

```

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 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

        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.Intersects(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.Intersects(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.Intersects(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

```

1
2 namespace zombieShooter
3 {
4     class bullet
5     {
6
7         // start the variable
8

```

```

9      public string direction; // creating a public string called direction
10     public int speed = 20; // creating a integer called speed and assigning a value of 20
11     PictureBox Bullet = new PictureBox(); // create a picture box
12     Timer tm = new Timer(); // create a new timer called tm.
13
14     public int bulletLeft; // create a new public integer
15     public int bulletTop; // create a new public integer
16
17     // end of the variables
18
19     public void mkBullet(Form form)
20     {
21         // this function will add the bullet to the game play
22         // it is required to be called from the main class
23
24         Bullet.BackColor = System.Drawing.Color.White; // set the colour white for the bullet
25         Bullet.Size = new Size(5, 5); // set the size to the bullet to 5 pixel by 5 pixel
26         Bullet.Tag = "bullet"; // set the tag to bullet
27         Bullet.Left = bulletLeft; // set bullet left
28         Bullet.Top = bulletTop; // set bullet right
29         Bullet.BringToFront(); // bring the bullet to front of other objects
30         form.Controls.Add(Bullet); // add the bullet to the screen
31
32         tm.Interval = speed; // set the timer interval to speed
33         tm.Tick += new EventHandler(tm_Tick); // assignment the timer with an event
34         tm.Start(); // start the timer
35
36     }
37
38     public void tm_Tick(object sender, EventArgs e)
39     {
40         // if direction equals to left
41         if (direction == "left")
42         {
43             Bullet.Left -= speed; // move bullet towards the left of the screen
44         }
45         // if direction equals right
46         if (direction == "right")
47         {
48             Bullet.Left += speed; // move bullet towards the right of the screen
49         }
50         // if direction is up
51         if (direction == "up")
52         {
53             Bullet.Top -= speed; // move the bullet towards top of the screen
54         }
55         // if direction is down
56         if (direction == "down")
57         {
58             Bullet.Top += speed; // move the bullet bottom of the screen
59         }
60
61         // if the bullet is less the 16 pixel to the left OR
62         // if the bullet is more than 860 pixels to the right OR
63         // if the bullet is 10 pixels from the top OR

```

```
64      // if the bullet is 616 pixels to the bottom OR
65      // IF ANY ONE OF THE CONDITIONS ARE MET THEN THE FOLLOWING CODE WILL BE
66 EXECUTED
67
68      if (Bullet.Left < 16 || Bullet.Left > 860 || Bullet.Top < 10 || Bullet.Top > 616)
69      {
70          tm.Stop(); // stop the timer
71          tm.Dispose(); // dispose the timer event and component from the program
72          Bullet.Dispose(); // dispose the bullet
73          tm = null; // nullify the timer object
74          Bullet = null; // nullify the bullet object
75      }
76  }
77 }
78 }
79
80
81
82
83
84
```