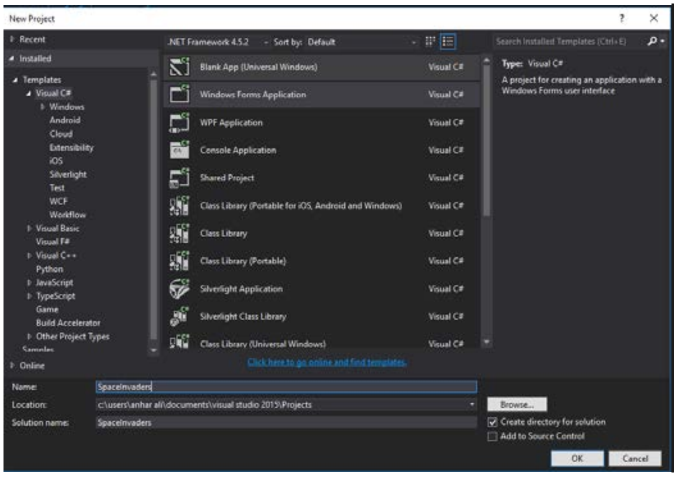
Start New Project In Visual Studio – Choose C# Windows Form Application – Name it **SpaceInvaders** and Click OK.


New Project

Recent | Installed | Templates | Visual C# | Windows | Android | Cloud | Extensibility | iOS | Silverlight | Test | WCF | Workflow | Visual Basic | Visual F# | Visual C++ | Python | JavaScript | TypeScript | Game | Build Accelerator | Other Project Types | Samples | Online

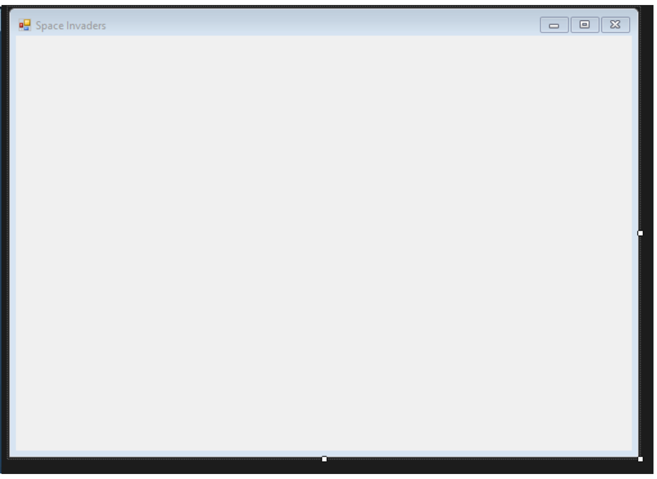
.NET Framework 4.5.2 – Sort by: Default

Blank App (Universal Windows)	Visual C#
Windows Forms Application	Visual C#
WPF Application	Visual C#
Console Application	Visual C#
Shared Project	Visual C#
Class Library (Portable for iOS, Android and Windows)	Visual C#
Class Library	Visual C#
Class Library (Portable)	Visual C#
Silverlight Application	Visual C#
Silverlight Class Library	Visual C#
Class Library (Universal Windows)	Visual C#

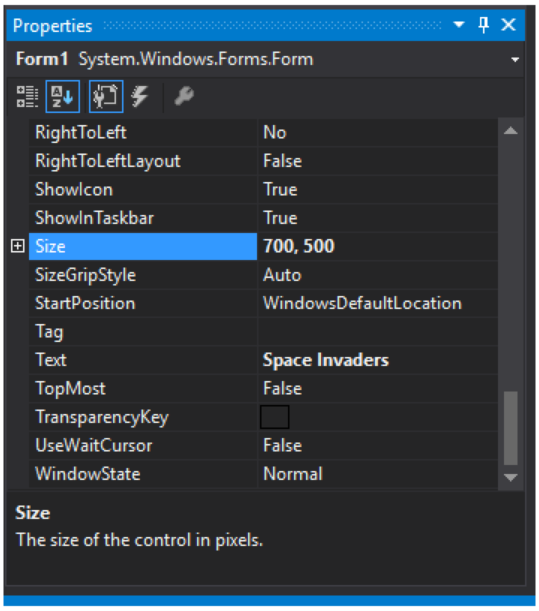
Click here to go online and find templates.

Type: Visual C#
A project for creating an application with a Windows Forms user interface

Name: SpaceInvaders
Location: c:\users\anhar ali\documents\visual studio 2015\Projects
Solution name: SpaceInvaders
[✓] Create directory for solution
[ ] Add to Source Control
OK Cancel



Click on the empty form and apply the following options to the properties Windows.


Properties
Form1 System.Windows.Forms.Form

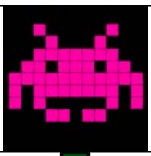
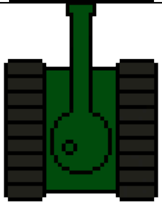
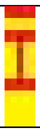
RightToLeft	No
RightToLeftLayout	False
ShowIcon	True
ShowInTaskbar	True
Size	**700, 500**
SizeGripStyle	Auto
StartPosition	WindowsDefaultLocation
Tag	
Text	**Space Invaders**
TopMost	False
TransparencyKey	
UseWaitCursor	False
WindowState	Normal

Size
The size of the control in pixels.

This is the properties window; this window is available for all the visual studio components.

Change size to **700, 500** and Text to **Space Invaders**

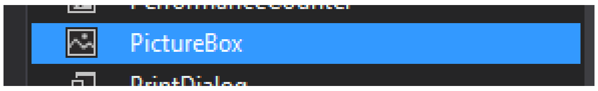
These are images we will be using for the game. The basic idea of the space invaders game is to have multiple enemies on top of the screen and they will gradually come down on the screen. The player will use the space bar to shoot the enemies on the top. Player will also be able to move the TANK image left or right of the form.

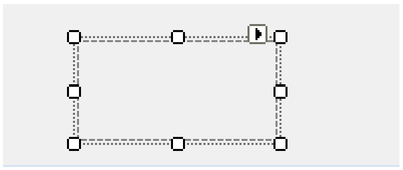
	Invader Image
	Tank Image for the plaer
	Bullet image

more tutorials on **www.mooict.com**

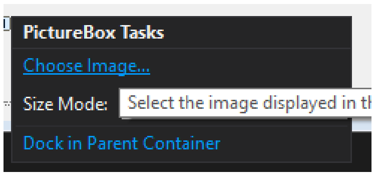
Now lets add the main components of the game. We will use Picture Boxes, Label and Timer for this game.

First add a picture box to the form.

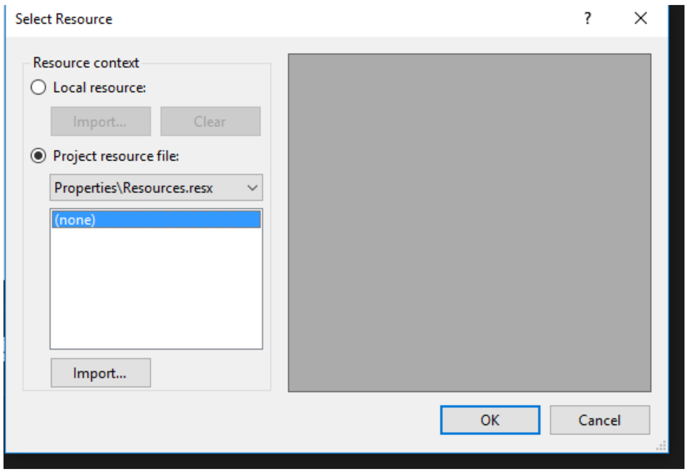




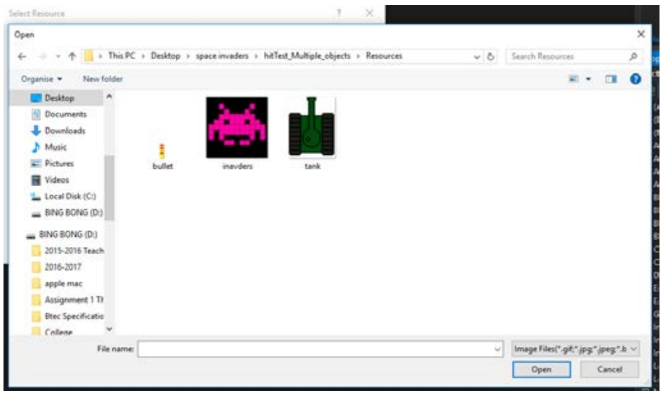
Click on the little triangle on the top of the picture



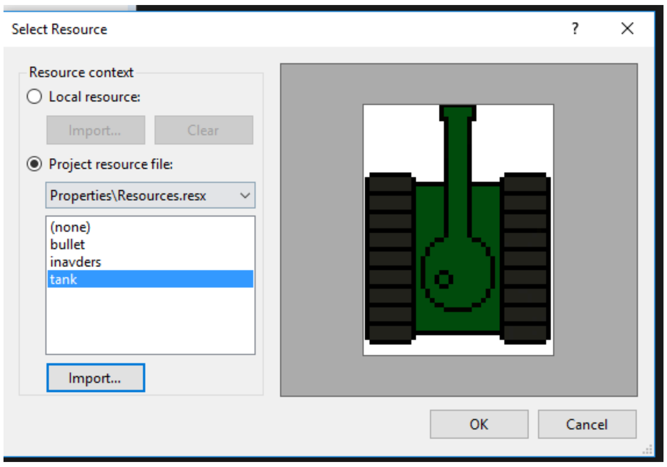
Click on choose image



Tick on the project resource file and click on import

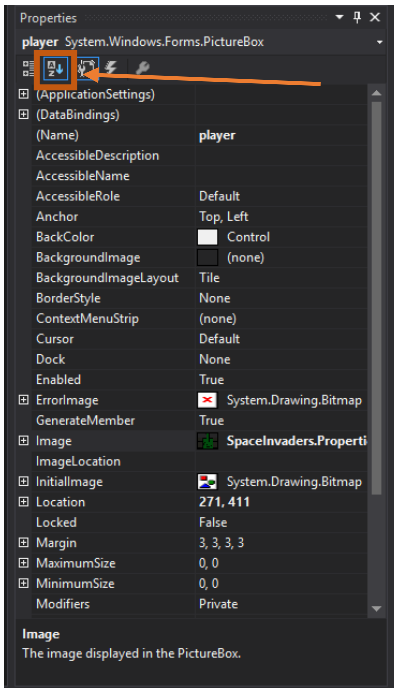


Select all of them and click on open



Select the Tank for the picture box and click OK

Apply the following changes to the properties window below -



player System.Windows.Forms.PictureBox

⊞ (ApplicationSettings)	
⊞ (DataBindings)	
(Name)	**player**
AccessibleDescription	
AccessibleName	
AccessibleRole	Default
Anchor	Top, Left
BackColor	⬜ Control
BackgroundImage	⬛ (none)
BackgroundImageLayout	Tile
BorderStyle	None
ContextMenuStrip	(none)
Cursor	Default
Dock	None
Enabled	True
⊞ ErrorImage	System.Drawing.Bitmap
GenerateMember	True
⊞ Image	**SpaceInvaders.Properti**
ImageLocation	
⊞ InitialImage	System.Drawing.Bitmap
⊞ Location	**271, 411**
Locked	False
⊞ Margin	3, 3, 3, 3
⊞ MaximumSize	0, 0
⊞ MinimumSize	0, 0
Modifiers	Private

Image
The image displayed in the PictureBox.

Make the changes in the properties window. If your properties window looks different than simply click on the A-Z button on top and it will organise it alphabetically.

(Name) = player

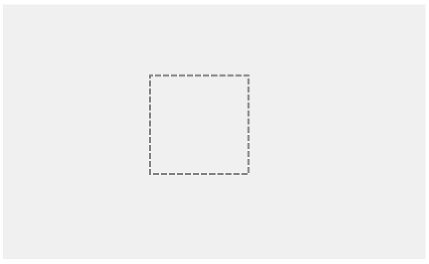
Location = 271, 411

Size = 50, 50

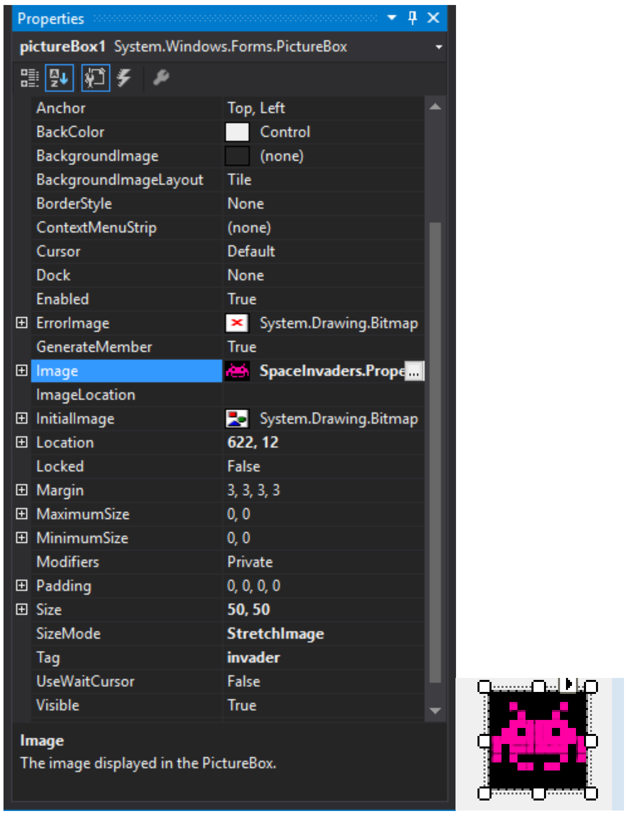
SizeMode = Stretch Image

Lets add one more picture boxes to the screen.

This one will be used for the invaders. We will be making multiple copies of it afterwards but for now we need to set up the initial space invaders object.



Change the following in the picture box 2 properties



Properties

pictureBox1 System.Windows.Forms.PictureBox

Anchor	Top, Left
BackColor	Control
BackgroundImage	(none)
BackgroundImageLayout	Tile
BorderStyle	None
ContextMenuStrip	(none)
Cursor	Default
Dock	None
Enabled	True
ErrorImage	System.Drawing.Bitmap
GenerateMember	True
Image	SpaceInvaders.Prope
ImageLocation	
InitialImage	System.Drawing.Bitmap
Location	**622, 12**
Locked	False
Margin	3, 3, 3, 3
MaximumSize	0, 0
MinimumSize	0, 0
Modifiers	Private
Padding	0, 0, 0, 0
Size	**50, 50**
SizeMode	**StretchImage**
Tag	**invader**
UseWaitCursor	False
Visible	True

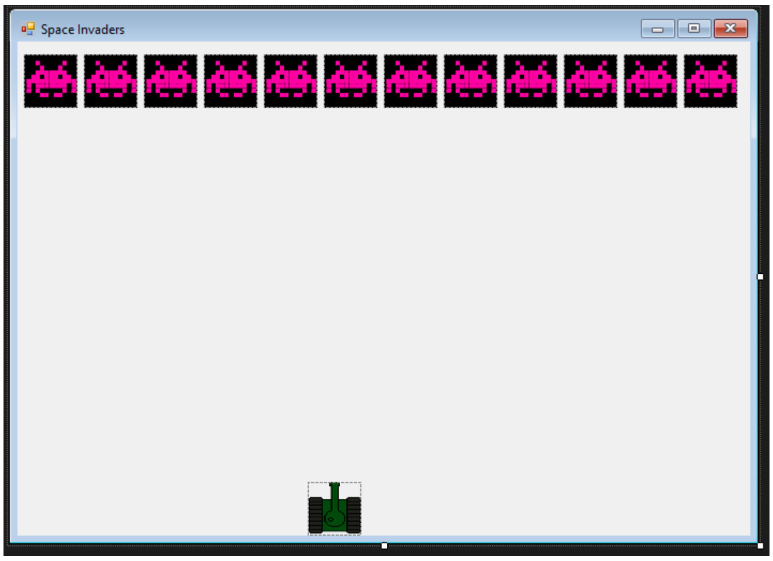
Image
The image displayed in the PictureBox.

Change the image to the space invaders image (use the same steps as you did before, but this time you will not need to import the images again, simply click on change images and it will have all 3 in the list.), **Location** to 622, 12, **Size** to 50, 50, **Size mode** to Stretch Image and **TAG** to invaders.

TAG is very important since we will have multiple invaders on the screen we will need to have a method to identify them in the game. So in this case we can use the TAG option. Do not miss out this step for this game.

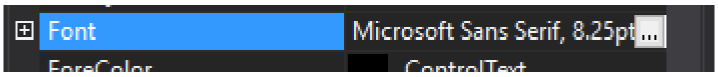
The bullet picture box we will create dynamically through the code so we don't need to add it the screen manually.

Since one of the space invaders has been set up we can now copy and paste it so it looks like this.

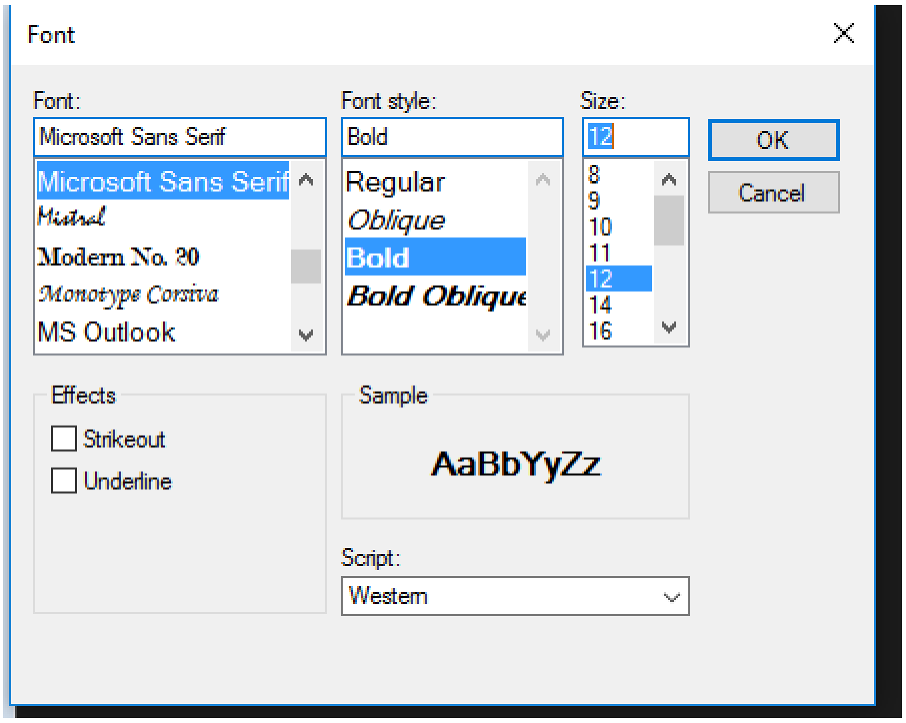


Now we have 12 invaders on the screen. They all have the same picture, size and tags.

Now we will need a label and a timer for the games GUI.


Font Microsoft Sans Serif, 8.25pt

Click on the … three dots button to the right

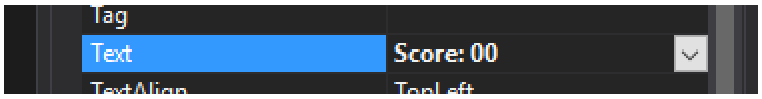

Font

Font: Microsoft Sans Serif

Font style: Bold

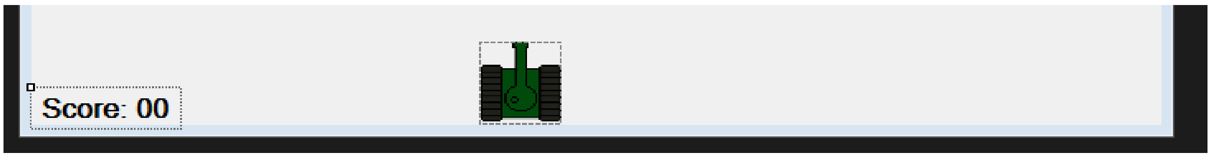
Size: 12

This box will come up, now change the font style to BOLD and size to 12.

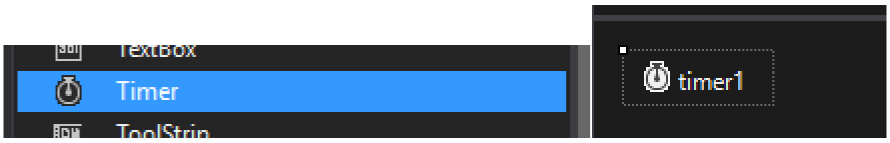


Text Score: 00

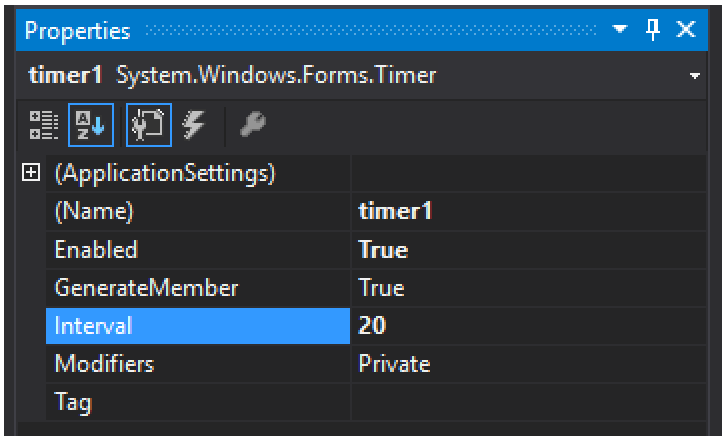
Change the text option in the properties window to Score: 00.



Place the label on the corner of the screen.



Find a timer object on the tool box and drag and drop it to the form.



Properties

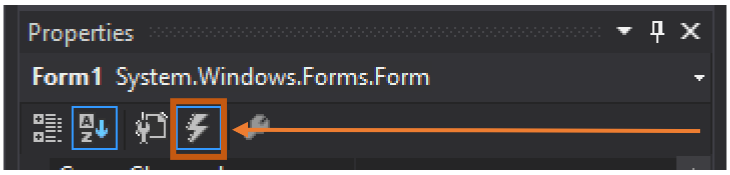
timer1 System.Windows.Forms.Timer

(ApplicationSettings)	
(Name)	timer1
Enabled	True
GenerateMember	True
Interval	20
Modifiers	Private
Tag	

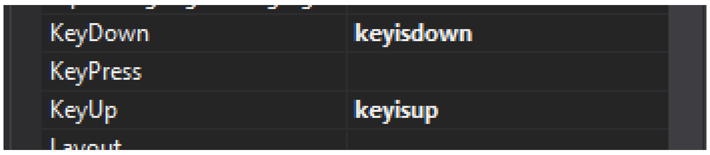
Make the following changes to it on the properties window

Enabled = True (meaning when the form starts running the timer will automatically start from the beginning)

Interval = 20 (all of the code inside the timer function will run every 20 milliseconds)



Click on the little lighting bolt near the form properties window. This is the events list window. We need to add two different events directly to the form. Key Down and Key Up.



KeyDown	keyisdown
KeyPress	
KeyUp	keyisup

Once you find them both type keyisdown for the Key down Event and press enter. This will take you into the code view, no problem come back to the design view and find the Key up event and type keyisup. Come back to the form design view and with those two events done lastly double click on the **timer object** on the form it will add its own function to the code.

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

namespace SpaceInvaders
{
public partial classForm1 : Form
    {

public Form1()
        {
            InitializeComponent();
        }

private void keyisdown(object sender, KeyEventArgs e)
        {

        }

private void keyisup(object sender, KeyEventArgs e)
        {

        }

private void timer1_Tick(object sender, EventArgs e)
        {

        }
    }
}
```

Above is the events and code so far all of which are entered by visual studio. We have our **keyisdown** and **keyisup** also we have the **timer1_tick** function which was added after we have double clicked on the timer1 object on the form.

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

namespace SpaceInvaders
{
public partial classForm1 : Form
    {

bool goleft;

bool goright;

int speed = 5;

int score = 0;

bool isPressed;

int totalEnemies = 12;

int playerSpeed = 6;

public Form1()
        {
            InitializeComponent();
        }
private void keyisdown(object sender, KeyEventArgs e)
        {

        }

private void keyisup(object sender, KeyEventArgs e)
        {

        }

private void timer1_Tick(object sender, EventArgs e)
        {

        }
    }
}
```

These are the global variables we have entered in the code.

bool goleft – this is a Boolean which can only store true or false. The idea for this one is if the left is pressed then we change it to TRUE also we will set up its own IF Statement in the timer function to trigger the movement of the player. It helps us keep the code logical .

bool goright – this has the same purpose as go left but only with the right button.

int speed – this is a single integer which contains the value 5 inside it. This will be used to set the speed of the invaders on screen.

int score – this integer will contain the score. When we hit an invader we will get one point and then we will add that one point to this integer.

bool isPressed – this Boolean is created so we can stop players from spamming the shooting button. The idea is we want the player to only shoot once per pressing of the space bar. This Boolean will help us reduce the spamming of bullets on the screen.

int totalEnemies = this integer is holding how many enemies we have on the screen. Right now, we have 12 enemies so I set it to 12.

int playerSpeed = this integer holds the value 6. We want to move our player left to right at 6 pixels each time we press left or right.

```
private void keyisdown(object sender, KeyEventArgs e)
        {
if (e.KeyCode == Keys.Left)
            {
                goleft = true;
            }

if (e.KeyCode == Keys.Right)
            {
                goright = true;
            }
        }

private void keyisup(object sender, KeyEventArgs e)
        {
if (e.KeyCode == Keys.Left)
            {

                goleft = false;
            }

if (e.KeyCode == Keys.Right)
```

```
            {
                goright = false;
            }
        }
```

Notice the highlighted code above. In the key is pressed event once the left key is pressed we set the Boolean **goleft** to true and same for the **goright** Boolean.

In the **keyisup** event when the player released the left button we can change it back to false and same for the right button.

Notice we haven't put in the space key yet. Don't worry that will be coming soon.

```
private void timer1_Tick(object sender, EventArgs e)
        {
if(goleft)
            {
                player.Left -= playerSpeed;
            }
else if(goright)
            {
                player.Left += playerSpeed;
            }
        }
```

Lets look at the timer function Add the following code in to it.

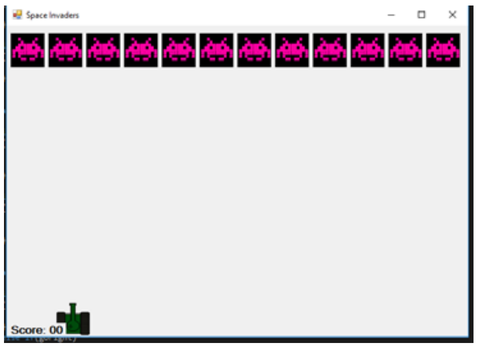
The highlighted code is checking whether the **goleft** or **goright** variable is turning true and if the are we allow the player move left or right.

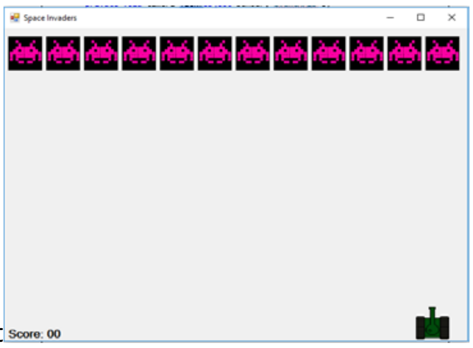
In C# every object has the option LEFT we can determine where the player is and from there we can change the position with a custom number or variable.

Left is -= minus equals to player speed. So the player will move closer to the left border

Right is += plus equals to player speed and the player object will move further away from the left border.

Try it.

Going Left 

Going Right 

Now we need to make an independent function called make bullet. The idea is when the space bar is pressed we will call this function and it will add a new bullet to the screen.

Lets make a new function called make bullet

```
privatevoid makeBullet()
        {
PictureBox bullet = new PictureBox();
```

more tutorials on **www.mooict.com**

```
            bullet.Image = Properties.Resources.bullet;

            bullet.Size = new Size(5, 20);

            bullet.Tag = "bullet";

            bullet.Left = player.Left + player.Width / 2;

            bullet.Top = player.Top - 20;

this.Controls.Add(bullet);

            bullet.BringToFront();

        }
```

Lets explain from the TOP.

```
privatevoid makeBullet()

        {
```

This is where the function starts, now we need to make sure that its not inside any other function. Make sure there is } final curly brackets to determine where the function ends.

`PictureBox bullet = new PictureBox();`This line is creating a new instance of the Picture Box class. We are basically making a new virtual bullet picture box.

`bullet.Image = Properties.Resources.bullet;` The bullet image which we have imported into the resources earlier is being called here. Once the bullet is created we are assigning the bullet image for the picture box.

`bullet.Size = new Size(5, 20);` Now we are giving this bullet a size. We want it to have 5 width and 20height. That's the size of the image.

`bullet.Tag = "bullet";` We are getting the tag for the bullet as bullet. This is very important because we will be searching tags to find it when the game is running.

`bullet.Left = player.Left + player.Width / 2;` When the space is pressed we want to the bullet to appear middle of the tank. So we are calculating the left of the player and the width then divide that by 2. This will give us the dead centre of the picture box.

`bullet.Top = player.Top - 20;` Same as the left we are setting up the top of the picture to appear a little over the tank and not on it. So we ware reducing 20 pixels from the top of the bullet.

`this.Controls.Add(bullet);` we are calling the main parent of the form which is THIS.CONTROLS it will automatically add the picture box to the form.

`bullet.BringToFront();` now since it's a picture box it can sometimes get behind other picture box on the form. Lets use Bring to Front function to bring it front of other objects on the screen.

Finally we use the last } to end the function

Now we need to make the game over function.

Make another function called game over

```
private void gameOver()
        {
            timer1.Stop();
            label1.Text += " Game Over";
        }
```

in the function above its going to stop the timer and then show game over on the label 1.

Look back at the key down function and add the following

```
privatevoid keyisdown(object sender, KeyEventArgs e)
        {
if (e.KeyCode == Keys.Left)
            {
                goleft = true;
            }

if (e.KeyCode == Keys.Right)
            {
                goright = true;
            }
if (e.KeyCode == Keys.Space && !isPressed)
            {
                isPressed = true;

                makeBullet();
            }
        }
```

We have discussed the key is down function before but now we entered the space button functions inside it. Lets see IF the space bar is pressed AND (&&) is pressed Boolean is false. We are using a short hand code to specify when the Boolean is false. For example if it was shown as if(isPressed) { } this means it will run only when the Boolean is true, however if we put an exclamation mark in front it then it will only run when the statement if salse !isPressed. We can also do it like this
If (isPressed == false) { }.
Once the button is pressed and the conditions are met we then change the false isPressed to true and run the makeBullet() function.

```
privatevoid keyisup(object sender, KeyEventArgs e)
        {
if (e.KeyCode == Keys.Left)
            {

                goleft = false;
            }

if (e.KeyCode == Keys.Right)
            {

                goright = false;
            }

if (isPressed)
            {
                isPressed = false;
            }
        }
```

In this function we are checking when the keys are if isPressed Boolean is true then we simply change it back to false. This will stop the players from spamming the space bar by holding it down and throwing our infinite bullets.

```
private void timer1_Tick(object sender, EventArgs e)
        {

//player moving left and right
if(goleft)
            {
                player.Left -= playerSpeed;
            }
elseif(goright)
            {
                player.Left += playerSpeed;
            }
// end of player moving left and right


//enemies moving on the form
foreach (Control x in this.Controls)
            {

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

In the highlighted code on the left we have entered the new features of in the timer function.

First we are running a foreach loop. Inside the loop we are giving the condition to loop through all of the controls in the form.
The loop always starts with the open curly brackets {
Then we are stating an if statement, in the statement we are looking for x variable which we declared on the loop earlier to see is X is a type of picture box and it has the tag of "invader". Remember when it was mentioned the picture boxes tags are important, this is why.

When we have found the picture boxes tagged invaders then we need to check if they collide with the player if so then we can call the game over function.

```
if
(((PictureBox)x).Bounds.IntersectsWith(player.Bounds))
                    {
                        gameOver();

                    }

                    ((PictureBox)x).Left += speed;

if (((PictureBox)x).Left > 720)
                    {
                        ((PictureBox)x).Top +=
((PictureBox)x).Height + 10;

                        ((PictureBox)x).Left = -50;
                    }
                }
            }
// end of enemies moving on the form

        }
```

`((PictureBox)x).Left += speed;`
This line allows the picture boxes (invaders) to move to the right when the game starts

`if (((PictureBox)x).Left > 720)`
If the picture boxes are over the edge on the right meaning if the picture boxes left property is more 720 pixels

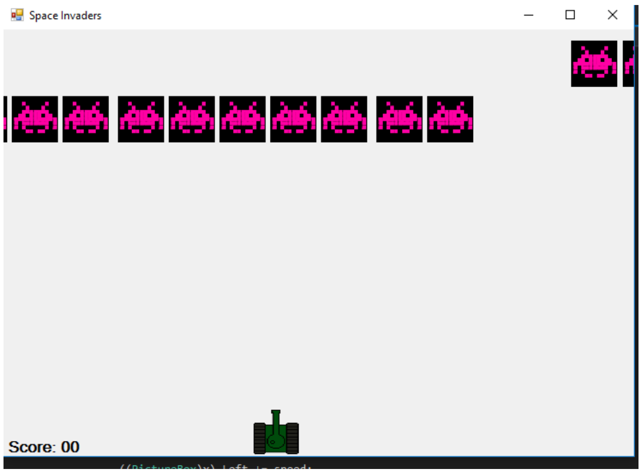
`((PictureBox)x).Top += ((PictureBox)x).Height + 10;`

Then we can move them down by their height
`((PictureBox)x).Left = -50;`
And reset their position off the screen to the left so they can come back again.

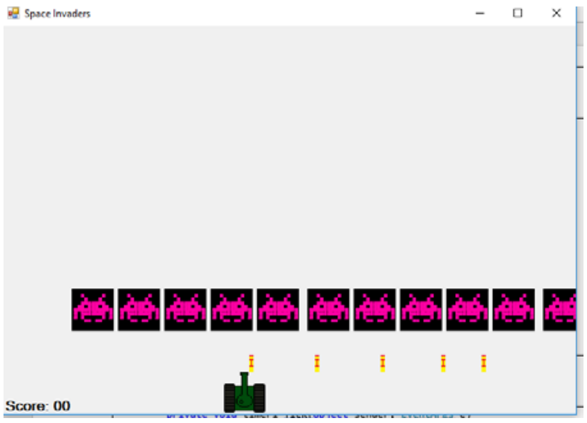
Make sure you count the closing curly brackets because if have opened a main for loop, initial if and the if inside that one so there should be 3 closing brackets minus the closing brackets for timer function.

Run the game now see what happens

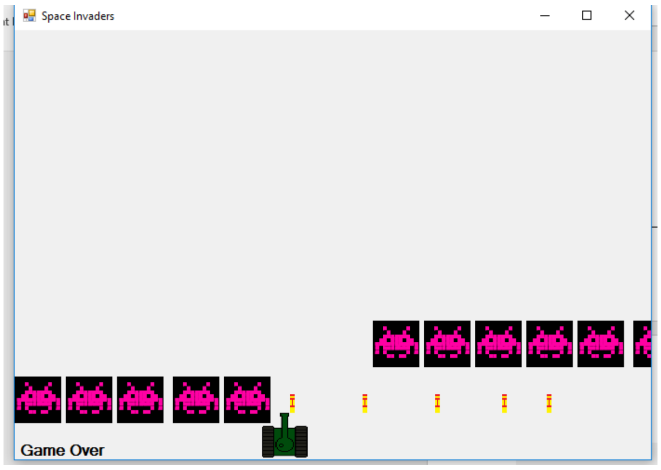
Enemies moving down



Bullets are appearing when the space bar in pressed, but they are not moving up. We will add that function later.



When the enemy picture boxes bump into the player picture box the game over is coming up and the whole game stops.

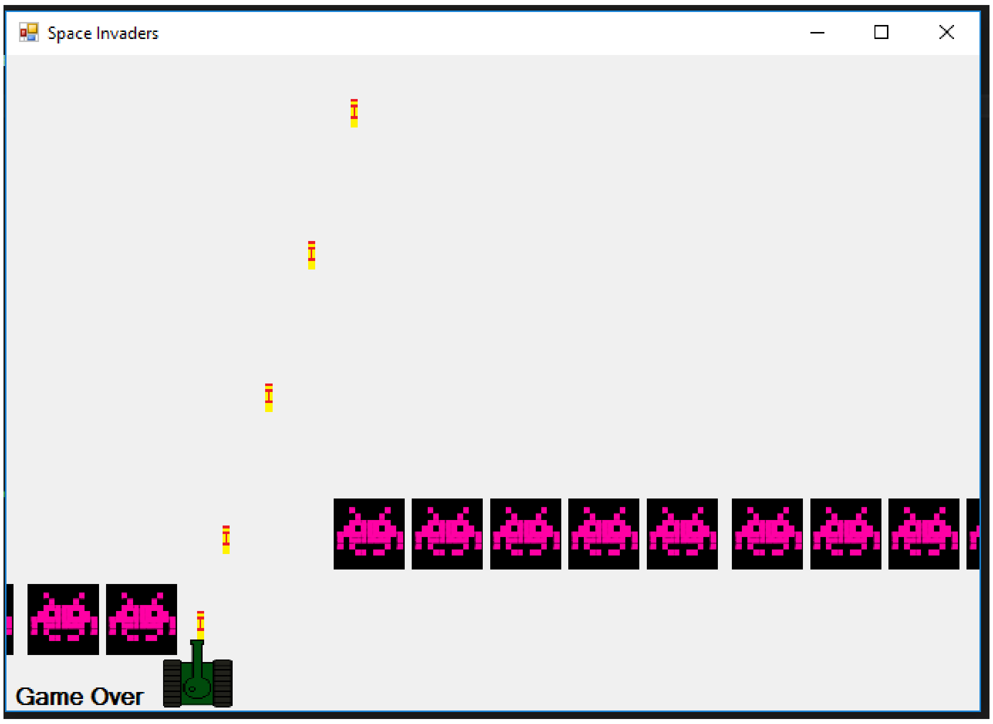


So far so good for us.

Now we need to animate the bullets going up when they are fired.

`//animating the bullets and removing them when they have left the scene`
`foreach (Control y in this.Controls)`
`{`
`if (y is PictureBox && y.Tag == "bullet")`
`{`
`y.Top -= 20;`
`if (((PictureBox)y).Top < this.Height - 490)`
`{`
`this.Controls.Remove(y);`
`}`
`}`
`}`
`// end of animating the bullets.`
	In the same timer function, under the animation loop we will enter a second loop which will identify, animate and dispose the bullets from the screen.
	
	`foreach (Control y in this.Controls)`
	Once again we run the for loop in this case we can use y instead of x.
	`if (y is PictureBox && y.Tag == "bullet")`
	This timer we are checking if y is a picture box and its tag is bullet.
	
	If so
	`y.Top -= 20;`
	We can animate the bullet by moving it up by 20 pixels per trigger.
	
	`if (((PictureBox)y).Top < this.Height - 490)`
	If the bullet reaches top of the form
	
	`this.Controls.Remove(y);`
	We can safely remove it from the form.

Try running the code now with the new loop inside it.



Fire away. Bullets are moving up and they are disappearing when they reaches top of the form.

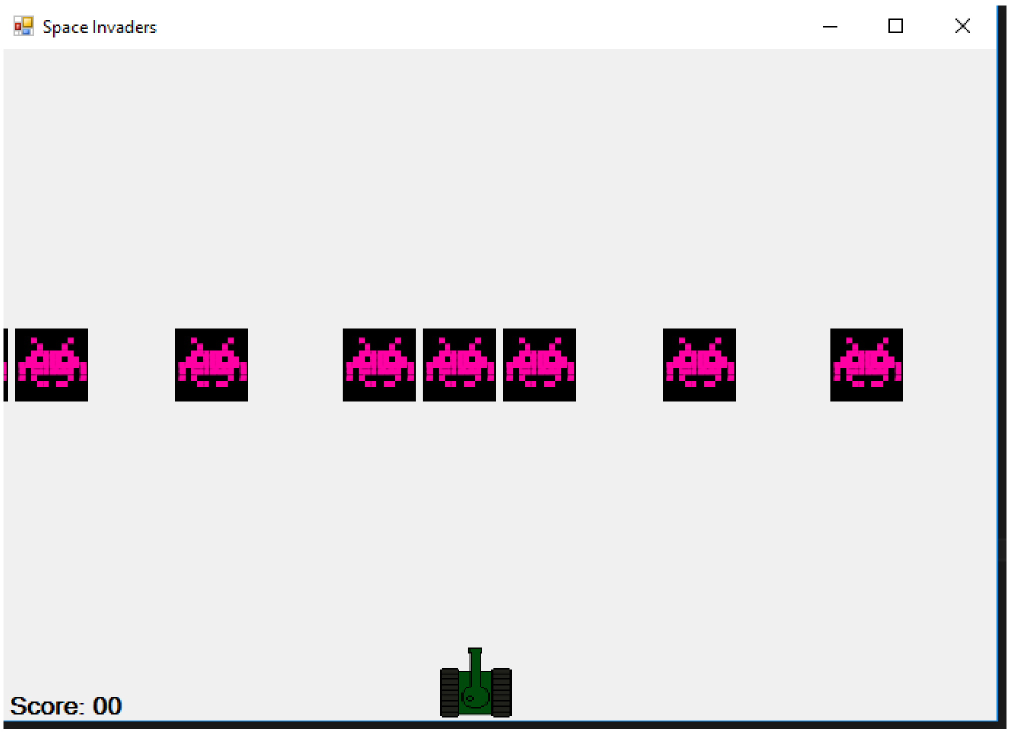
```
// bullet and enemy collision start
foreach (Control i in this.Controls)
            {
foreach (Control j in this.Controls)
                {
if (i is PictureBox && i.Tag == "invader")
                    {
if (j is PictureBox && j.Tag == "bullet")
                        {

if (i.Bounds.IntersectsWith(j.Bounds))
                            {
                                score++;
this.Controls.Remove(i);
this.Controls.Remove(j);
                            }
                        }
                    }
                }
            }
// bullet and enemy collision end
```

Now it gets a little complicated. In order for the bullet and invaders to collide we cannot use the **bullet.bounds.intersectsWith(invader.bounds)** Because they both exist in two different loops and we cannot interrupt from the outside of the loop.

`foreach (Control i in this.Controls)`
So we are creating a new loop this time calling I variable with the controls
{
`foreach (Control j in this.Controls)`
Then we are declaring another loop inside the one before after the opening curly brackets off course.
{
`if (i is PictureBox && i.Tag == "invader")`
Now we are identifying the invaders by checking if i is a type of picture box and then if they have the tag of invader.
{
`if (j is PictureBox && j.Tag == "bullet")`
{
now we are identifying the bullets by checking if j is a type of picture box and j has tag of bullet.
{
`if (i.Bounds.IntersectsWith(j.Bounds))`
Now we can check I which is the invaders if they collider with J which is the bullet
{
`score++;`
Then we increase the score by 1.
`this.Controls.Remove(i);`
Now we remove the i meaning we remove the invader which was hit.
`this.Controls.Remove(j);`
Now remove the bullet that hit the invader

	} this closing curly bracket is the bounds if statement } this closing curly bracket is for the bullet if statement } this closing curly bracket is the for invaders if statement } this closing curly bracket is for the bullet loop } this closing curly bracket is for the invader loop



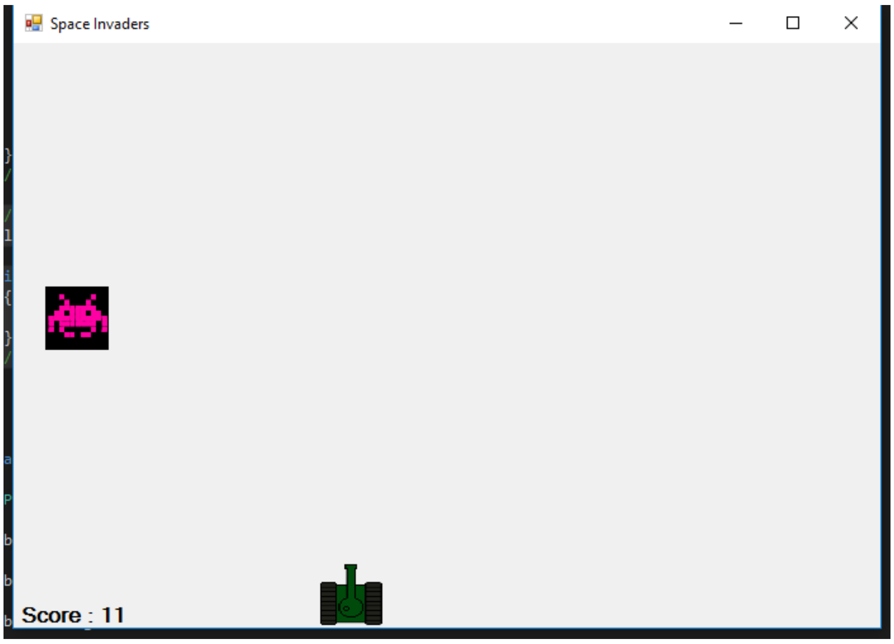
Ohh yea

```
// keeping and showing score
label1.Text = "Score : " + score;

if (score > totalEnemies - 1)
            {
                gameOver();
                MessageBox.Show("You Saved Earth");

            }
// end of keeping and showing score.
```
	We are almost at the end of the timer function.
	
	Here we are changing the label1 text with the score.
	
	In the if statement we are checking we killed all of the aliens from the screen then we can run the game over function.
	
	At the end it will show a message box that says you saved earth.

more tutorials on **www.mooict.com**



One more left, shoot this guy and win the game.

Full Code for Reference

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

namespace SpaceInvaders
{
public partial classForm1 : Form
    {

bool goleft;

bool goright;

int speed = 5;

int score = 0;

bool isPressed;

int totalEnemies = 12;

int playerSpeed = 6;


public Form1()
        {
            InitializeComponent();
        }

private void keyisdown(object sender, KeyEventArgs e)
```

```
        {
if (e.KeyCode == Keys.Left)
            {
                goleft = true;
            }

if (e.KeyCode == Keys.Right)
            {
                goright = true;
            }
if (e.KeyCode == Keys.Space && !isPressed)
            {
                isPressed = true;

                makeBullet();
// do work
            }
        }

private void keyisup(object sender, KeyEventArgs e)
        {
if (e.KeyCode == Keys.Left)
            {

                goleft = false;
            }

if (e.KeyCode == Keys.Right)
            {

                goright = false;
            }

if (isPressed)
            {
                isPressed = false;
            }
        }

private void timer1_Tick(object sender, EventArgs e)
        {

//player moving left and right
if (goleft)
            {
                player.Left -= playerSpeed;
            }
elseif (goright)
            {
                player.Left += playerSpeed;
            }
// end of player moving left and right


//enemies moving on the form
foreach (Control x in this.Controls)
            {

if (x is PictureBox && x.Tag == "invader")
                {
if (((PictureBox)x).Bounds.IntersectsWith(player.Bounds))
                    {
                        gameOver();

                    }
                    ((PictureBox)x).Left += speed;

if (((PictureBox)x).Left > 720)
                    {
                        ((PictureBox)x).Top += ((PictureBox)x).Height + 10;

                        ((PictureBox)x).Left = -50;
                    }
                }
            }
// end of enemies moving on the form


//animating the bullets and removing them when the have left the scene
foreach (Control y in this.Controls)
            {
if (y is PictureBox && y.Tag == "bullet")
                {
                    y.Top -= 20;

      if (((PictureBox)y).Top <this.Height - 490)
                    {
         this.Controls.Remove(y);
                    }
                }
            }
// end of animating the bullets.


// bullet and enemy collision start
foreach (Control i in this.Controls)
      {
foreach (Control j in this.Controls)
      {
if (i is PictureBox && i.Tag == "invader")
      {
if (j is PictureBox && j.Tag == "bullet")
      {

if (i.Bounds.IntersectsWith(j.Bounds))
      {
            score++;
            this.Controls.Remove(i);
            this.Controls.Remove(j);
                            }
                        }
                    }
                }
            }
// bullet and enemy collision end

// keeping and showing score
            label1.Text = "Score : " + score;

if (score > totalEnemies - 1)
            {
                gameOver();
                MessageBox.Show("You Win");
            }
// end of keeping and showing score.

        }



private void makeBullet()
```

```
        {
            PictureBox bullet = new PictureBox();

            bullet.Image = Properties.Resources.bullet;

            bullet.Size = new Size(5, 20);

            bullet.Tag = "bullet";

            bullet.Left = player.Left + player.Width / 2;

            bullet.Top = player.Top - 20;

            this.Controls.Add(bullet);

            bullet.BringToFront();

        }

private void gameOver()
        {
            timer1.Stop();
            label1.Text += " Game Over";
        }
    }
}
```