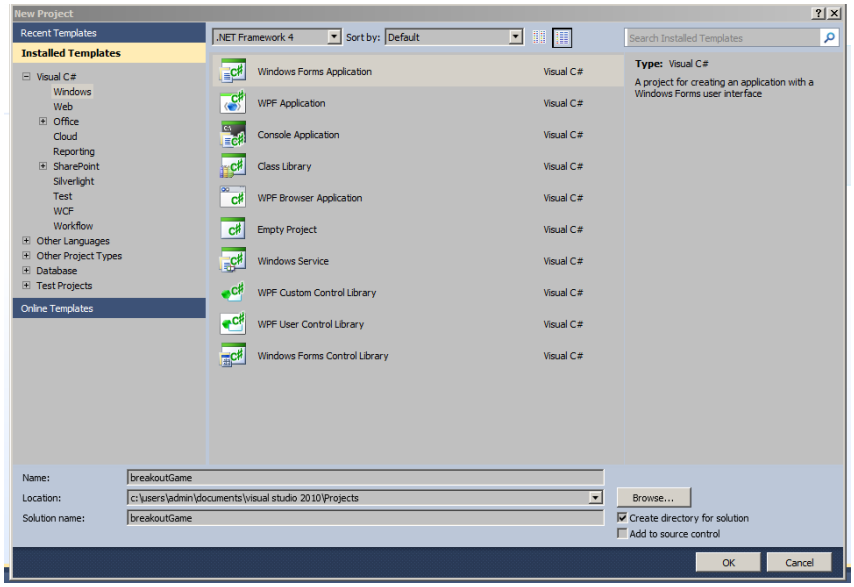
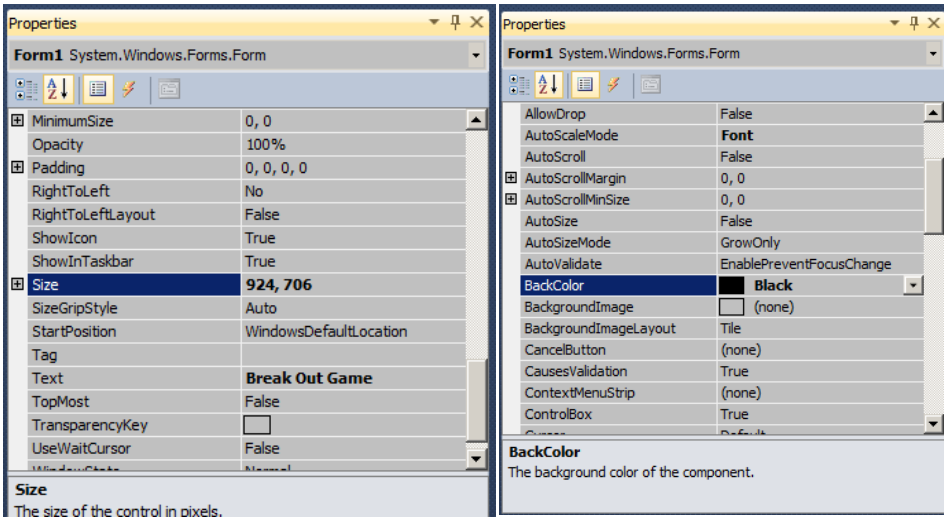
C# Breakout Game Development Tutorial

This is a tutorial for the beginner programmers. There are many different way to create this classic game, in this tutorial we have aimed it for the beginner programmer to get a result for their efforts.

Open Visual Studio, Create a new project in C# Windows Form application and name it **breakoutgame.**



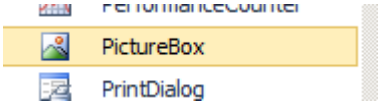
Click on the form and change the following in the properties window

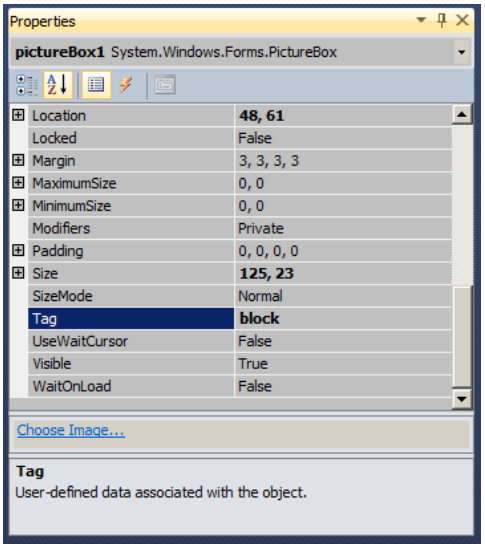


Change the Text to Break Out Game and change the size 924 and 706. 924 is the width of the form and 706 is the height of the form.

Find the **BackColor** option and change it to black for the form.

We will add a picture box to the form

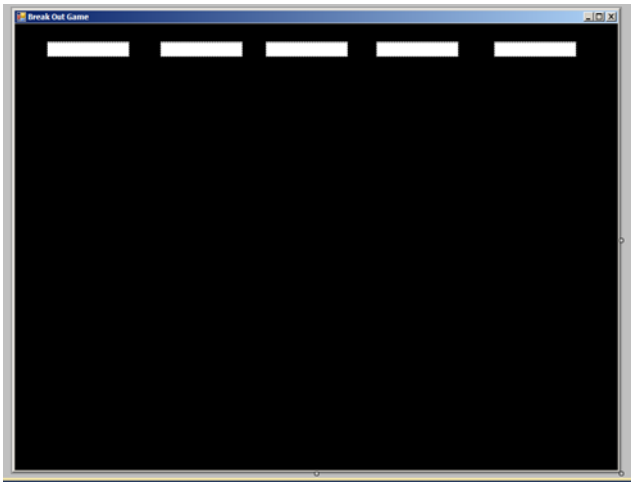




Change the size to **125** and **23** also change the TAG section to **block**

Change the **BackColor** to white so we can see it. 

Now we are going to copy this picture box 4 times to side

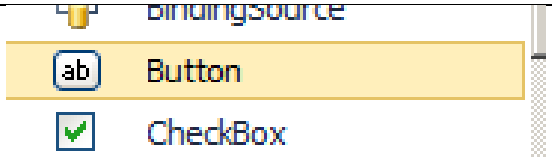


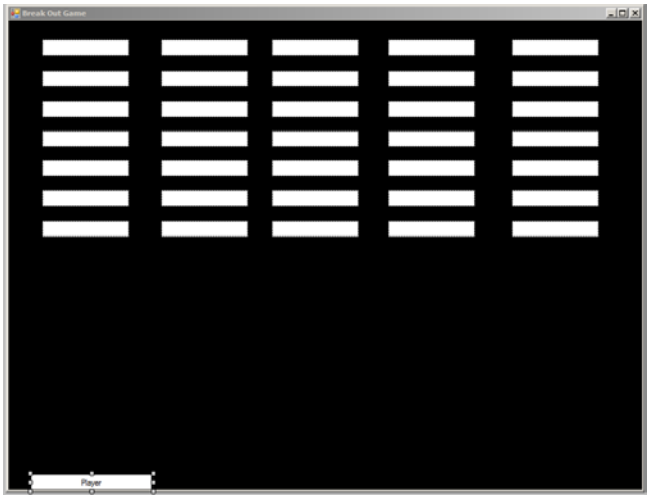
Now drag across and select all of them and copy and paste to the bottom of it 6 times till you get the same outlook as below.

Now we have total of 35 picture boxes tagged as block in the form. It's important to remember this number.

All of these picture boxes have the same height, width and tag name which will be used in the code to identify them.

Now add a button to the form

	Change the following in the properties for the button **(Name)** = player **BackColor** = White **Enabled** = false **Location** = 33, 657 **Size** = 175, 23 **Text** = Player



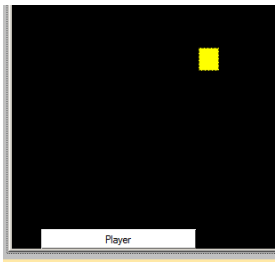
Now let's add another picture box to form. This picture box will used as the ball.

Once you added the picture box to the form change the properties to the following

(Name) = ball

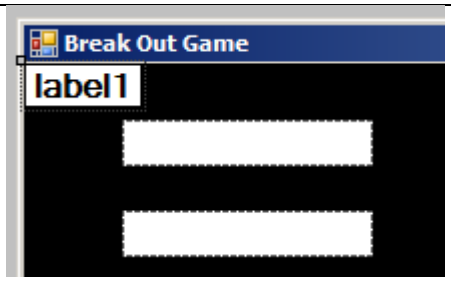
BackColor = Yellow

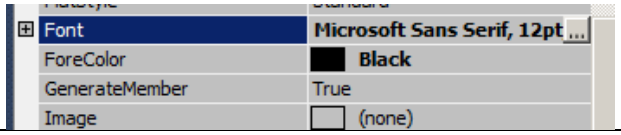
Size = 23, 26



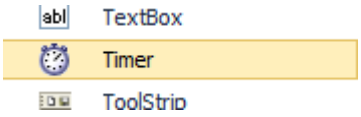
Lastly add a label to screen this will keep the score for us.

	Change the following in the properties. Place this label on top left corner of the screen.

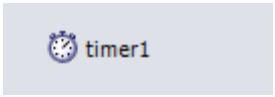
<table>
<tr>
<td> 

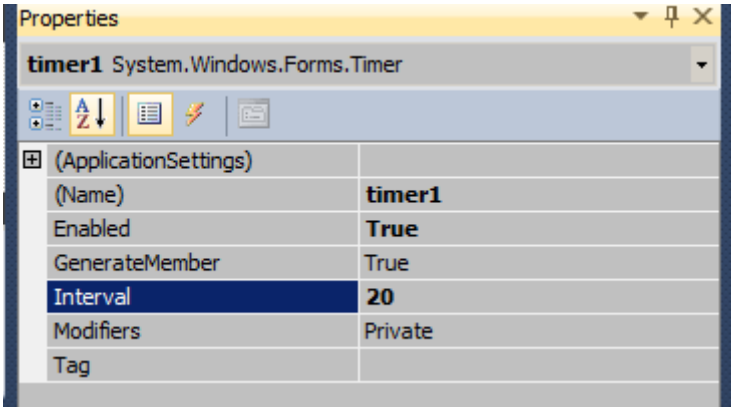

</td>
<td>Font - click on the 3 dots and change the size to 12 and BOLD
ForeColor = Black
BackColor = White</td>
</tr>
</table>

Lastly we need to add our main component the **TIMER**



Drag the timer to form





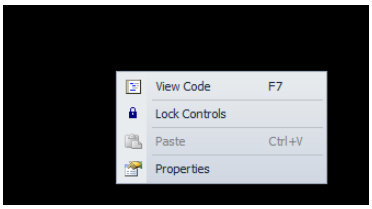
Change the following

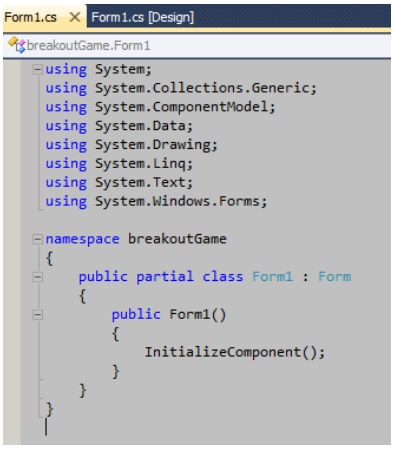
Enabled = true

Interval = 20

Now let's get started with the CODE yeaayy

First right click on the form and click on View Code






```
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 breakoutGame
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

This is the basic code which is the template for every windows

```
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 breakoutGame
{
    public partial class Form1 : Form
    {
        bool goRight;
        bool goLeft;
        int speed = 10;

        int ballx = 5;
        int bally = 5;

        int score = 0;

        private Random rnd = new Random();


        public Form1()
        {
```

```
            InitializeComponent();
        }
    }
}
```

Look carefully at how the code is laid out. We made some space on top of the form1() line and entered these variables. All of these variables are declared outside any function so they are classed as global variables.

bool goRight is a Boolean which is set to false by default. We will set it to true when the player press the right button.

bool goLeft is a Boolean which is set to false as before, this will also be set to True when player presses the left button. Both of these Booleans will turn back to false when player releases the said buttons left or right.

int ballx is a integer with the value of 5.

int bally is a integer with the value of 5. Both of these integers are used to move the ball in the X or Y axis in the windows form.

int score is the integer which will hold out total score for the game. This integer will add 1 to itself when we hit a block and take it off the screen.

```
private Random rnd = new Random();
```

This line above is creating a new instance of the CLASS, we have created a **rnd** variable which will be used to create a random number. We will be using this RND variable to select a random colour for the blocks. See below

```
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 breakoutGame
{
    public partial class Form1 : Form
    {
        bool goRight;
        bool goLeft;
        int speed = 10;

        int ballx = 5;
        int bally = 5;

        int score = 0;

        public Form1()
        {
            InitializeComponent();

            foreach (Control x in this.Controls)
            {
                if (x is PictureBox && x.Tag == "block")
                {
                    Color randomColor = Color.FromArgb(rnd.Next(256), rnd.Next(256), rnd.Next(256));
                    x.BackColor = randomColor;
                }
            }
        }
    }
}
```

Inside the Form1 function we are going to run a for each loop which will check the components used in this application. We are able to do various things with for each loop in this case we are looking for picture boxes with the tag of BLOCK.

Start from the top

```
foreach (Control x in this.Controls)
```

this is the first declaration of the for each loop. We are giving it a variable called X with the type of control. This will help us identify any component of windows form.

We always start with the open curly bracket after the condition of the loop {

```
if (x is PictureBox && x.Tag == "block")
```

Above we are starting the IF statement to find the picture box for instance we are saying if x is a type of picture AND x has the tag of block then do the following. Once again we start the IF statement with the open curly bracket {

```
Color randomColor = Color.FromArgb(rnd.Next(256), rnd.Next(256), rnd.Next(256));
```

we are creating a local variable called **randomColor** and then we are using the colour properties to randomly choose from a range of RGB (red, green and blue) elements with our RND variable.

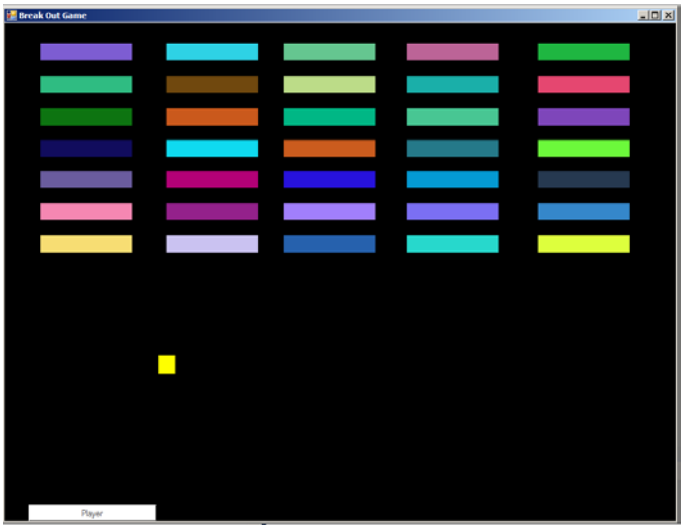
```
x.BackColor = randomColor;
```

This line assigning a single colour to each x -> picture box found on the form.

Let's not forget to end the IF statement curly brackets }

and the for loop curly brackets }

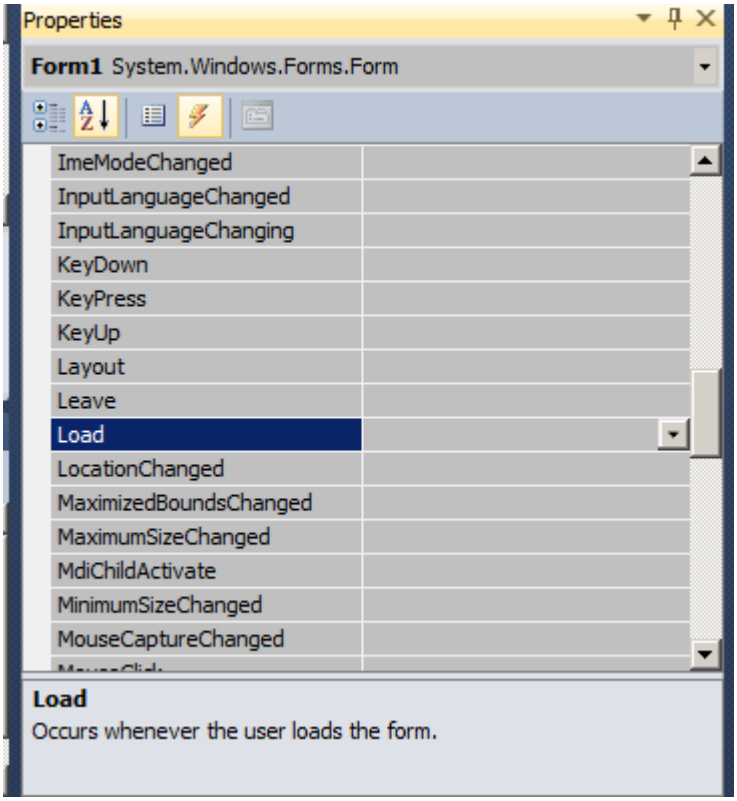
Run the game now.



Looking good.

Let go back to for the form and add the key down and key up events

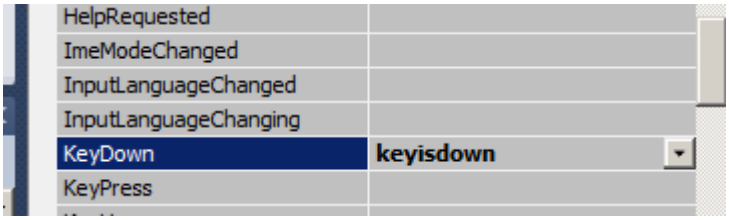
After you have clicked on the Form look at the properties window. Click on the little lightning bolt icon and you will see the following options



Here you add various different events to the form to create your up application.

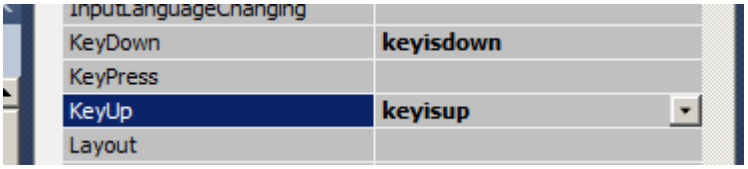
The two events we want are key down and key up.

Let's find the key down first.



Type **keyisdown** (no spaces) and press enter.

you will go back to the code view. Come back to the form view and now find the key up.



type **keyisup** and press enter.

Let's find the key is down function and enter the code inside of it.

```
        private void keyisdown(object sender, KeyEventArgs e)
        {
            //if the player pressed the left key AND the player is inside the panel
            // then we set the car left boolean to true
            if (e.KeyCode == Keys.Left && player.Left > 0)
            {
                goLeft = true;
            }
```

```
            // if player pressed the right key and the player left plus player width is less then the
panel1 width
            // then we set the player right to true
            if (e.KeyCode == Keys.Right && player.Left + player.Width < 920)
            {
                goRight = true;
            }
        }
```

```
if (e.KeyCode == Keys.Left && player.Left > 0)

            {

                goLeft = true;

            }
```

The code reads that if key code is equals to LEFT AND player button left is greater than 0 then we change the go left to true. This statement stops our player from leaving the form. We always want the player to stay within the playground.

```
if (e.KeyCode == Keys.Right && player.Left + player.Width < 920)

            {

                goRight = true;

            }
```

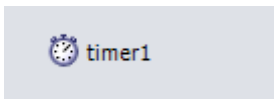
This if statement is for the right key. The same as the left key but a little different. IF the key code is equals to RIGHT AND player left plus the width which gives us the full width to the right of the button is less than 920 which is the width of our form then we change the go right to true. This way the player cannot leave the form to the right either.

Now find the key is up function and enter the code below

```
        private void keyisup(object sender, KeyEventArgs e)
        {
            // if the LEFT key is up we set the car left to false
            if (e.KeyCode == Keys.Left)
            {
                goLeft = false;
            }
            // if the RIGHT key is up we set the car right to false
            if (e.KeyCode == Keys.Right)
            {
                goRight = false;
            }
        }
```

In the function above once the player releases the left or right key we change the go left and right back to false. So the player button cannot move when the button isn't pressed.

Go back to the form design view and double click on the timer



This will automatically create a link for the timer. This is the most important part.

Let's see how it works with simple 2 lines of code to start off.

```
        private void timer1_Tick(object sender, EventArgs e)
        {
            ball.Left += ballx;
            ball.Top += bally;
        }
```

ball.Left += ballx; This line of code will move the ball to the left each time the timer ran. ball.Top += bally will move the ball downwards each time the timer runs.

If you run the game now. The yellow square will disappear in the bottom of the screen.

This will be a long function all of the codes are explained on the right side of the table.

`private void timer1_Tick(object sender, EventArgs e)` `{`	This is the top part of the function. This gives the name of the function and the object which is bound to it. In this case its linked to the timer on the form.
`ball.Left += ballx;`	We discussed this earlier it will move the ball left.
`ball.Top += bally;`	We discussed this line was discussed earlier too.
`label1.Text = "Score: " + score;`	This line is calling the label1 we added to the form and we will dynamically change the text as the score increased throughout the same. We have inserted the score variable which will increase each time a block is taken out.
`if (goLeft) { player.Left -= speed; } // move left`	This line is checking if the go left variable is true if so then we allow the user to move left by stating -= minus equals to the speed variable. Doing minus equals moves it to the left.
`if (goRight) { player.Left += speed; } // move right`	This line is checking if the go right is equals to true, if so it will do a plus equals to speed meaning it will add the speed towards the right and move the button to the right.

`if (player.Left < 1)`
`{`
`goLeft = false; // stop the car from going off screen`
`}`
	Although we have mentioned that we don't want the player to leave the scene, sometimes it will still allow the player to somewhat overlap the form and leave the scene, however we can put some code in the timer to stop them from moving out. This line is checking if the players left is less than 1 then we change the go left to false.

`else if (player.Left + player.Width > 920)`
`{`
`goRight = false;`
`}`
	Same again for this one, we are checking is the player left plus the player width is greater than 920 meaning the right side of the form then we change it back to false. Thus the player paddle doesn't move out of the scene.

`if (ball.Left + ball.Width > ClientSize.Width || ball.Left < 0)`
`{`
`ballx = -ballx;`
`// this will bounce the object from the left or right side`
`}`
	This if statement is multi condition. They don't have to be true meaning it can be one OR the other. IF ball left plus the ball width is greater than the form (Client Size) width then OR (||) ball left is less than 0 meaning its on the edge of left side of the

	screen then we change the **ballx** from positive to negative. This will reverse the effect and bounce the ball off the sides.

`if (ball.Top < 0 || ball.Bounds.IntersectsWith(player.Bounds))`
`{`
`bally = -bally; // this will bounce the object from top and bottom border`
`}`
	IF ball top position is greater than 0 meaning if it hits the TOP of the form OR (||) it intersects with the paddle (player) then we can change the bally from positive to negative. This is bounce the ball off the paddle and from the top of the screen.

`if (ball.Top + ball.Height > ClientSize.Height)`
`{`
`gameOver();`
`}`
	In this if statement we are checking if the ball reaches bottom of the screen. If the player cannot reach the ball and misses it then we can run the gameOver function. Once a function is created we can run the function with its name only.

`foreach (Control x in this.Controls)`	As we did before with the colour changing bit we are going to use the same formula to detect the hit between the block and the ball. To begin we start with the for each loop again, we are giving it a variable x as controls.
`{` `if (x is PictureBox && x.Tag == "block")`	IF x is a type of picture box AND (&&) x has a tag of block
`{` `if (ball.Bounds.IntersectsWith(x.Bounds))`	Inside the main if we have set up another if statement, IF ball bounds intersects with bounds of X meaning the blocks then we do the following

`{`
`this.Controls.Remove(x);`
`bally = -bally;`
`score++;`
`}`
	First we remove the X which collided with the ball
	We change the balls direction from up to down by changing the force from positive to negative
	And we increase the score by 1. doing ++ sign means it will increase the original number with a number of we can also do score += 1 but both works the same.

`}` `}`	These are the left over curly brackets from the IF and for loop. Always make sure to end them using the closing curly brackets.

`if (score > 34)`
`{`
`gameOver();`
`MessageBox.Show("You Win");`
`}`
	We have total of 35 picture boxes on the scene. Each picture box is worth 1 point. So if our player has score over 35 meaning 35 then we can decide that they have won. Run the game over function and show a message box that states you win.

`}`	This is the ending curly bracket of the timer function. Do not forget about this.

We need to add the last function the game.

```
        private void gameOver()
        {
            timer1.Stop();
        }
```

The game over function will only stop the main timer so the whole game stops.

Full Code For the break out game made in C# with Visual Studio.

```
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 breakoutGame
{
    public partial class Form1 : Form
    {
        bool goRight;
        bool goLeft;
        int speed = 10;

        int ballx = 5;
        int bally = 5;

        int score = 0;

        private Random rnd = new Random();

        public Form1()
        {
            InitializeComponent();

            foreach (Control x in this.Controls)
            {
                if (x is PictureBox && x.Tag == "block")
                {
                    Color randomColor = Color.FromArgb(rnd.Next(256), rnd.Next(256), rnd.Next(256));
                    x.BackColor = randomColor;
                }
            }
        }

        private void keyisdown(object sender, KeyEventArgs e)
        {
            //if the player pressed the left key AND the player is inside the panel
            // then we set the car left boolean to true
            if (e.KeyCode == Keys.Left && player.Left > 0)
            {
                goLeft = true;
            }
            // if player pressed the right key and the player left plus player width is less then the
panel1 width
            // then we set the player right to true
            if (e.KeyCode == Keys.Right && player.Left + player.Width < 920)
            {
                goRight = true;
            }
        }

        private void keyisup(object sender, KeyEventArgs e)
        {
            // if the LEFT key is up we set the car left to false
            if (e.KeyCode == Keys.Left)
            {
                goLeft = false;
            }
            // if the RIGHT key is up we set the car right to false
            if (e.KeyCode == Keys.Right)
            {
                goRight = false;
            }
        }
```

```
        private void timer1_Tick(object sender, EventArgs e)
        {
            ball.Left += ballx;
            ball.Top += bally;

            label1.Text = "Score: " + score;

            if (goLeft) { player.Left -= speed; } // move left
            if (goRight) { player.Left += speed; } // move right

            if (player.Left < 1)
            {
                goLeft = false; // stop the car from going off screen
            }
            else if (player.Left + player.Width > 920)
            {
                goRight = false;
            }
            if (ball.Left + ball.Width > ClientSize.Width || ball.Left < 0)
            {
                ballx = -ballx; // this will bounce the object from the left or right border
            }

            if (ball.Top < 0 || ball.Bounds.IntersectsWith(player.Bounds))
            {
                bally = -bally; // this will bounce the object from top and bottom border
            }

            if (ball.Top + ball.Height > ClientSize.Height)
            {
                gameOver();
            }
            foreach (Control x in this.Controls)
            {
                if (x is PictureBox && x.Tag == "block")
                {
                    if (ball.Bounds.IntersectsWith(x.Bounds))
                    {
                        this.Controls.Remove(x);
                        bally = -bally;
                        score++;
                    }
                }
            }

            if (score > 34)
            {
                gameOver();
                MessageBox.Show("You Win");
            }
        }

        private void gameOver()
        {
            timer1.Stop();
        }

    }
}
```

Remember to follow this tutorial accurately, at any time if you think there was a mistake go back to the steps.