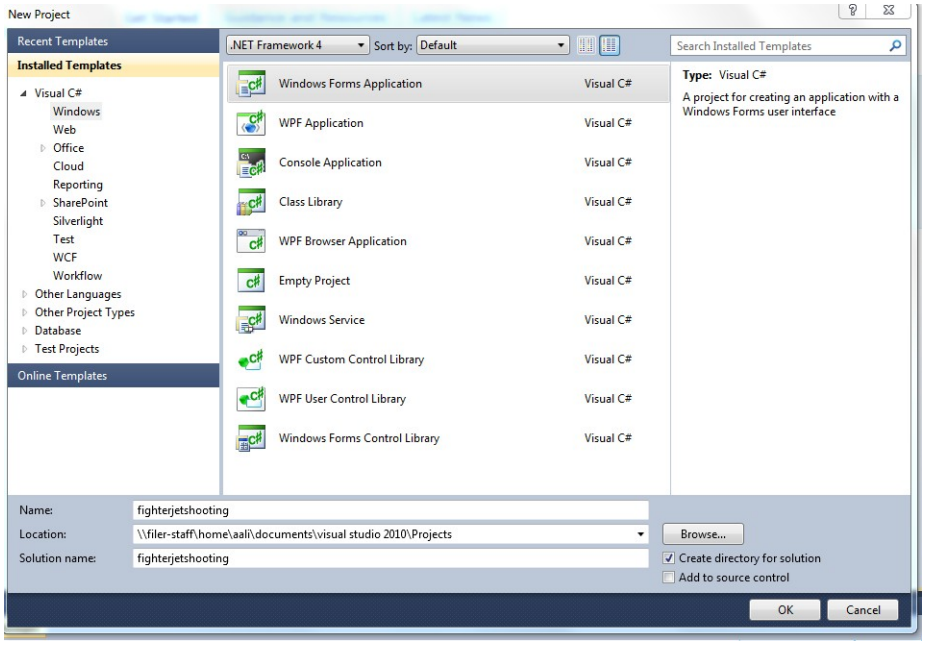
C# Tutorial – Fighter Jet Shooting Game

Welcome to this exciting game tutorial. In this tutorial we will be using Microsoft Visual Studio with C# to create a simple fighter jet shooting game. We have the images ready for you to use within the assets at MOOICT.COM however you can use your images if you please. Since we will be using positions and pixel perfect movements we would recommend you use our images to start with. In this tutorial you will be using keyboard to control the player and shoot at enemies. We will keep the total score of the game and show it on the screen.

Lesson objectives –

1. Create a simple keyboard controlled game using keyboard events
2. Use of Booleans, strings and integers
3. Using bounds and hit test in C#
4. Calculating two objects colliding with each other
5. Removing images from display and re animating them
6. Keeping score in the game
7. Illustrating shooting bullets
8. Using the timer event to animate all objects on the screen
9. Using multiple enemies in the display
10. End the game and reset all components
11. Show the score and reset the game for the player to play again

Let's get our Visual Studio ready for this exciting project.
Start a new C# Windows Form Application name it FighterJetShooting.

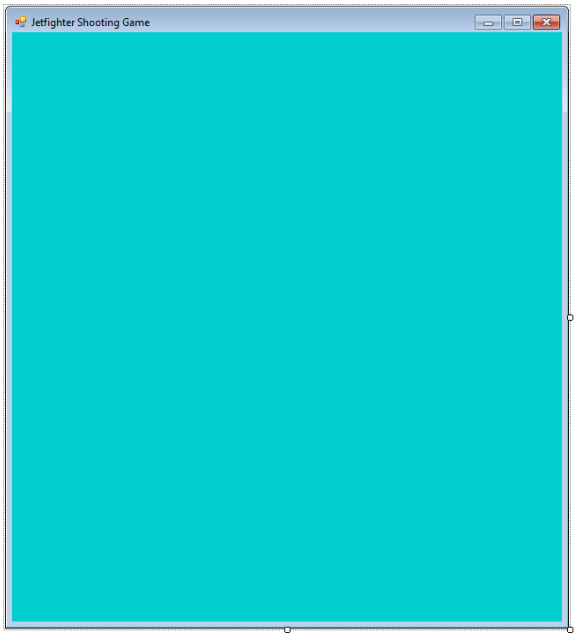


Lets set up the form

Text – **Fighter Jet Shooting Game**

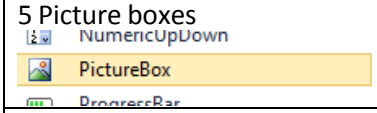
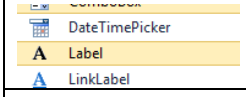
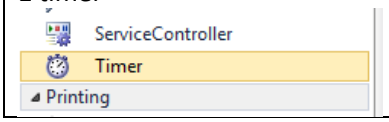
Size - **640, 706**

Back Colour – **DarkTurquoise**

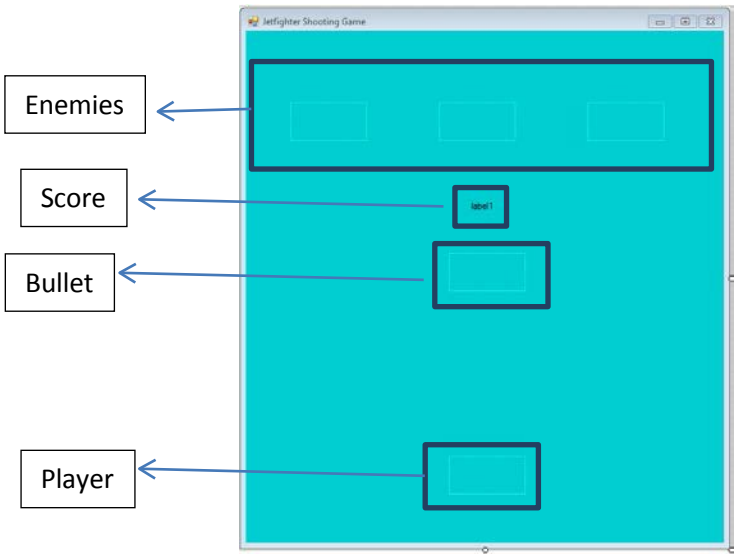


This will be the display background of our game.

Components we will need for this game

Component	Purpose
5 Picture boxes 	3 Enemies 1 Bullet 1 Player
1 Label 	Showing the score on screen
1 timer 	This will control all of the movements and animation of the game.

Let's set them up on the screen.



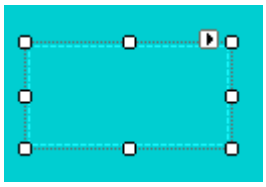
Here we have all of our assets set up. We will need to resize them accordingly.

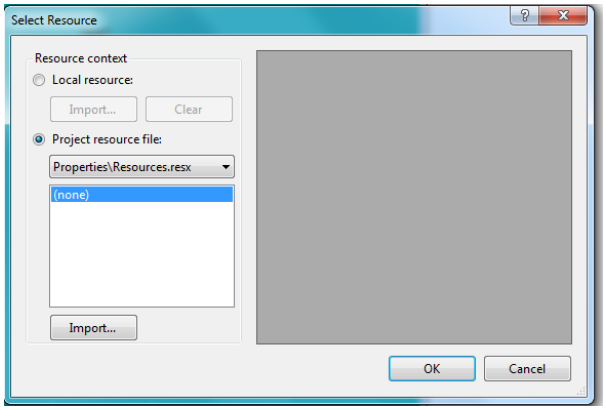
Let's start with the pictures. We will need to import all the assets we prepared and size them so they don't look out of place.

		
This is the bullet image we will use in this game	This is the enemy image	This is the player

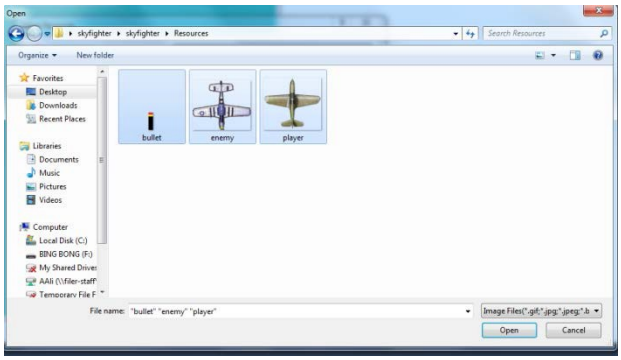
The 3 picture boxes will have the same enemy image. There will be only one player on screen and the rules of the game which we will go into more detail later but for now the player can only shoot a single bullet at a time. We don't want players spamming the space bar and get quick points. Use demskillz/ git gud or whatever ☺.

Now lets import these images into Visual Studio and link them to the picture boxes.

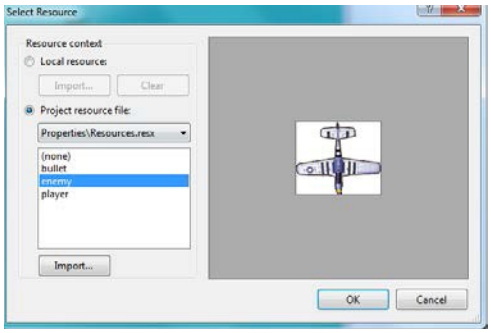

Click on a picture box and click on that little triangle on the corner it will give you options to choose image.



Now import those images into the resources



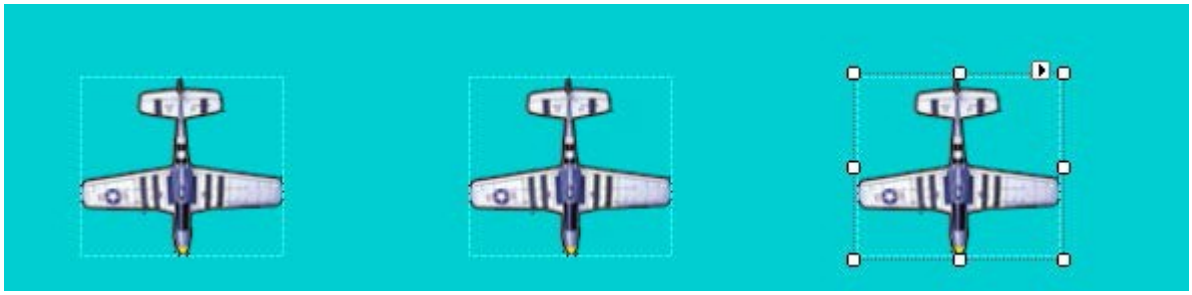
Click on open



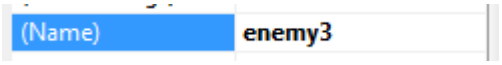
Now they are in the resources. You don't have to do this for each picture box we have imported all of them in the resources now and we can simply click on choose image again and pick the others as we go long.

For the enemies picture box change the following properties in the properties window.

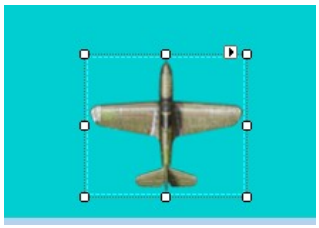
Change the size to 102, 90 and change the size mode to stretch image.



This is what they look like now. Pretty neat. The images are all PNG format therefore it has got its own transparency.

Inside the properties window Change the names of these picture boxes from left to right **enemy1**, **enemy2** and **enemy3**. 

Now the player picture box. First change the name to **player,** add the player image to it and size to 111, 97 and size mode to stretch image.

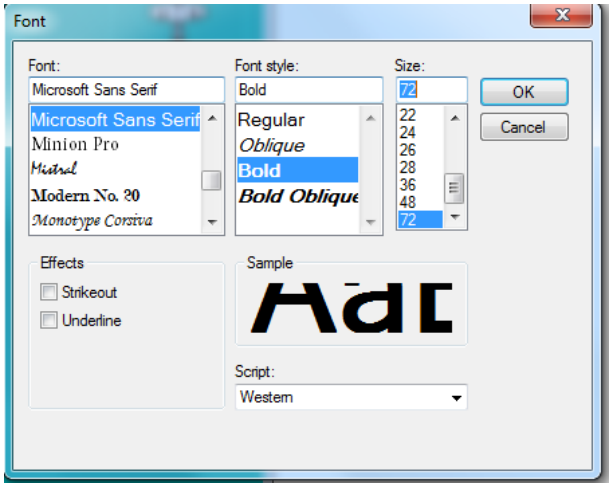


Now the bullet picture box. First change the name to **bullet,** add the bullet image to it and size to 7, 27 and size mode to stretch image.

Lastly the label.

We want the label to be seen in the background of the form. It should only update the player has hit an enemy on screen.

Click on the label, look in the properties window. Find the option for font, expand it and change to the following.

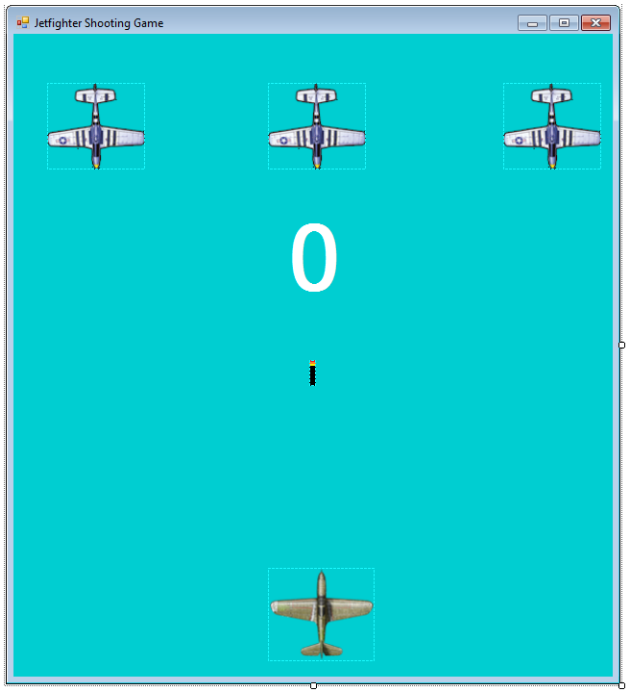


Also change the fore colour to White.

Change the label name to **scoreText**

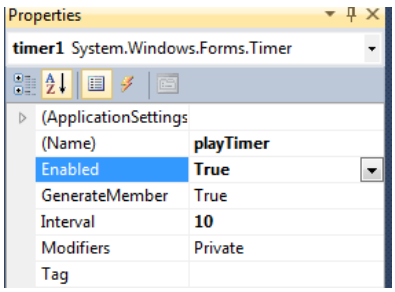
And text to **0**

Here is the final view of the Game so far.



At last we add the main ingredient for this game. We add our timer.

Drag and drop the timer to the form and change the properties to the following.



name – playTimer, enabled – true and interval - 10

Interval stands for how often the timer should trigger. We change it to 10 milliseconds which will give our game a rather good FPS. Since there aren't a lot of components on the screen it will run smoothly.

Now to start with the programming.

This is where we will need to break down the game logic and then get started on the programming task.

NEVER EVER blindly start programming you will be wasting valuable hours. Once you plan it carefully you can easily achieve your goals and it will be fewer headaches because you will know exactly where to go next.

To begin we will need the game to do the following

1. Player can move left and right
2. Player shoots bullet with SPACE
3. Player can shoot only 1 bullet at a time. (While the bullet is on screen player cannot shoot again)
4. Bullet travels straight up
5. Each enemy travels downward on the form
6. If bullet hit enemy they will respawn up and then travel down
7. If enemy reaches end of the form they will respawn up and then travel down
8. Each time the bullet hits the enemy it will add 1 to the score
9. The score will be updated on the label on screen

Go to the Code view in your program.

Right click on the form and click on View Code or Press F7.

```
//player moves
int moveLeft = 0;
int enemyMove = 5;
Random rnd = new Random();
int bulletSpeed = 8;
bool shooting = false;
int score = 0;
```

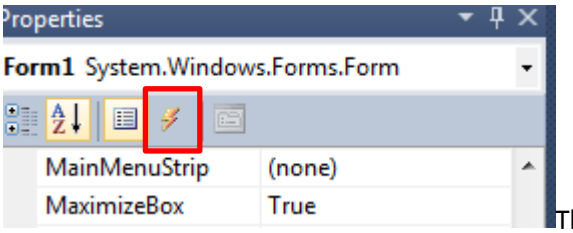
Above are the variables we will use in this game. Move left variable will determine where the character is going on the screen. Enemy move variable will set a speed for which the enemies will come down on the form. We need to generate a random number in order to respawn the player. The bullet will determine how fast the bullet will go. We are using a Boolean variable which will set whether the player is firing a bullet or not. Last variable is the score which will keep track of how many enemies we have terminated from the game.

Now we need to give a default value to our game elements such as the bullet and enemies.

```
public Form1()
        {
InitializeComponent();
enemy1.Top = -500;
enemy2.Top = -900;
enemy3.Top = -1300;
bullet.Top = -100;
bullet.Left = -100;
        }
```

The picture boxes in visual studio has the built in functions called TOP and LEFT which means we can place these items where we want on the form. Since this is the first function to run we want to remove the bullet and enemies from the display and then start the game. So here we are pushing the picture boxes away from the form only to make it look like they are naturally appearing on the screen.

Now go back to the Design View. CLICK on the form Look for this little ICON by the properties window.

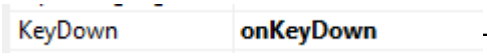


That's the events window. Make sure you have clicked on the form and then clicked on the event.

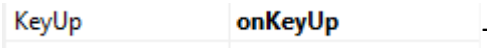
There are two events that we want.

1. Key Down Event
2. Key Up Event

Above are the both.



Type in **onKeyDown** for the key down event. Press Enter.



Type in **onKeyUp** for the key up event. Press Enter.

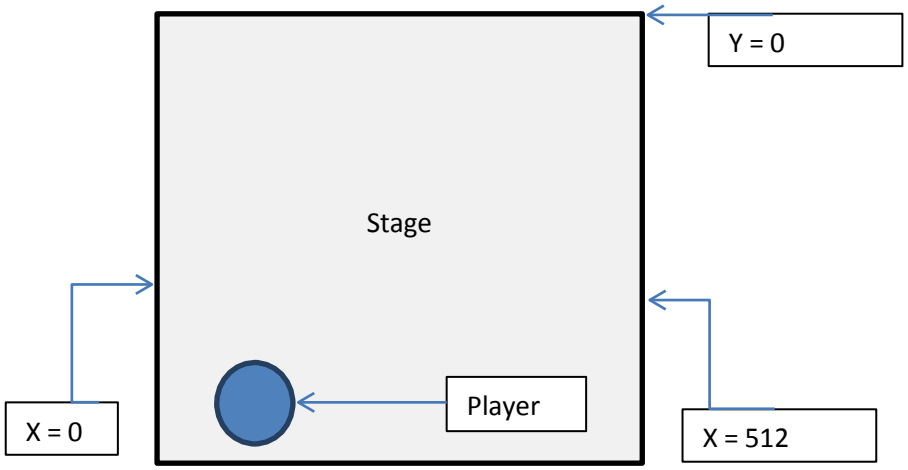
Inside the on Key Down function enter the following

```
if(e.KeyCode == Keys.Left)
            {
if (player.Location.X< 0)
                {
        moveLeft = 0;
                }
else
                {
        moveLeft = -5;
                }
            }
```

The code above is using an if statement to check whether the LEFT key on the keyboard is being press if it is then we will do the following.

Note because we are saying move left -5 doesn't mean it will only more 5 pixels to the left it will continuously move 5 pixels because we will add it with our timer. So bear with this before thinking nothing works yet.

To the understand the code lets visually demonstrate what we are doing -



Left corner of the form is always 0 for horizontal and vertical.

If key code equals to left key

If player's location is less than 0 which means player has left the screen then we set move left to 0 because we want the player to stay in the screen.

Else meaning now while the players location is not less than 0 then we can move the player to the left.

We will need more if statements in this function

```
else if(e.KeyCode == Keys.Right)
            {
if (player.Location.X> 512)
                {
         moveLeft = 0;
                }
else
                {
         moveLeft = 5;
                }
            }
```

We are doing an else if statement to check if the right key on the key board is press and then we will execute the following instructions.

Since we are dealing with the right direction it will be positive 5 instead of negative 5. So we are giving the form a limit of 512 pixels and for the player to move round. If player's location gets over 512 we will stop else we will continue to add 5 pixels to that direction.

Now for the firing of the bullet.

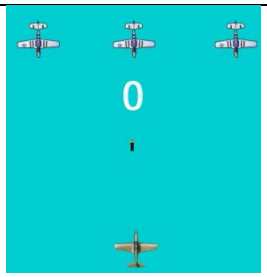
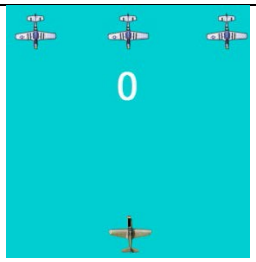
```
else if (e.KeyCode == Keys.Space)
            {

if (shooting == false)
                 {
bulletSpeed = 8;
bullet.Left = player.Left + 50;
bullet.Top = player.Top;
shooting = true;

                 }
            }
```

We have the similar set up as the left and right key here. This one will control the bullet trajectory.

Inside the else if statement there is a statement which is saying that if the shooting Boolean is equals to false then we do something. To stop players from often spamming the screen will unlimited bullets we will only allow players to shoot one bullet at a time.

```
bulletSpeed = 8;
bullet.Left = player.Left + 50;
bullet.Top = player.Top;
shooting = true;
```

Now here we will state how the bullet will behave in the game. The speed is set to 8, picture box that contains the bullet image must be aligned with the player so we will centre the image with the player. We are adding 50 more pixels on the left of the bullet to match the centre of the player image. We are also setting the image on the top of player image so it looks like the player is firing a bullet. Lastly we set the shooting to true so player cannot fire any more until that changes, which we will do later on.

 Bullet in the middle	 Bullet on top of the plane now.
Before pressing space key	After pressing the space key

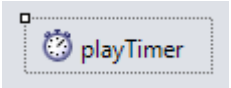
Now let's look at the on key up event

Inside the on key up function enter the following

```
if (e.KeyCode == Keys.Left)
            {
moveLeft = 0;
            }
else if (e.KeyCode == Keys.Right)
            {
moveLeft = 0;
            }
```

This function is very simple. We want to set everything to zero once the player has left the left or right key on the keyboard. Now notice we haven't given an instruction for the space key. Since there is already a Boolean to check if the player is shooting or not thus we don't need to give one here. Make things easy for ourselves.

Now for the timer function. **Double click** on the timer function on the form and it will automatically create a timer event.



```
player.Left += moveLeft;
bullet.Top -= bulletSpeed;
enemy1.Top += enemyMove;
enemy2.Top += enemyMove;
enemy3.Top += enemyMove;
scoreText.Text = "" + score;
```

Here we make things move. **+=**this sign means the program will increment the value each timer we press the button for example if the player is on 400px on the screen we press the it will add 5px to it and the longer we hold the key it will continuously add to it as it goes along unless we tell it to stop which we have in the on key down function.

Since the bullet only travels up it don't need a positive increment it needs to have negative so it's -= to the bullet speed.

Enemy 1 2 and 3 will be coming down the screen so we are giving them the positive increment.

We will update the score text on the screen according to the score variable.

```
if (enemy1.Top == 660 || enemy2.Top == 660 || enemy3.Top == 660)
            {
gameOver();
            }
```

Inside the timer function we will check if the enemy has left the stage if they have then we end the game and allow the user to restart. We will declare the game over function later on for now check how I have organized the if statement. Instead of having only one condition I have added 3. Inside the if statement it is possible to check multiple condition for example here they way its read is **if enemy 1 top is equals to 660 or enemy 2 top is equals to 660 or enemy 3 top is equals to 660**. If either of those things is true the game will meaning someone snuck passed the defense.

Now for the bullet

```
if (shooting && bullet.Top < 0)
            {
                shooting = false;
bulletSpeed = 0;
bullet.Top = -100;
bullet.Left = -100;
            }
```

In this if statement we will check multiple conditions however this one both needs to be true together in order for the instructions to execute.

If shooting is true (I am using a short hand in this code here) and bullet top is less 0 meaning bullet has left the stage area then we can enable the player to shoot bullets again by setting the shooting to false also we need to set the bullet speed to 0 and hide the bullet outside the stage thus the -100 and -100 so its out of the stage viewing area.

```
enemyHit();
```

Add this line after the last if statement. We will create the function which will contain what to do once we hit the enemy with the bullet next.

```
private void enemyHit()
{
//enemy hit codes here
}
```

Above we have declared the enemy hit function. In c sharp we have a function call Bounds which checks the height and width of a given object and we also can check if it collides with another object.

For example

Player.Bounds.IntersectsWith();

The line above will check a player object and whether it's intersecting with another which we need to mention inside the brackets.

Player.Bounds.IntersectsWith(enemy.Bounds);

Here we are checking if the player is crossing bounds with the enemy. If they are then we simply take certain action or increase the score or even kill the player. Whatever the game rules are.

In our game we need to check if the bullet is colliding with the enemy then we move the enemy and we reset the bullet for future use.

```
if (bullet.Bounds.IntersectsWith(enemy1.Bounds))
            {
                score += 1;
                enemy1.Top = -500;
                int ranP = rnd.Next(1, 300);
                enemy1.Left = ranP;
                shooting = false;
                bulletSpeed = 0;
                bullet.Top = -100;
                bullet.Left = -100;
            }
```

Notice we are checking the bounds and intersect with enemy here. So if the bullet hits this enemy then we will add 1 to the score, reset the enemy to -500 outside the form display, generate a random number between 1 and 300 to randomize the respawn location, set the enemy to that random respawn location, set the shooting to false since the enemy already been hit and rest the bullet properties as we have done before.

```
else if (bullet.Bounds.IntersectsWith(enemy2.Bounds))
            {
 score += 1;
enemy2.Top = -900;
int ranP = rnd.Next(1, 400);
enemy2.Left = ranP;
shooting = false;
bulletSpeed = 0;
bullet.Top = -100;
bullet.Left = -100;

            }
else if (bullet.Bounds.IntersectsWith(enemy3.Bounds))
            {
 score += 1;
enemy3.Top = -1300;
int ranP = rnd.Next(1, 500);
enemy3.Left = ranP;
shooting = false;
bulletSpeed = 0;
bullet.Top = -100;
bullet.Left = -100;
            }
```

Above are the other 2 enemies collision if statements. So notice it's the same except for where they will respawn.

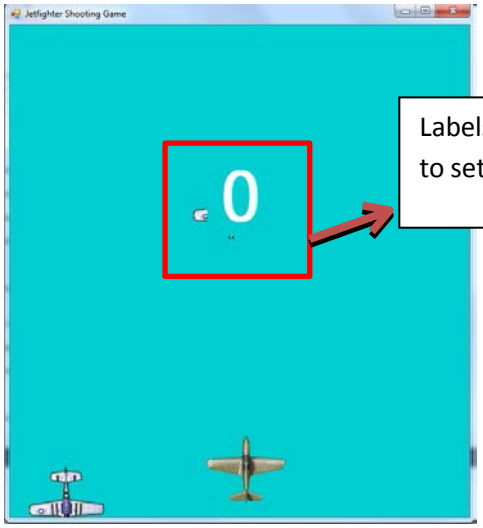
Well you don't want a predictable movement by the enemy then it's not so much fun, we want to make it interesting so we change things up.

Here have change the enemy 2 and 3 respawn location notice the highlighted fields.

```
private void gameOver()
{
playTimer.Enabled = false;
MessageBox.Show("You Score = " + score + " Click OK to play Again");
score = 0;
scoreText.Text = "0";
enemy1.Top = -500;
enemy2.Top = -900;
enemy3.Top = -1300;
bullet.Top = -100;
bullet.Left = -100; playTimer.Enabled = true;
}
```

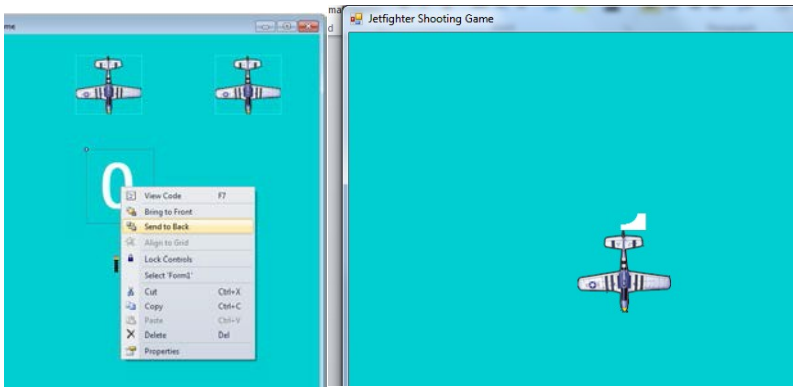
This is the game over function. When the enemy will cross over to the allied side it will end the game. First we will disable the timer so nothing runs anymore. We will then show a message box to show the players score and when they click ok we will reset the score, enemy's location, and bullet and then restart the timer. Simple and easy always wins.

Fixing issues -



Labels border is above the player we need to set the to the back ground.

Right click on the label in the design view and click on **send to back - Might have to do it couple of times to make sure its behind all 3 enemy pictures.**



Now it's working.

Double check the code and double check the design to make sure everything is working fine.