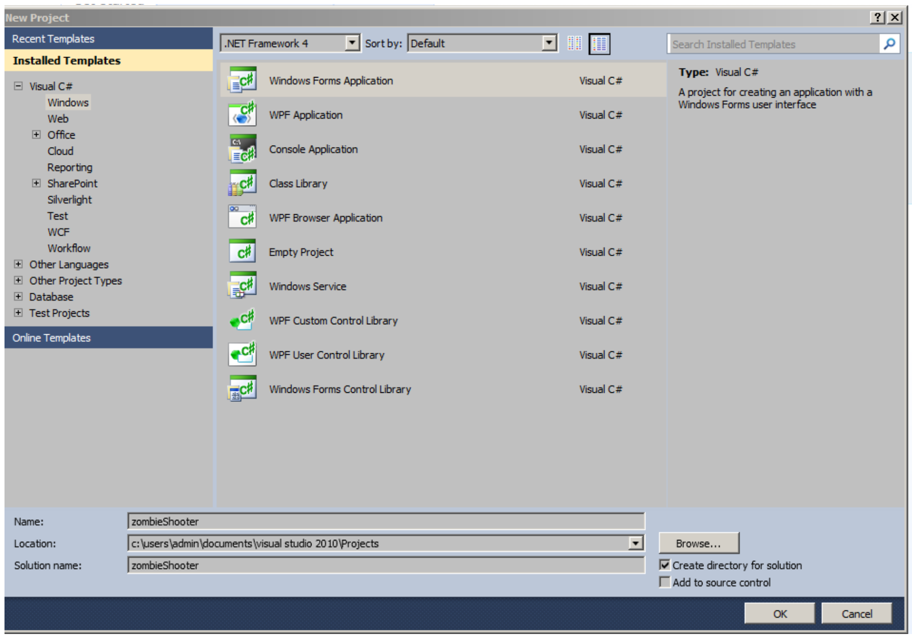
We are working on Visual Studio 2010 but this project can be remade in any other version of visual studio.



New Project

Recent Templates

Installed Templates

- Visual C#
  - Windows
  - Web
  - Office
  - Cloud
  - Reporting
  - SharePoint
  - Silverlight
  - Test
  - WCF
  - Workflow
- Other Languages
- Other Project Types
- Database
- Test Projects

Online Templates

.NET Framework 4 | Sort by: Default | Search Installed Templates

Windows Forms Application	Visual C#
WPF Application	Visual C#
Console Application	Visual C#
Class Library	Visual C#
WPF Browser Application	Visual C#
Empty Project	Visual C#
Windows Service	Visual C#
WPF Custom Control Library	Visual C#
WPF User Control Library	Visual C#
Windows Forms Control Library	Visual C#

Type: Visual C#

A project for creating an application with a Windows Forms user interface

Name:	zombieShooter
Location:	c:\users\admin\documents\visual studio 2010\Projects
Solution name:	zombieShooter

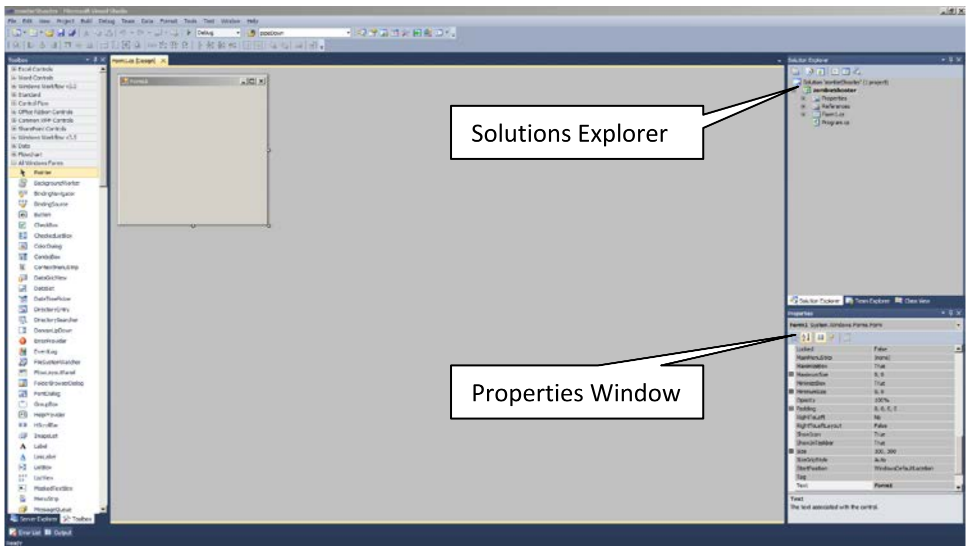
Browse...

☑ Create directory for solution

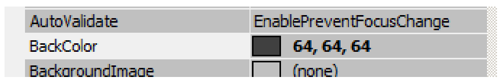
☐ Add to source control

OK Cancel

Start a new project in Visual Studio, make this a C# Windows Form Application and name it **zombieShooter**.

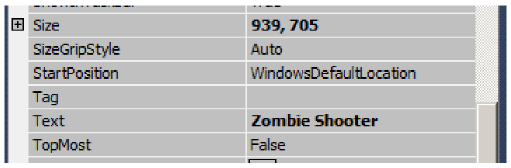


This is the empty form we will be working with in this game. Look under the properties window and find the following and change them.



AutoValidate	EnablePreventFocusChange
BackColor	■ **64, 64, 64**
BackgroundImage	☐ (none)

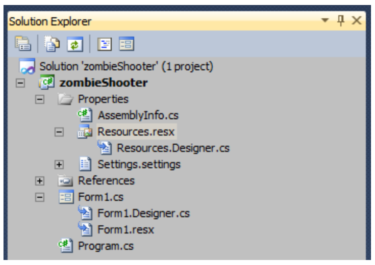
Change the back colour to **64, 64, 64** it's a dark grey colour.



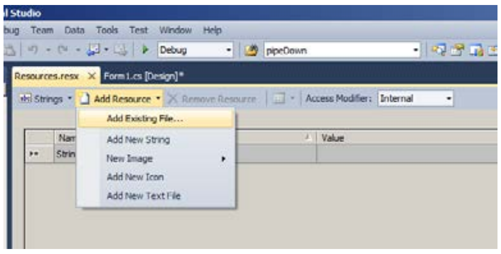
⊞ Size	**939, 705**
SizeGripStyle	Auto
StartPosition	WindowsDefaultLocation
Tag	
Text	**Zombie Shooter**
TopMost	False

Change the size to 939, 705

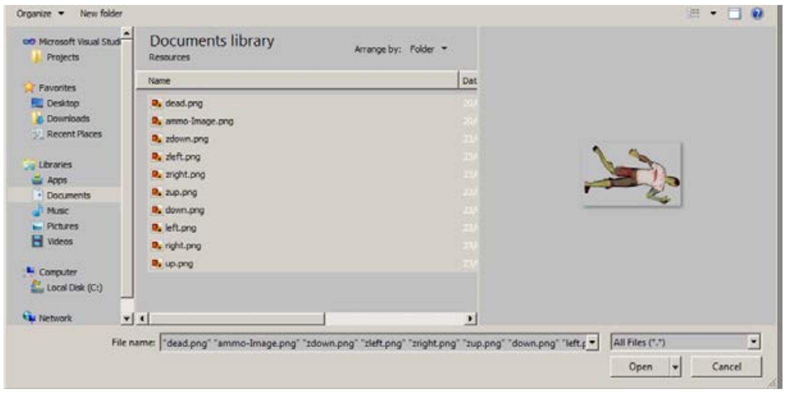
Text change to Zombie Shooter



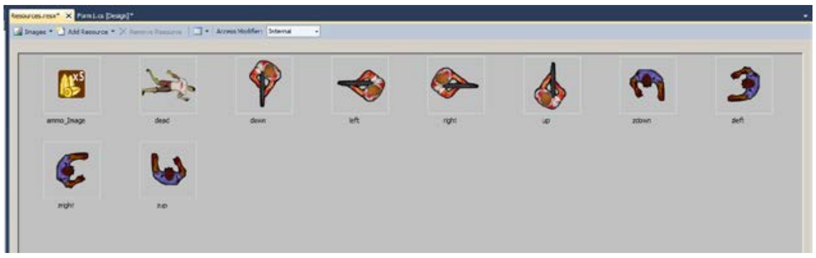
Under the Solution Explore, under the properties tab double click on the **Resources.resx**



In the resources panel click on the add resources option, from the drop down click on Add Existing File

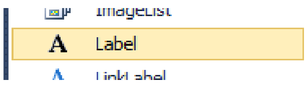


Highlight all of the images downloaded from **MOOICT.COM** tutorial page and click open.

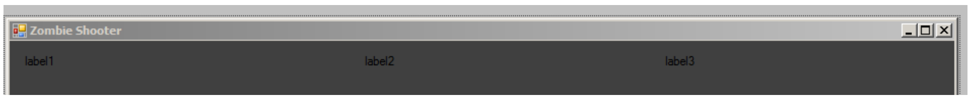


Here is has been added to the resources panel.

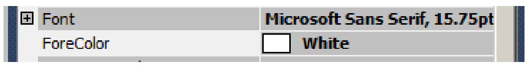
Now make sure to press CTRL + S or Save All button. (Do not do a save as or anything it will only save part of the project in a different folder. Always do Save ALL)



from the tool box drag and drop 2 labels to the form.

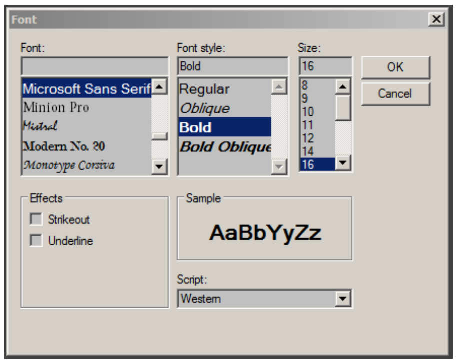


Above 3 labels has been added. Change the following in the properties window



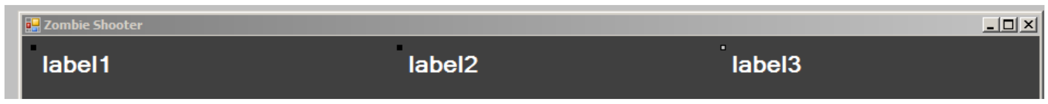
Font	Microsoft Sans Serif, 15.75pt
ForeColor	White

First change the fore colour to white. then click on the font option and click on the three dotted button (...)

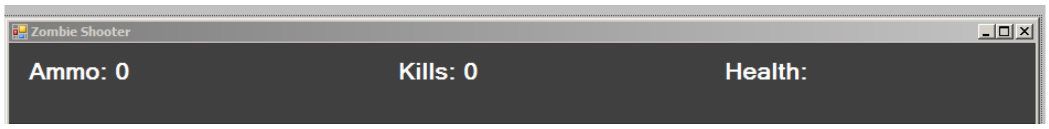


Font style: Bold
Size: 16

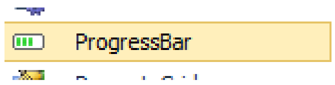
In this dialog box change the settings to BOLD and SIZE 16.



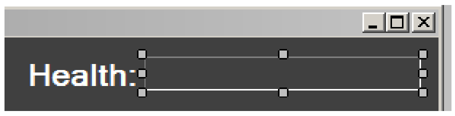
This is the changed labels in the form. Now change the text to the following -



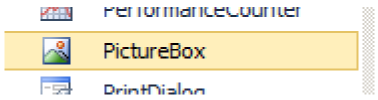
Label 1 change to Ammo: 0, label 2 change to Kills: 0 and change Label 3 to Health:



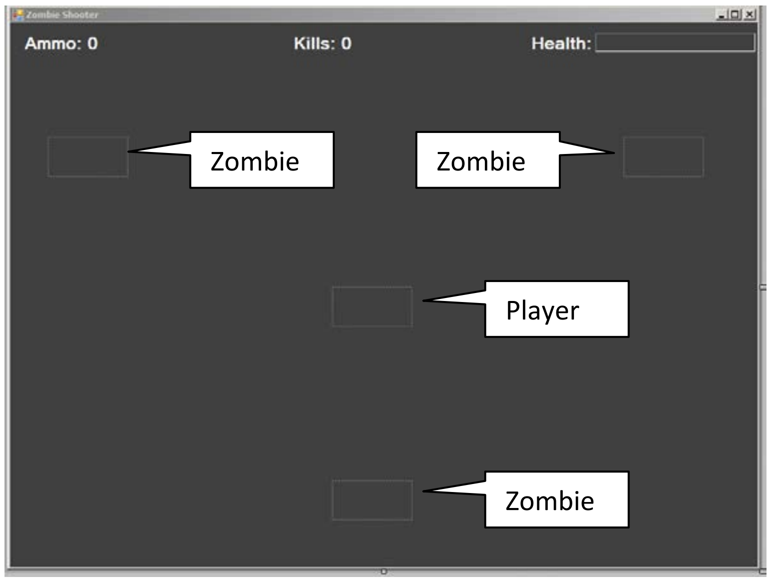
Find a progress bar from the tool box and drag it to the form.



Adjust the progress bar next to the health label. This will be used to show the health of the player.



Find the picture box from the tool box and drag and drop 4 picture boxes to the form.



4 picture boxes has been added to the form. Make sure they are well separated from each other.

Select the first picture box on the left and make the changes to the properties of the picture box.

Click on the Image option and then click on the 3 dotted button next to it.

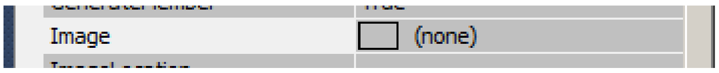
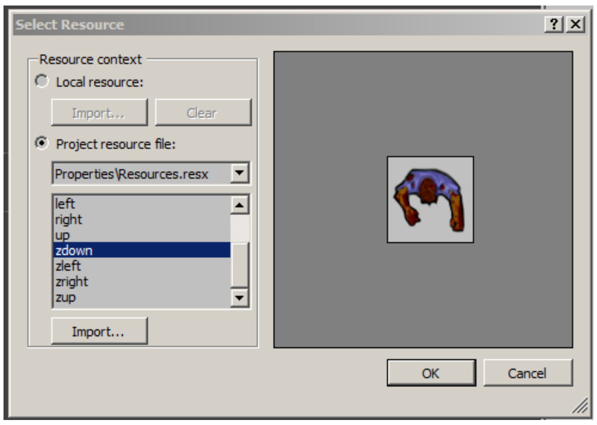
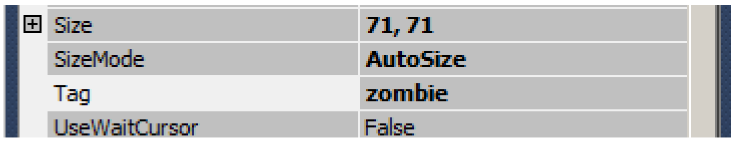


Image	(none)



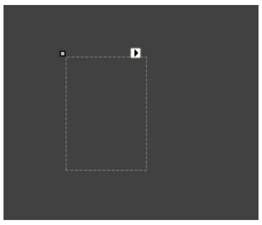
From the resources select the **zdown** picture and click OK.



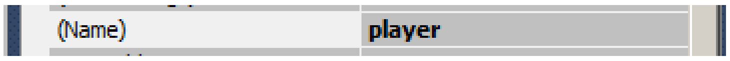
Size	71, 71
SizeMode	AutoSize
Tag	zombie
UseWaitCursor	False

Change the size mode to Auto Size, change the tag to zombie (lower case)

Do the same for the other zombie pictures.

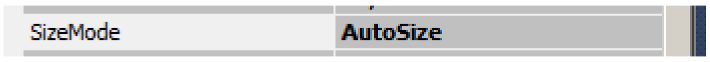


Select the middle picture box, look under the properties window.



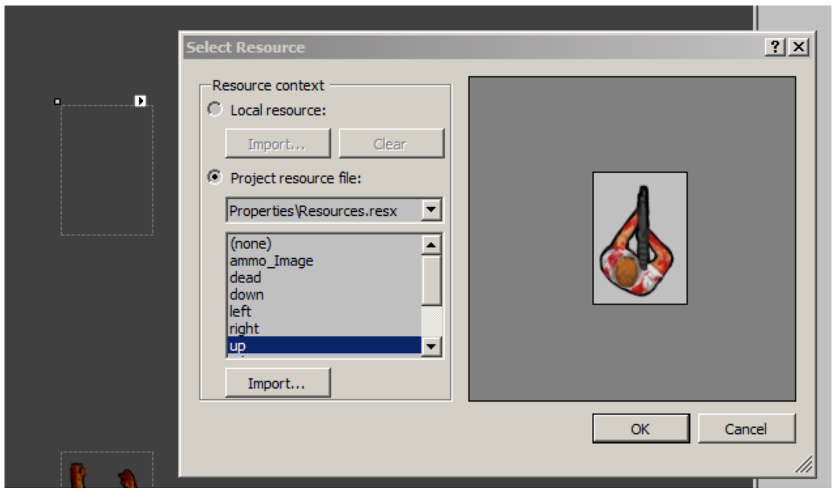
(Name)	player

change the name to player (lower case)

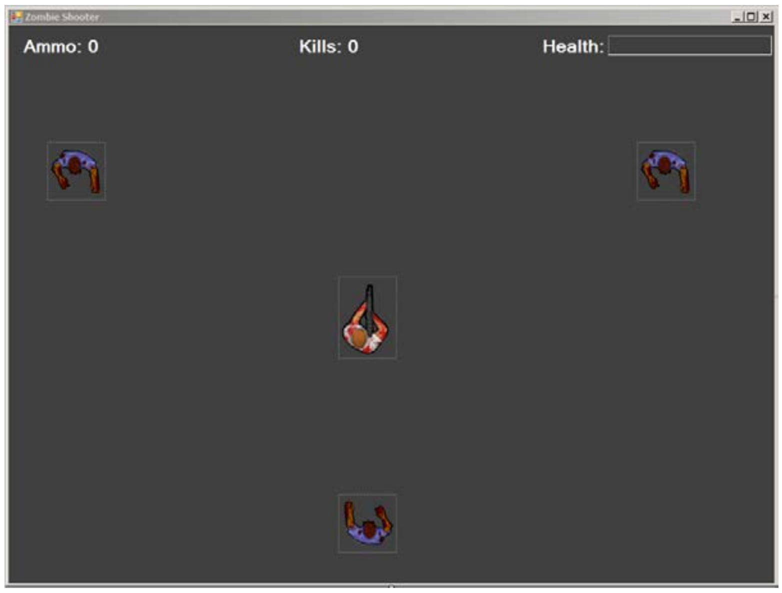


SizeMode	AutoSize

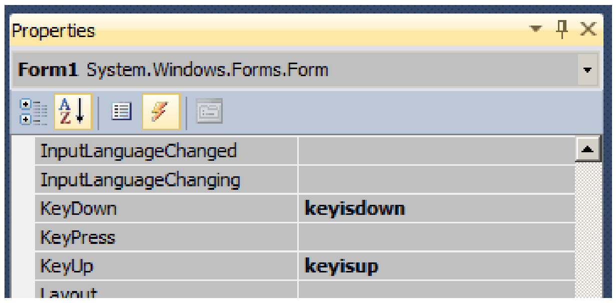
Change the size mode to Auto Size.



change the image of the player to UP. Click OK.



This is the view of the form so far. Looking good.



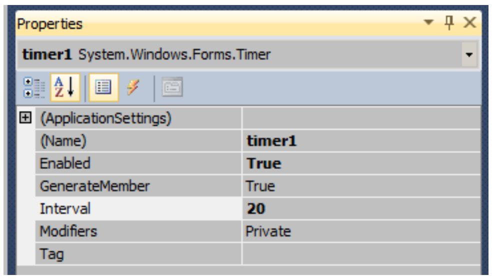
Properties

Form1 System.Windows.Forms.Form

InputLanguageChanged	
InputLanguageChanging	
KeyDown	**keyisdown**
KeyPress	
KeyUp	**keyisup**
Layout	

Click on the form and go to the events manager by clicking on the lightning bolt icon next to the properties icon. Find the key down and type in **keyisdown** press enter, in the key up event and type **keyisup** and press enter.

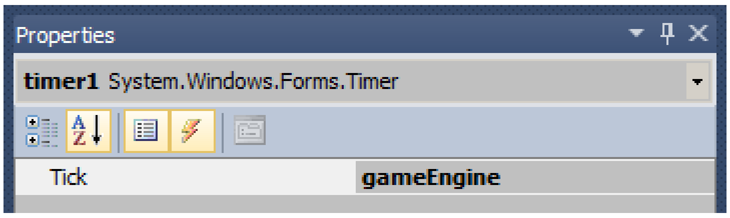
For the last component, find the timer component in the tool box and drag it down to the form.


Properties

timer1 System.Windows.Forms.Timer

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

Under the properties window for the timer change the enabled to **true**, interval to **20**. This means when the form will run the timer will start and it will tick 20 milliseconds.

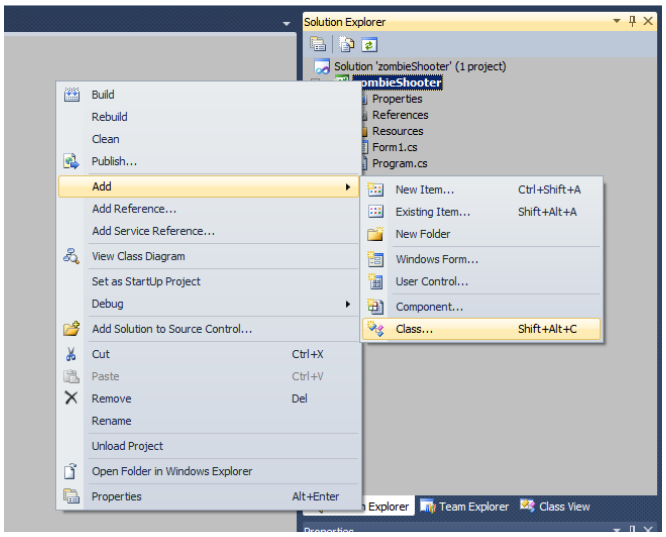

Properties

timer1 System.Windows.Forms.Timer

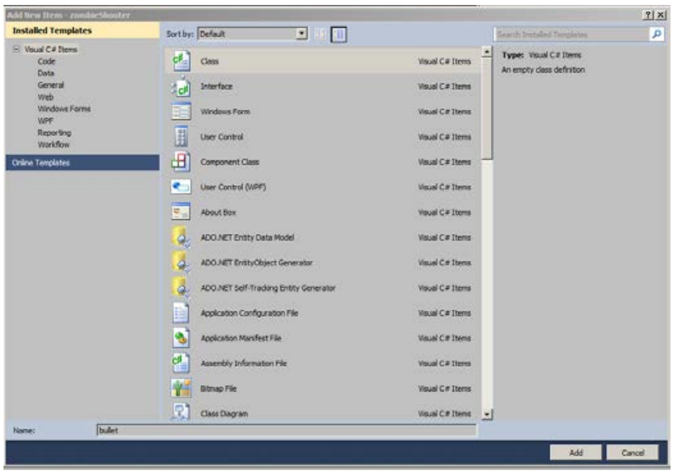
Tick	**gameEngine**

While the timer is selected, Click on that lightning bolt icon and type **gameEngine** and press enter.

We need to add the bullet, bullets will be added dynamically in the game when its fired but we need to program it in with the game. We will need to create a bullet CLASS. This is where we are touching on the subject of OOP or Object Oriented Programming. This model is very popular in game programming. First we need to create the class and then we can call in to do its thing in the game.



Right click on the **zombieShooter** in the solution Explorer. Hover over ADD then click on Class.

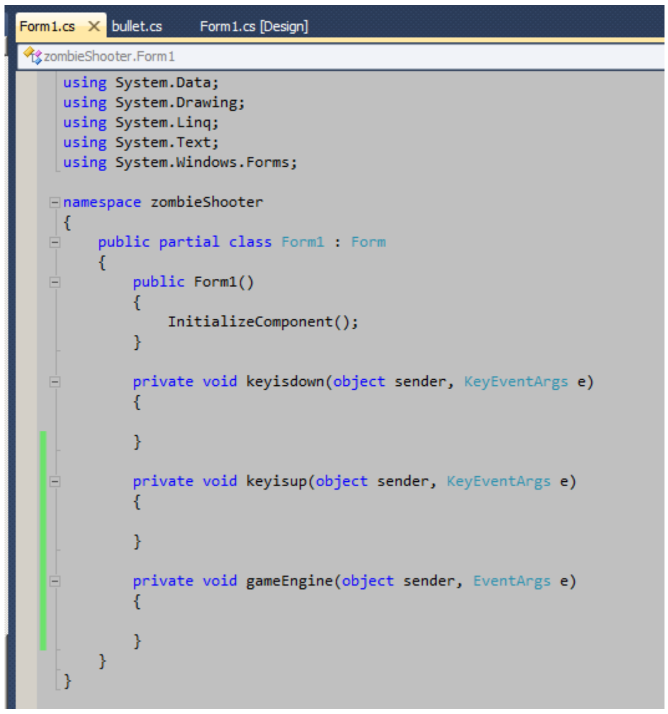

Add New Item - zombieShooter
Installed Templates
Visual C# Items
Code
Data
General
Web
Windows Forms
WPF
Reporting
Workflow
Online Templates
Sort by: Default
Search Installed Templates
Class Visual C# Items
Interface Visual C# Items
Windows Form Visual C# Items
User Control Visual C# Items
Component Class Visual C# Items
User Control (WPF) Visual C# Items
About Box Visual C# Items
ADO.NET Entity Data Model Visual C# Items
ADO.NET EntityObject Generator Visual C# Items
ADO.NET Self-Tracking Entity Generator Visual C# Items
Application Configuration File Visual C# Items
Application Manifest File Visual C# Items
Assembly Information File Visual C# Items
Bitmap File Visual C# Items
Class Diagram Visual C# Items
Add
Cancel

Type: Visual C# Items

An empty class definition

Name: bullet

Type in bullet and click OK. This will create a blank class for us with nothing in it. We will add things to it afterwards. Keep following the tasks.



```
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace zombieShooter
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void keyisdown(object sender, KeyEventArgs e)
        {

        }

        private void keyisup(object sender, KeyEventArgs e)
        {

        }

        private void gameEngine(object sender, EventArgs e)
        {

        }
    }
}
```

Let's look into the Form1.cs the code view of the main form. You can see the key is up, key is down and game engine events are already on there. Now we need to add our own variables and functions to this.

In the current codes we will add three of our own functions in it.

1 Drop Ammo function - this function will be used when the user needs more ammo in the game.

2 Shoot function - this function will be used when the player is shooing in the game.

3 Make Zombies function - this function will be used when we need to make more zombies in the game.

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

        private void keyisdown(object sender, KeyEventArgs e)
        {

        }

        private void keyisup(object sender, KeyEventArgs e)
        {

        }

        private void gameEngine(object sender, EventArgs e)
        {

        }

        private void DropAmmo()
        {

        }

        private void shoot(string direct)
        {
        }

        private void makeZombies()
        {

        }

    }
}
```

Highlighted lines are the custom function we added to the code.

Below are the variables

```
        // start the variables

bool goup; // this boolean will be used for the player to go up the screen
bool godown; // this boolean will be used for the player to go down the screen
bool goleft; // this boolean will be used for the player to go left to the screen
bool goright; // this boolean will be used for the player to right to the screen
string facing = "up"; // this string is called facing and it will be used to guide the bullets
double playerHealth = 100; // this double variable is called player health
int speed = 10; // this integer is for the speed of the player
int ammo = 10; // this integer will hold the number of ammo the player has start of the game
int zombieSpeed = 3; // this integer will hold the speed which the zombies move in the game
int score = 0; // this integer will hold the score the player achieved through the game
bool gameOver = false; // this boolean is false in the beginning and it will be used when the game is finished
Random rnd = new Random(); // this is an instance of the random class we will use this to create a random number
for this game

        // end of listing variables
```

Each of the variables are explained through the comments which are the green codes.

keyisdown Function

```
        private void keyisdown(object sender, KeyEventArgs e)
        {
            if (gameOver) return; // if game over is true then do nothing in this event

            // if the left key is pressed then do the following
            if (e.KeyCode == Keys.Left)
            {
                goleft = true; // change go left to true
                facing = "left"; //change facing to left
                player.Image = Properties.Resources.left; // change the player image to LEFT image
            }

            // end of left key selection

            // if the right key is pressed then do the following
            if (e.KeyCode == Keys.Right)
            {
                goright = true; // change go right to true
                facing = "right"; // change facing to right
                player.Image = Properties.Resources.right; // change the player image to right
            }
            // end of right key selection

            // if the up key is pressed then do the following
            if (e.KeyCode == Keys.Up)
            {
                facing = "up"; // change facing to up
                goup = true; // change go up to true
                player.Image = Properties.Resources.up; // change the player image to up
            }

            // end of up key selection

            // if the down key is pressed then do the following
            if (e.KeyCode == Keys.Down)
            {
                facing = "down"; // change facing to down
                godown = true; // change go down to true
                player.Image = Properties.Resources.down; //change the player image to down
            }
            // end of the down key selection
        }
```

All the code is explained through the comments.

keyisup Function

```
        private void keyisup(object sender, KeyEventArgs e)
        {
            if (gameOver) return; // if game is over then do nothing in this event

            // below is the key up selection for the left key
            if (e.KeyCode == Keys.Left)
            {
                goleft = false; // change the go left boolean to false
            }

            // below is the key up selection for the right key
            if (e.KeyCode == Keys.Right)
            {
                goright = false; // change the go right boolean to false
            }
            // below is the key up selection for the up key
            if (e.KeyCode == Keys.Up)
            {
                goup = false; // change the go up boolean to false
            }
            // below is the key up selection for the down key
            if (e.KeyCode == Keys.Down)
            {
                godown = false; // change the go down boolean to false
            }

            //below is the key up selection for the space key
            if (e.KeyCode == Keys.Space && ammo > 0) // in this if statement we are checking if the space bar is 
up and ammo is more than 0

            {
                ammo--; // reduce ammo by 1 from the total number
                shoot(facing); // invoke the shoot function with the facing string inside it
                //facing will transfer up, down, left or right to the function and that will shoot the bullet 
that way.

                if (ammo < 1) // if ammo is less than 1
                {
                    DropAmmo(); // invoke the drop ammo function
                }

            }
        }
```

This is the key up function. You will notice we have included the SPACE bar in this function. This is because we don't want the player to spam the Space key, the bullet will only release when the space bar is released.

Inside the function we included if the ammo is less than 1 meaning if its 0 then the game will drop an ammo box to the screen.

The game engine function

- caution this will be long. Pay extra attention DO NOT FORGET THE OPEN AND CLOSING CURLY BRACKETS.

```
       private void gameEngine(object sender, EventArgs e)
        {
            if (playerHealth > 1) // if player health is greater than 1
            {
                progressBar1.Value = Convert.ToInt32(playerHealth); // assign the progress bar to the player 
health integer
            }
            else
            {
                // if the player health is below 1
                player.Image = Properties.Resources.dead; // show the player dead image
                timer1.Stop(); // stop the timer
                gameOver = true; // change game over to true
            }

            label1.Text = "   Ammo:   " + ammo; // show the ammo amount on label 1
            label2.Text = "Kills: " + score; // show the total kills on the score

            // if the player health is less than 20
            if (playerHealth < 20)
            {
                progressBar1.ForeColor = System.Drawing.Color.Red; // change the progress bar colour to red.
            }

            if (goleft && player.Left > 0)
            {
                player.Left -= speed;
                // if moving left is true AND pacman is 1 pixel more from the left
                // then move the player to the LEFT
            }
            if (goright && player.Left + player.Width < 930)
            {
                player.Left += speed;
                // if moving RIGHT is true AND player left + player width is less than 930 pixels
                // then move the player to the RIGHT
            }
            if (goup && player.Top > 60)
            {
                player.Top -= speed;
                // if moving TOP is true AND player is 60 pixel more from the top
                // then move the player to the UP
            }

            if (godown && player.Top + player.Height < 700)
            {
                player.Top += speed;
                // if moving DOWN is true AND player top + player height is less than 700 pixels
                // then move the player to the DOWN
            }

            // run the first for each loop below
            // X is a control and we will search for all controls in this loop
            foreach (Control x in this.Controls)
            {
                // if the X is a picture box and X has a tag AMMO

                if (x is PictureBox && x.Tag == "ammo")
                {
                    // check is X in hitting the player picture box

                    if (((PictureBox)x).Bounds.IntersectsWith(player.Bounds))
                    {
                        // once the player picks up the ammo

                        this.Controls.Remove(((PictureBox)x)); // remove the ammo picture box

                        ((PictureBox)x).Dispose(); // dispose the picture box completely from the program
                        ammo += 5; // add 5 ammo to the integer
                    }
                }

                // if the bullets hits the 4 borders of the game
                // if x is a picture box and x has the tag of bullet

                if (x is PictureBox && x.Tag == "bullet")
                {
                    // if the bullet is less the 1 pixel to the left
                    // if the bullet is more then 930 pixels to the right
                    // if the bullet is 10 pixels from the top
                    // if the bullet is 700 pixels to the buttom

        if (((PictureBox)x).Left < 1 || ((PictureBox)x).Left > 930 || ((PictureBox)x).Top < 10 ||
((PictureBox)x).Top > 700)
                    {
                        this.Controls.Remove(((PictureBox)x)); // remove the bullet from the display
                        ((PictureBox)x).Dispose(); // dispose the bullet from the program
                    }
                }

                // below is the if statement which will be checking if the player hits a zombie

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

                    // below is the if statament thats checking the bounds of the player and the zombie

                    if (((PictureBox)x).Bounds.IntersectsWith(player.Bounds))
                    {
                        playerHealth -= 1; // if the zombie hits the player then we decrease the health by 1
                    }

                    //move zombie towards the player picture box

                    if (((PictureBox)x).Left > player.Left)
                    {
                     ((PictureBox)x).Left -= zombieSpeed; // move zombie towards the left of the player
                     ((PictureBox)x).Image = Properties.Resources.zleft; // change the zombie image to the left
                    }

                    if (((PictureBox)x).Top > player.Top)
                    {
                        ((PictureBox)x).Top -= zombieSpeed; // move zombie upwards towards the players top
                        ((PictureBox)x).Image = Properties.Resources.zup; // change the zombie picture to the top
pointing image
                    }
                    if (((PictureBox)x).Left < player.Left)
                    {
                     ((PictureBox)x).Left += zombieSpeed; // move zombie towards the right of the player
                     ((PictureBox)x).Image = Properties.Resources.zright; // change the image to the right image
                    }
                    if (((PictureBox)x).Top < player.Top)
                    {
                      ((PictureBox)x).Top += zombieSpeed; // move the zombie towards the bottom of the player
                      ((PictureBox)x).Image = Properties.Resources.zdown; // change the image to the down zombie
                    }
                }

                // below is the second for loop, this is nexted inside the first one
                // the bullet and zombie needs to be different than each other
                // then we can use that to determine if the hit each other

                foreach (Control j in this.Controls)
                {
                    // below is the selection thats identifying the bullet and zombie

                    if ((j is PictureBox && j.Tag == "bullet") && (x is PictureBox && x.Tag == "zombie"))
                    {
                        // below is the if statement thats checking if bullet hits the zombie
                        if (x.Bounds.IntersectsWith(j.Bounds))
                        {
                            score++; // increase the kill score by 1
                            this.Controls.Remove(j); // this will remove the bullet from the screen
                            j.Dispose(); // this will dispose the bullet all together from the program
                            this.Controls.Remove(x); // this will remove the zombie from the screen
                            x.Dispose(); // this will dispose the zombie from the program
                            makeZombies(); // this function will invoke the make zombies function to add another
zombie to the game
                        }
                    }
                }
            }
        }
```

There are two for each loop and several if statement which are inside this event. All of the codes are explained in the comments next to them. Pay extra attention to them, if there are any red lines under the code then come back to this and check the code.

Drop Ammo function -

```
        private void DropAmmo()
        {
            // this function will make a ammo image for this game

            PictureBox ammo = new PictureBox(); // create a new instance of the picture box
            ammo.Image = Properties.Resources.ammo_Image; // assignment the ammo image to the picture box
            ammo.SizeMode = PictureBoxSizeMode.AutoSize; // set the size to auto size
            ammo.Left = rnd.Next(10, 890); // set the location to a random left
            ammo.Top = rnd.Next(50, 600); // set the location to a random top
            ammo.Tag = "ammo"; // set the tag to ammo
            this.Controls.Add(ammo); // add the ammo picture box to the screen
            ammo.BringToFront(); // bring it to front
            player.BringToFront(); // bring the player to front
        }
```

Shoot function -

```
        private void shoot(string direct)
        {
            // this is the function thats makes the new bullets in this game

            bullet shoot = new bullet(); // create a new instance of the bullet class
            shoot.direction = direct; // assignment the direction to the bullet
            shoot.bulletLeft = player.Left + (player.Width / 2); // place the bullet to left half of the player
            shoot.bulletTop = player.Top + (player.Height / 2); // place the bullet on top half of the player
            shoot.mkBullet(this); // run the function mkBullet from the bullet class.
        }
```

I will explain this function in more detail because as you type it might not recognise some of properties you are inputting in there, that's not because it's wrong that's because we haven't created that bullet class just yet.

Let's start with the beginning of the function

`private void shoot(string direct)`
you will notice this function has a argument inside the smaller brackets called string direct. This is a local variable only for this function. What this means is we will need to pass in a string value inside this function. When this function is called some sort of string needs to be passed to it for it to function.

for example we can use shoot("never") or shoot("i am clever") it will work but it will not produce the results we want it to. For the function to work as we want to we need to pass in the facing string we created earlier and that's responding to the key down event too.

`bullet shoot = new bullet(); // create a new instance of the bullet class`

In the line above we are creating an instance of the bullet class. By doing it like this we can bring in multiple instances of this class just by calling it. Good thing about OOP is when you call a class like this you bring in all of the properties with it.

`shoot.direction = direct; // assignment the direction to the bullet`

Now that the name SHOOT has been linked with the bullet class we can access the properties of the class and modify them any way we want to. In this case there will be a variable called direction inside the bullet class and this will determine which way the bullet is going up, down, left or right.

`shoot.bulletLeft = player.Left + (player.Width / 2); // place the bullet to left half of the player`
The line above is accessing the bulletLeft variable and assigning its value to it. This variable is used to determine the start position of the bullet. For this case we are assigning to the players left position and we are dividing the players height by 2 which means it will be positioned right in the middle of the player and where the guns pointed.

`shoot.bulletTop = player.Top + (player.Height / 2); // place the bullet on top half of the player`
This line above is doing the same thing as before but for the top position of the bullet.

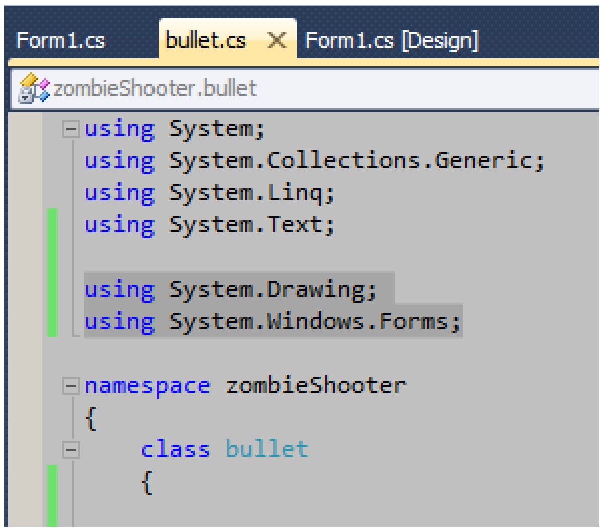
`shoot.mkBullet(this); // run the function mkBullet from the bullet class.`

This line is allowing us to call this function inside the class. We will be creating it, so if there is a red line under it now do not worry we are getting there. Notice that we are stating (this) on the function which means same as this function we are sending some arguments to help adding the bullet to screen.

Make Zombies function -

```
        private void makeZombies()
        {
            // when this function is called it will make zombies in the game

            PictureBox zombie = new PictureBox(); // create a new picture box called zombie
            zombie.Tag = "zombie"; // add a tag to it called zombie
            zombie.Image = Properties.Resources.zdown; // the default picture for the zombie is zdown
            zombie.Left = rnd.Next(0, 900); // generate a number between 0 and 900 and assignment that to the new
zombies left
            zombie.Top = rnd.Next(0, 800); // generate a number between 0 and 800 and assignment that to the new
zombies top
            zombie.SizeMode = PictureBoxSizeMode.AutoSize; // set auto size for the new picture box
            this.Controls.Add(zombie); // add the picture box to the screen
            player.BringToFront(); // bring the player to the front
        }
```

Now it's time to start on the bullet class -



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using System.Drawing;
using System.Windows.Forms;

namespace zombieShooter
{
    class bullet
    {
```

First under the using part of the class add the two highlighted above. First one is **System.Drawing;** and second is **System.Windows.Forms;** by adding these two we are able to use the windows components such as times, picture boxes and edit their properties. This is very important, without these two you cannot get access to the windows components make sure you add them both in this class.

Let's start with the variables first.

```
        // start the variable

        public string direction; // creating a public string called direction
        public int speed = 20; // creating a integer called speed and assigning a value of 20
        PictureBox Bullet = new PictureBox(); // create a picture box
        Timer tm = new Timer(); // create a new timer called tm.

        public int bulletLeft; // create a new public integer
        public int bulletTop; // create a new public integer

        // end of the variables
```

Note - when we are creating variables in a class and those we want to use outside the class we need to put a public in front of the variable declaration. Remember the shoot function we creating this is what that was doing before, we are calling the variables from this class to the other function.

mkBullet Function

Below is the **mkBullet** function. Notice this one has a public declaration which also means we need to access this from out of this class. We used this in the shoot function.

```
        public void mkBullet(Form form)
        {
            // this function will add the bullet to the game play
            // it is required to be called from the main class

            Bullet.BackColor = System.Drawing.Color.White; // set the colour white for the bullet
            Bullet.Size = new Size(5, 5); // set the size to the bullet to 5 pixel by 5 pixel
            Bullet.Tag = "bullet"; // set the tag to bullet
            Bullet.Left = bulletLeft; // set bullet left
            Bullet.Top = bulletTop; // set bullet right
            Bullet.BringToFront(); // bring the bullet to front of other objects
            form.Controls.Add(Bullet); // add the bullet to the screen

            tm.Interval = speed; // set the timer interval to speed
            tm.Tick += new EventHandler(tm_Tick); // assignment the timer with an event
            tm.Start(); // start the timer
        }
```

In the **mkBullet** function the first 7 lines of code is self explanatory and it has been explained in the comments. The lines I will be expanding on are the last 3.

```
tm.Interval = speed; // set the timer interval to speed
```

TM is our timer we created earlier in the CLASS. The interval is set to the speed, in this case its 20 which means the bullet will move 20 milliseconds towards any direction we set.

```
tm.Tick += new EventHandler(tm_Tick); // assignment the timer with an event
```

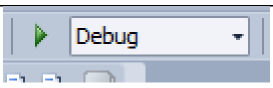
TM needs a tick event linked to the time in order for us to move the bullet across the screen. It's a good idea to assign a time to each of the bullets created through this and each button will be independent from the last. **+= new EventHandler()** is the event invoker and we have inserted **tm_Tick** is our event. We also need to declare the tm tick event next.

```
tm.Start(); // start the timer
```

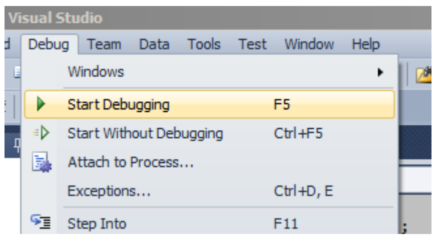
Lastly we are starting the timer in this function.

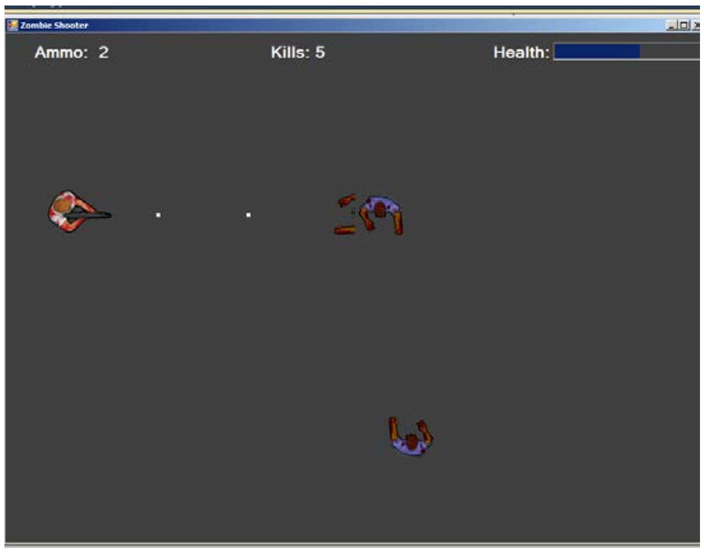
tm_Tick event - Unlike the game engine event this will be a smaller event function.

```
        public void tm_Tick(object sender, EventArgs e)
        {
            // if direction equals to left
            if (direction == "left")
            {
                Bullet.Left -= speed; // move bullet towards the left of the screen
            }
            // if direction equals right
            if (direction == "right")
            {
                Bullet.Left += speed; // move bullet towards the right of the screen
            }
            // if direction is up
            if (direction == "up")
            {
                Bullet.Top -= speed; // move the bullet towards top of the screen
            }
            // if direction is down
            if (direction == "down")
            {
                Bullet.Top += speed; // move the bullet bottom of the screen
            }

            // if the bullet is less the 16 pixel to the left OR
            // if the bullet is more than 860 pixels to the right OR
            // if the bullet is 10 pixels from the top OR
            // if the bullet is 616 pixels to the bottom OR
            // IF ANY ONE OF THE CONDITIONS ARE MET THEN THE FOLLOWING CODE WILL BE EXECUTED

            if (Bullet.Left < 16 || Bullet.Left > 860 || Bullet.Top < 10 || Bullet.Top > 616)
            {
                tm.Stop(); // stop the timer
                tm.Dispose(); // dispose the timer event and component from the program
                Bullet.Dispose(); // dispose the bullet
                tm = null; // nullify the timer object
                Bullet = null; // nullify the bullet object
            }
        }
```

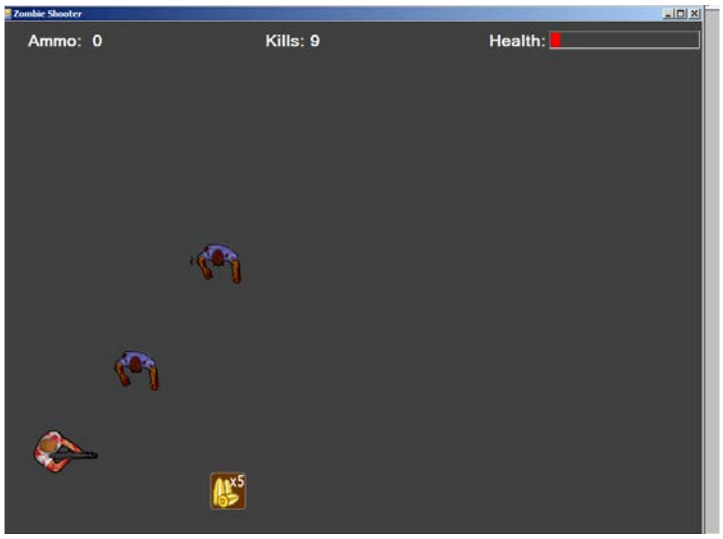
With all this done now it's time to debug and test the game out. 

or





Shooting bullets work and zombies die from the screen and re-spawn when one dies in the game.



Ammo drop is working too and so is the health bar it changed colour when health dropped below 20.

Full Form1.cs 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 zombieShooter
{
    public partial class Form1 : Form
    {

        // start the variables

        bool goup; // this boolean will be used for the player to go up the screen
        bool godown; // this boolean will be used for the player to go down the screen
        bool goleft; // this boolean will be used for the player to go left to the screen
        bool goright; // this boolean will be used for the player to right to the screen
        string facing = "up"; // this string is called facing and it will be used to guide the bullets
        double playerHealth = 100; // this double vaiable is called player health
        int speed = 10; // this interget is for the speed of the player
        int ammo = 10; // this integer will hold the number of ammo the player has start of the game
        int zombieSpeed = 3; // this integer will hold the speed which the zombies move in the game
        int score = 0; // this integer will hold the score the player achieved through the game
        bool gameOver = false; // this boolean is false in the beginning and it will be used when the game is
finished
        Random rnd = new Random(); // this is an instance of the random class we will use this to create a random
```

```
number for this game

        // end of listing variables


        public Form1()
        {
            InitializeComponent();
        }

        private void keyisdown(object sender, KeyEventArgs e)
        {
            if (gameOver) return; // if game over is true then do nothing in this event

            // if the left key is pressed then do the following
            if (e.KeyCode == Keys.Left)
            {
                goleft = true; // change go left to true
                facing = "left"; //change facing to left
                player.Image = Properties.Resources.left; // change the player image to LEFT image

            }

            // end of left key selection

            // if the right key is pressed then do the following
            if (e.KeyCode == Keys.Right)
            {
                goright = true; // change go right to true
                facing = "right"; // change facing to right
                player.Image = Properties.Resources.right; // change the player image to right

            }
            // end of right key selection

            // if the up key is pressed then do the following
            if (e.KeyCode == Keys.Up)
            {
                facing = "up"; // change facing to up
                goup = true; // change go up to true
                player.Image = Properties.Resources.up; // change the player image to up

            }

            // end of up key selection

            // if the down key is pressed then do the following
            if (e.KeyCode == Keys.Down)
            {
                facing = "down"; // change facing to down
                godown = true; // change go down to true
                player.Image = Properties.Resources.down; //change the player image to down

            }
            // end of the down key selection

        }

        private void keyisup(object sender, KeyEventArgs e)
        {
            if (gameOver) return; // if game is over then do nothing in this event

            // below is the key up selection for the left key
            if (e.KeyCode == Keys.Left)
            {
                goleft = false; // change the go left boolean to false
            }

            // below is the key up selection for the right key
            if (e.KeyCode == Keys.Right)
            {
                goright = false; // change the go right boolean to false
            }
            // below is the key up selection for the up key
            if (e.KeyCode == Keys.Up)
            {
                goup = false; // change the go up boolean to false
            }
            // below is the key up selection for the down key
            if (e.KeyCode == Keys.Down)
            {
                godown = false; // change the go down boolean to false
            }

            //below is the key up selection for the space key
            if (e.KeyCode == Keys.Space && ammo > 0) // in this if statement we are checking if the space bar is 
up and ammo is more than 0
            {
                ammo--; // reduce ammo by 1 from the total number
                shoot(facing); // invoke the shoot function with the facing string inside it
                //facing will transfer up, down, left or right to the function and that will shoot the bullet 
that way.

                if (ammo < 1) // if ammo is less than 1
                {
                    DropAmmo(); // invoke the drop ammo function
                }
            }
        }

        private void gameEngine(object sender, EventArgs e)
        {
            if (playerHealth > 1) // if player health is greater than 1
            {
                progressBar1.Value = Convert.ToInt32(playerHealth); // assign the progress bar to the player 
health integer
            }
            else
            {
                // if the player health is below 1
                player.Image = Properties.Resources.dead; // show the player dead image
                timer1.Stop(); // stop the timer
                gameOver = true; // change game over to true
            }

            label1.Text = "   Ammo:  " + ammo; // show the ammo amount on label 1
            label2.Text = "Kills: " + score; // show the total kills on the score

            // if the player health is less than 20
            if (playerHealth < 20)
            {
                progressBar1.ForeColor = System.Drawing.Color.Red; // change the progress bar colour to red.
            }

            if (goleft && player.Left > 0)
            {
                player.Left -= speed;
                // if moving left is true AND pacman is 1 pixel more from the left
                // then move the player to the LEFT
            }
            if (goright && player.Left + player.Width < 930)
            {
                player.Left += speed;
                // if moving RIGHT is true AND player left + player width is less than 930 pixels
                // then move the player to the RIGHT
            }
            if (goup && player.Top > 60)
            {
                player.Top -= speed;
                // if moving TOP is true AND player is 60 pixel more from the top
                // then move the player to the UP
            }

            if (godown && player.Top + player.Height < 700)
            {
                player.Top += speed;
                // if moving DOWN is true AND player top + player height is less than 700 pixels
                // then move the player to the DOWN
            }

            // run the first for each loop below
            // X is a control and we will search for all controls in this loop
            foreach (Control x in this.Controls)
            {
                // if the X is a picture box and X has a tag AMMO

                if (x is PictureBox && x.Tag == "ammo")
                {
                    // check is X in hitting the player picture box

                    if (((PictureBox)x).Bounds.IntersectsWith(player.Bounds))
                    {
                        // once the player picks up the ammo

                        this.Controls.Remove(((PictureBox)x)); // remove the ammo picture box

                        ((PictureBox)x).Dispose(); // dispose the picture box completely from the program
                        ammo += 5; // add 5 ammo to the integer
                    }
                }
                // if the bullets hits the 4 borders of the game
                // if x is a picture box and x has the tag of bullet

                if (x is PictureBox && x.Tag == "bullet")
                {
                    // if the bullet is less the 1 pixel to the left
                    // if the bullet is more then 930 pixels to the right
                    // if the bullet is 10 pixels from the top
                    // if the bullet is 700 pixels to the buttom

                    if (((PictureBox)x).Left < 1 || ((PictureBox)x).Left > 930 || ((PictureBox)x).Top < 10 ||
((PictureBox)x).Top > 700)
                    {
                        this.Controls.Remove(((PictureBox)x)); // remove the bullet from the display
                        ((PictureBox)x).Dispose(); // dispose the bullet from the program
                    }
                }

                // below is the if statement which will be checking if the player hits a zombie

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

                    // below is the if statament thats checking the bounds of the player and the zombie

                    if (((PictureBox)x).Bounds.IntersectsWith(player.Bounds))
                    {
                        playerHealth -= 1; // if the zombie hits the player then we decrease the health by 1
                    }

                    //move zombie towards the player picture box

                    if (((PictureBox)x).Left > player.Left)
                    {
                        ((PictureBox)x).Left -= zombieSpeed; // move zombie towards the left of the player
                        ((PictureBox)x).Image = Properties.Resources.zleft; // change the zombie image to the
left
                    }

                    if (((PictureBox)x).Top > player.Top)
                    {
                        ((PictureBox)x).Top -= zombieSpeed; // move zombie upwards towards the players top
                        ((PictureBox)x).Image = Properties.Resources.zup; // change the zombie picture to the top
pointing image
                    }
                    if (((PictureBox)x).Left < player.Left)
                    {
                        ((PictureBox)x).Left += zombieSpeed; // move zombie towards the right of the player
                        ((PictureBox)x).Image = Properties.Resources.zright; // change the image to the right
image
                    }
                    if (((PictureBox)x).Top < player.Top)
                    {
                        ((PictureBox)x).Top += zombieSpeed; // move the zombie towards the bottom of the player
                        ((PictureBox)x).Image = Properties.Resources.zdown; // change the image to the down
zombie
                    }
                }

                // below is the second for loop, this is nexted inside the first one
                // the bullet and zombie needs to be different than each other
                // then we can use that to determine if the hit each other

                foreach (Control j in this.Controls)
                {
                    // below is the selection thats identifying the bullet and zombie

                    if ((j is PictureBox && j.Tag == "bullet") && (x is PictureBox && x.Tag == "zombie"))
                    {
                        // below is the if statement thats checking if bullet hits the zombie
                        if (x.Bounds.IntersectsWith(j.Bounds))
                        {
                            score++; // increase the kill score by 1
                            this.Controls.Remove(j); // this will remove the bullet from the screen
                            j.Dispose(); // this will dispose the bullet all together from the program
                            this.Controls.Remove(x); // this will remove the zombie from the screen
                            x.Dispose(); // this will dispose the zombie from the program
                            makeZombies(); // this function will invoke the make zombies function to add another
zombie to the game
                        }
                    }
                }
            }
        }
        private void DropAmmo()
        {
            // this function will make a ammo image for this game

            PictureBox ammo = new PictureBox(); // create a new instance of the picture box
            ammo.Image = Properties.Resources.ammo_Image; // assignment the ammo image to the picture box
            ammo.SizeMode = PictureBoxSizeMode.AutoSize; // set the size to auto size
            ammo.Left = rnd.Next(10, 890); // set the location to a random left
            ammo.Top = rnd.Next(50, 600); // set the location to a random top
            ammo.Tag = "ammo"; // set the tag to ammo
            this.Controls.Add(ammo); // add the ammo picture box to the screen
            ammo.BringToFront(); // bring it to front
            player.BringToFront(); // bring the player to front
        }

        private void shoot(string direct)
        {
            // this is the function thats makes the new bullets in this game

            bullet shoot = new bullet(); // create a new instance of the bullet class
            shoot.direction = direct; // assignment the direction to the bullet
            shoot.bulletLeft = player.Left + (player.Width / 2); // place the bullet to left half of the player
            shoot.bulletTop = player.Top + (player.Height / 2); // place the bullet on top half of the player
            shoot.mkBullet(this); // run the function mkBullet from the bullet class.
        }

        private void makeZombies()
        {
            // when this function is called it will make zombies in the game

            PictureBox zombie = new PictureBox(); // create a new picture box called zombie
            zombie.Tag = "zombie"; // add a tag to it called zombie
            zombie.Image = Properties.Resources.zdown; // the default picture for the zombie is zdown
            zombie.Left = rnd.Next(0, 900); // generate a number between 0 and 900 and assignment that to the new
zombies left
            zombie.Top = rnd.Next(0, 800); // generate a number between 0 and 800 and assignment that to the new
zombies top
            zombie.SizeMode = PictureBoxSizeMode.AutoSize; // set auto size for the new picture box
            this.Controls.Add(zombie); // add the picture box to the screen
            player.BringToFront(); // bring the player to the front
        }

    }
}
```

bullet.cs - bullet class full code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using System.Drawing;
using System.Windows.Forms;

namespace zombieShooter
{
    class bullet
    {

        // start the variable

        public string direction; // creating a public string called direction
        public int speed = 20; // creating a integer called speed and assigning a value of 20
        PictureBox Bullet = new PictureBox(); // create a picture box
        Timer tm = new Timer(); // create a new timer called tm.

        public int bulletLeft; // create a new public integer
        public int bulletTop; // create a new public integer

        // end of the variables

        public void mkBullet(Form form)
        {
            // this function will add the bullet to the game play
            // it is required to be called from the main class

            Bullet.BackColor = System.Drawing.Color.White; // set the colour white for the bullet
            Bullet.Size = new Size(5, 5); // set the size to the bullet to 5 pixel by 5 pixel
            Bullet.Tag = "bullet"; // set the tag to bullet
            Bullet.Left = bulletLeft; // set bullet left
            Bullet.Top = bulletTop; // set bullet right
            Bullet.BringToFront(); // bring the bullet to front of other objects
            form.Controls.Add(Bullet); // add the bullet to the screen

            tm.Interval = speed; // set the timer interval to speed
            tm.Tick += new EventHandler(tm_Tick); // assignment the timer with an event
            tm.Start(); // start the timer

        }

        public void tm_Tick(object sender, EventArgs e)
        {
            // if direction equals to left
            if (direction == "left")
            {
                Bullet.Left -= speed; // move bullet towards the left of the screen
            }
            // if direction equals right
            if (direction == "right")
            {
                Bullet.Left += speed; // move bullet towards the right of the screen
            }
            // if direction is up
            if (direction == "up")
            {
                Bullet.Top -= speed; // move the bullet towards top of the screen
            }
            // if direction is down
            if (direction == "down")
            {
                Bullet.Top += speed; // move the bullet bottom of the screen
            }

            // if the bullet is less the 16 pixel to the left OR
            // if the bullet is more than 860 pixels to the right OR
            // if the bullet is 10 pixels from the top OR
            // if the bullet is 616 pixels to the bottom OR
            // IF ANY ONE OF THE CONDITIONS ARE MET THEN THE FOLLOWING CODE WILL BE EXECUTED

            if (Bullet.Left < 16 || Bullet.Left > 860 || Bullet.Top < 10 || Bullet.Top > 616)
            {
                tm.Stop(); // stop the timer
                tm.Dispose(); // dispose the timer event and component from the program
                Bullet.Dispose(); // dispose the bullet
                tm = null; // nullify the timer object
                Bullet = null; // nullify the bullet object
            }
        }
    }
}
```