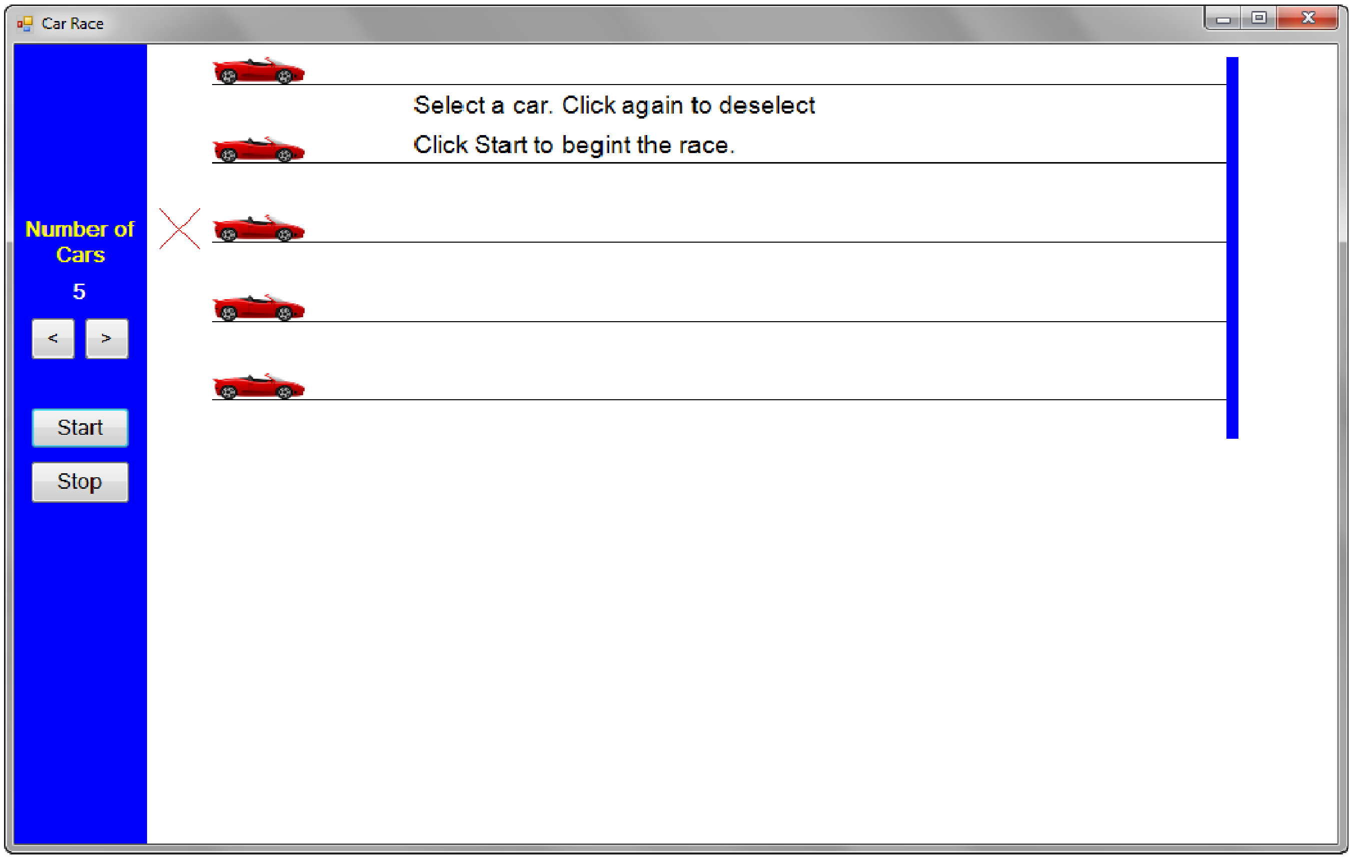
Click on the ⚡ and double-click on MouseDown to create the event.



```
private void pnlGame_MouseDown(object sender, MouseEventArgs e)
{
    int carClicked;
    if (gameStatus == "Stopped")
    {
        // want to pick two and only two cars
        //which car clicked
        carClicked = (int)Math.Floor(Convert.ToDouble((e.Y - 10) / 60));
        //if clicked before, remove from list
        if (carPicked[carClicked] == 1)
        {
            carsSelected--;
            carPicked[carClicked] = 0;
            gamePanel.FillRectangle(Brushes.White, 10,
                carClicked * 60 + 5, 40, 40);
        }
        else if (carsSelected < 1)
        {
            //add to list if a car has not been selected yet
            carsSelected++;
            carPicked[carClicked] = 1;
            gamePanel.DrawLine(Pens.Red, 10, carClicked * 60 + 5, 40,
                carClicked * 60 + 35);
            gamePanel.DrawLine(Pens.Red, 10, carClicked * 60 + 35, 40,
                carClicked * 60 + 5);
        }
    }
}
```

You should see how the clicks on the cars are handled.

Save and **Run** the program. Choose a number of cars (I picked 5). Click **Start**. Then, select two cars for your race. I picked the first and last car:



Select a car. Click again to deselect
Click Start to begint the race.

Make sure you can ‘unselect’ a car. Now, let’s race!

Code Design - Stopped to Moving State

Once cars are selected, we click the **Start** button to get the race started, moving to "**Moving**" state. So, when the **Start** button is clicked in "**Stopped**" state, we take these steps:

- Change **gameStatus** to "**Moving**".
- Clear instructions.
- For each car, pick a random final position before the finish line.
- Pick two winners at random. Assign these winners final positions after the finish line.
- Move the cars to their final positions.
- Change **gameStatus** to "**Initial**" to allow another race.

What we do is pick two of the displayed cars to be winners. These cars finish across the line – all others finish before the line.

To move the cars, we use the timer control (**tmrCar**). With each **Tick** event, we move each car some fraction of the difference between its initial and final postion. This method provides a smooth animation from start to finish. With the cars, we want to move each from its starting position to final position in 5000 milliseconds (5 seconds). Since **Interval** is 100 milliseconds, we move each car 1/50th of its moved distance with each **Tick** event. You can change these values if you want.

The code that starts the cars moving is under the **btnStart Click** event procedure:

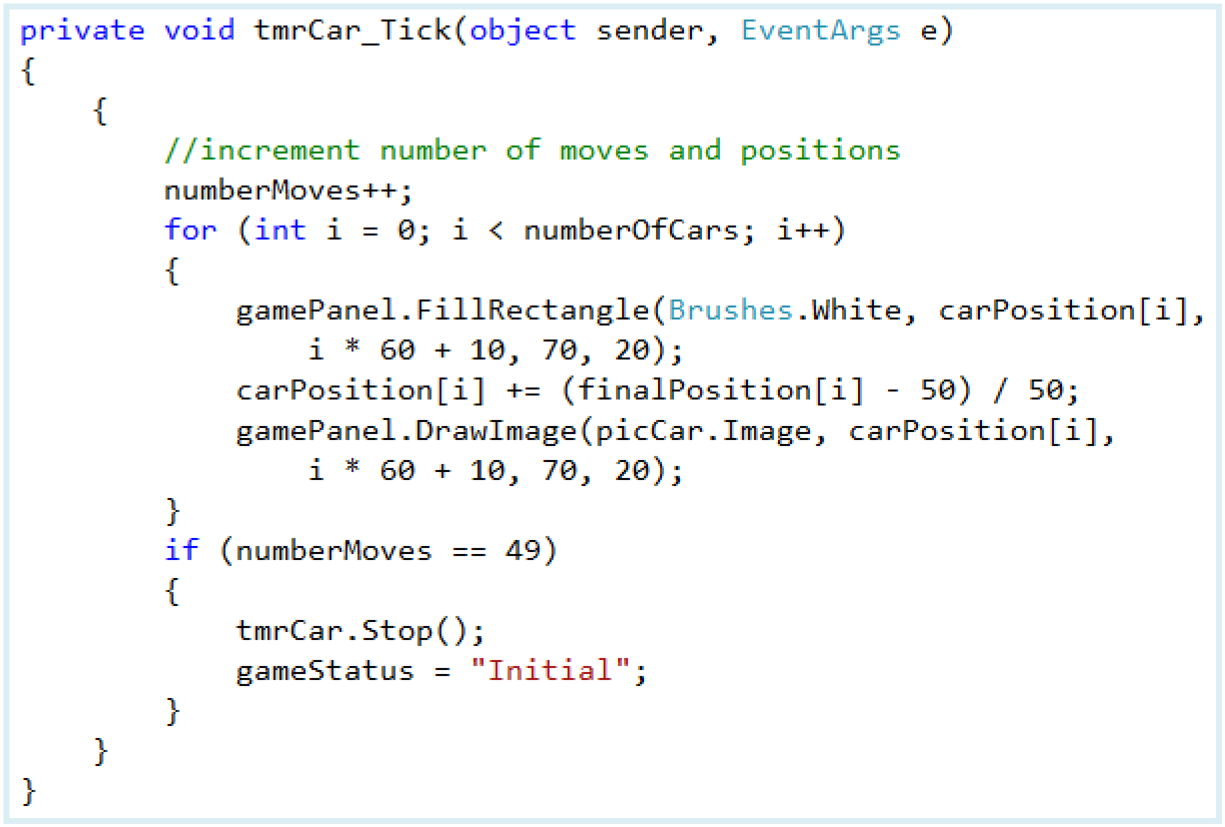
```
    carsSelected = 0;
    for (int i = 0; i < numberOfCars; i++)
        carPicked[i] = 0;
}
else if (gameStatus == "Stopped" && carsSelected == 1)
{
    //clicked start after picking your car
    gameStatus = "Moving";
    gamePanel.FillRectangle(Brushes.White, 200, 37, 400, 50);
    //set start position/get final positions
    numberMoves = 0;
    for (int i = 0; i < numberOfCars; i++)
    {
        carPosition[i] = 50;
        finalPosition[i] = 400 + myRandom.Next(360);
    }
    // pick a winner
    winner = myRandom.Next(numberOfCars);
    finalPosition[winner] = 850;
    tmrCar.Start();
}
```

Notice how the final positions and winners are determined. Several new variables are introduced and are declared in the general declarations area:

```
int numberMoves;
int[] carPosition = new int[10];
int[] finalPosition = new int[10];
int winner;
Random myRandom = new Random();
```

numberMoves is how many times the timer **Tick** event has been called. **carPosition** is an array of current car locations, while **finalPosition** is the array of final positions. The **winner** variable holds the winner. **myRandom** is used to generate random locations.

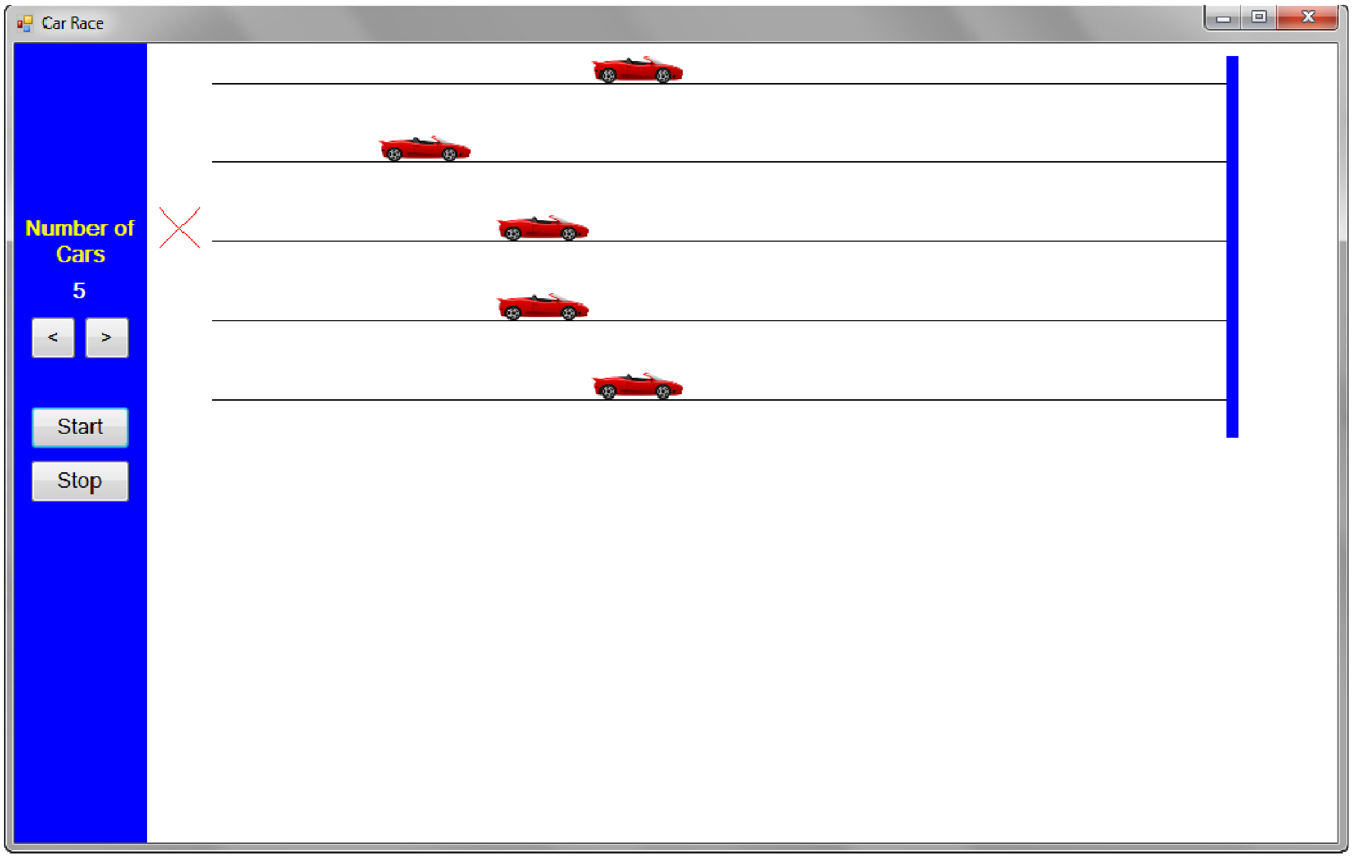
Motion begins by starting **tmrCar**. The code that moves the cars is placed in the **tmrCar Tick** event procedure:



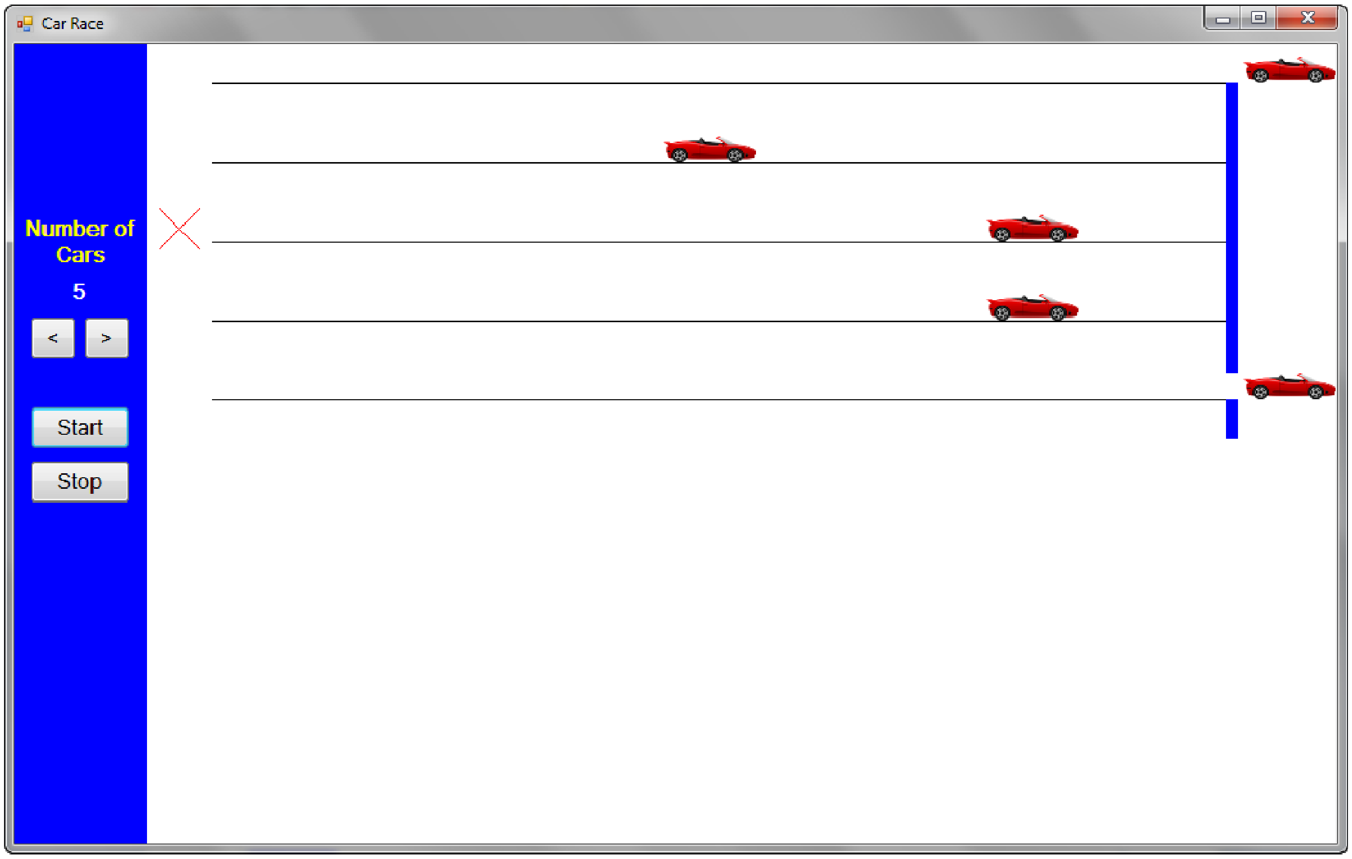
```
private void tmrCar_Tick(object sender, EventArgs e)
{
    {
        //increment number of moves and positions
        numberMoves++;
        for (int i = 0; i < numberOfCars; i++)
        {
            gamePanel.FillRectangle(Brushes.White, carPosition[i],
                i * 60 + 10, 70, 20);
            carPosition[i] += (finalPosition[i] - 50) / 50;
            gamePanel.DrawImage(picCar.Image, carPosition[i],
                i * 60 + 10, 70, 20);
        }
        if (numberMoves == 49)
        {
            tmrCar.Stop();
            gameStatus = "Initial";
        }
    }
}
```

Once the cars are moved, you can play again by choosing a new number of cars (if you want) and click **Start**. Or, you can click **Stop** to end the program.

One last time, **Save** and **Run** the program. Choose a number of cars (again, I picked 5). Click **Start**, choose two cars (I chose the top and bottom ones). Click **Start** –and the cars start the race. Here's the middle of my race:



And the finish:



Level 4+ Achievements

- Have different color cars for each racer.
- The program currently doesn't tell the user whether they have won or not. Have a message box display the results. (Win, Lost or Tie)