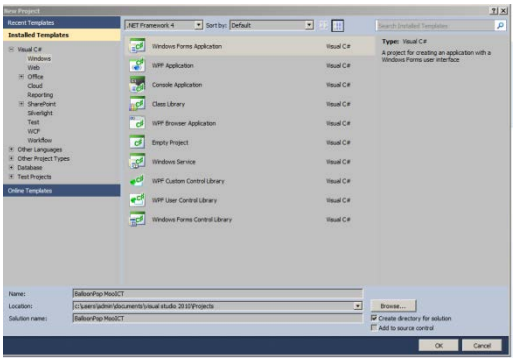
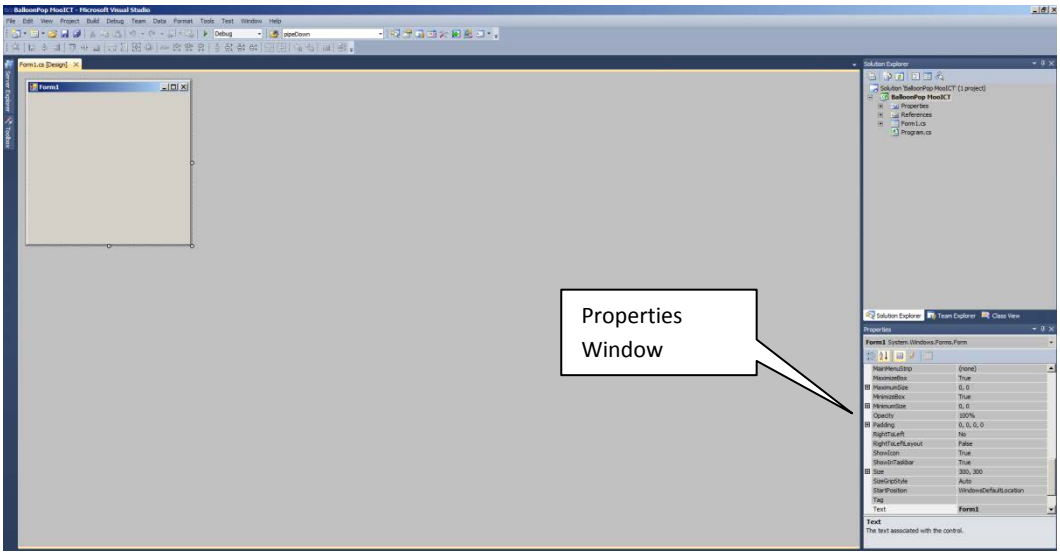
Start Visual Studio, start a new Windows Form project under the C# language, name the project **BalloonPop MooICT** and click OK.

Before you start - download the game assets from above or on MOOICT.COM to achieve the same result as us or you can use your own images if you want to.



Click OK.



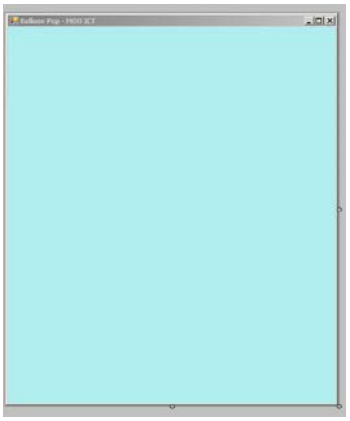
This is the empty form we are starting off with. We add all of our components to this form. We will change a few things to make the game look better.

In the properties Window change the following

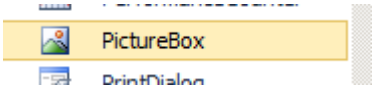
Back Color - **PaleTurquoise**

Size - **550, 650**

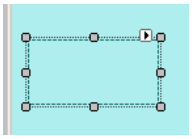
Text - **Balloon Pop - MOO ICT**


This is what the form currently looks like.

Look at the Tool Box - if the tool box is not present go to view and click on Tool Box

Find the picture box 

drag the picture box to the form.

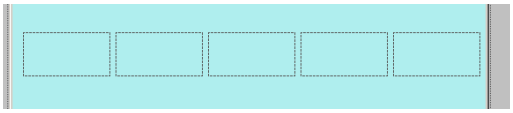
 While the picture box is active, look into the properties window and change the following.

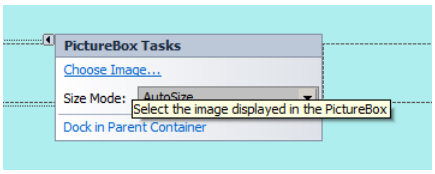
Back Color - **Transparent**

Size Moe - **Auto Size**

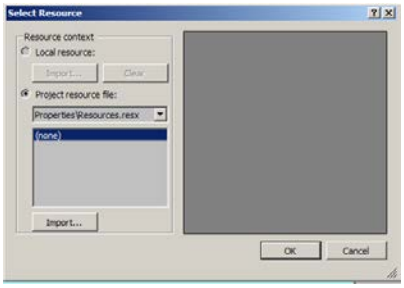
Tag - **Balloon**

Make 5 copies of the picture box on the form. You can copy and paste them to save time, if you copy the first one it will also copy the properties with it.

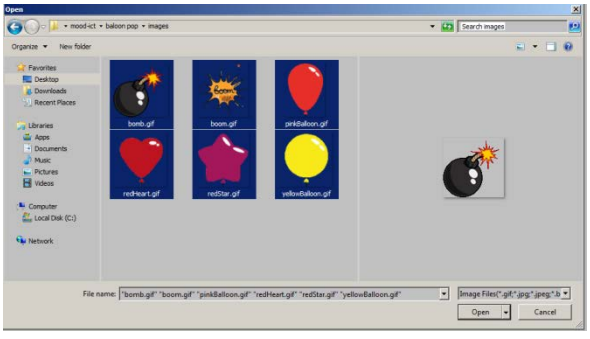
 Here is the picture boxes copied 5 times.



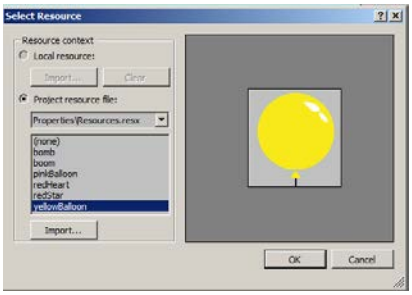
Click on that little triangle on top of the picture box [it appears when the picture box is clicked on] and click on choose image.



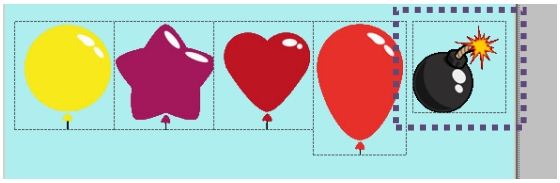
Click on Import under the Project resource file. Find the images you downloaded from MOOICT and import them all to this project.



Select all of the images at once and click OPEN.



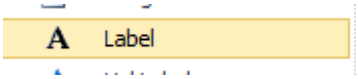
Choose an image from the list and click OK.

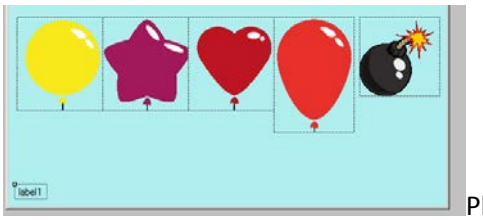

All of the images have been assigned to the picture boxes. Now CLICK on the BOMB picture and change the following to its properties.

Name - **bomb**

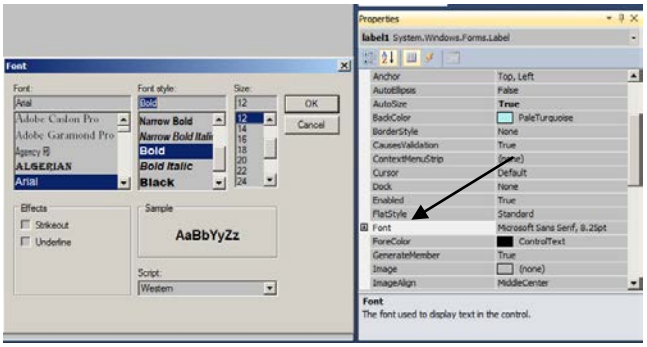
Tag - **bomb**

Look back at the tool box again and drag the label to the form.





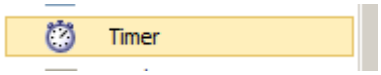
Place on the corner bottom left on the screen. While the label is active look at the properties window for the label.



Under the font option, choose Arial - BOLD - font size 12.

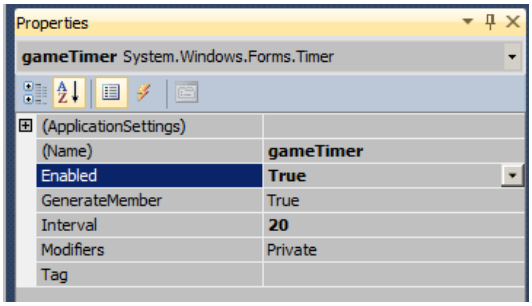
This is what the label looks like now.

Now for the final component to add to the game and this one is the main component of the game. It's the Timer.



Drag the time object to the form.

Timers are added to bottom of the form but not on the form itself.

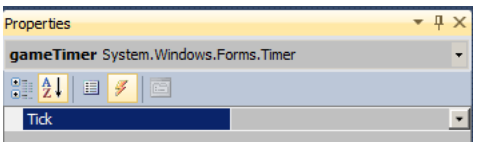


Change the following to the timers property window.

Name - **gameTimer**

Enabled - **True**

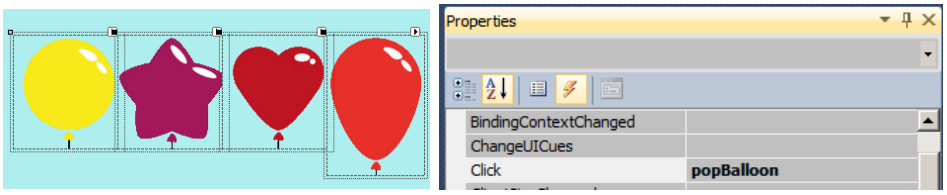
Interval - **20**



Click on the lightning bolt and it take you to the events window for that object. In the box type **gameEngine** and press enter.

Once you press enter it will take you to the code view, come back to the design view we need add a few more events before we get started coding.

Select the 4 Balloon on the screen

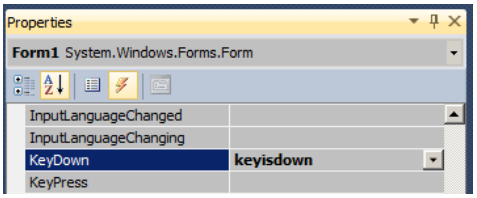


While the picture boxes are selected, under the events for the picture boxes find the click event and type **popBalloon** and press enter.

Now click on the bomb picture box and go back to the events window and type **popBomb** and press enter.

Now we need to add one more event to this game. This event will be linked directly to the FORM.

SO click on the form not on any of its components, while the form is active click on the events window and find the KEY DOWN event from the list.



Type keyisdown and press enter. This will conclude our events for this game. There is a lot but we need them, you understand right.

Let's take a look at the code view.

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

        private void gameEngine(object sender, EventArgs e)
        {

        }

        private void popBalloon(object sender, EventArgs e)
        {

        }

        private void popBomb(object sender, EventArgs e)
        {

        }

        private void keyisdown(object sender, KeyEventArgs e)
        {

        }

    }
}
```

We need to add one function to the code - its highlighted below. We will need to type up this function yourself since this is a custom function for this 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 BalloonPop_MooICT
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void gameEngine(object sender, EventArgs e)
        {

        }

        private void popBalloon(object sender, EventArgs e)
        {

        }

        private void popBomb(object sender, EventArgs e)
        {

        }

        private void keyisdown(object sender, KeyEventArgs e)
        {

        }

        private void resetGame()
        {

        }
    }
}
```

Let's start adding some code to our game.

Note - `// this is a comment line in the game. this you can type directly into the program and it will still work fine. Each line of code is explained right next to the line to help you further understand the code.`

First start with the variables.

```
        int speed = 5; // this integer holds the value 5
        int score = 0; // this integer holds the value 0
        Random rand = new Random(); // this will create a new instance of the random class called rand
        bool gameOver = false; // this Boolean will be used to check if the game is over or not
```

Above the public Form1() line add these variables to the code.

int speed = 5; this is a data type of integer, this integer will determine the speed of the objects in this game

int score = 0; this line is creating another integer which will store the score of the player. each time the player pops a balloon we will increase this by 1.

Random rand = new Random(); this line is importing the Random class called rand and creating a new instance of the class. The random class is responsible for creating a random number. Since we want to randomly place the balloons X and Y axis this will work just fine.

bool gameOver = false; this is a Boolean which will change between true and false when the is over and when it restarts.

```
        public Form1()
        {
            InitializeComponent();
            resetGame(); // invoke the reset game function.
        }
```

add the resetGame(); line under the initialize component line. This will run the reset function when the game starts.

```
        private void popBalloon(object sender, EventArgs e)
        {
            // is game over is false
            if (gameOver == false)
            {
                // then do the follow

                var baloon = (PictureBox)sender; // verify the picture box that sends the event

                baloon.Top = rand.Next(700, 1000); // change the top on a random location between 700 and 1000 pixels

                baloon.Left = rand.Next(5, 500); // change the left on a random location between 5 to 400 pixels

                score++; // increase the score by 1
            }
        }
```

This function controls the balloon picture boxes click event. All of the instructions inside the function only runs when the game over Boolean remains false. Meaning if the game is over then the click events will not run.

Once we verify the game is not over, we create a **var** called **baloon** which will contain the sender of this event. Since all 4 of the picture boxes are sending the same event we need to find which one was clicked, so inside the variable we will put the sender of this event. Once the event is clicked then we will change the TOP and LEFT of the balloon to a random location beneath the form. So it can scroll back up again (that will shown in the timer event later).

Lastly we will increase the score integer by 1.

```
        private void popBomb(object sender, EventArgs e)
        {
            // this event will trigger when the bomb picture box is clicked in the game
            // if game over is false
            if (gameOver == false)
            {
                // then do the following
                bomb.Image = Properties.Resources.boom; // change the picture to the boom picture
                gameTimer.Stop(); // stop the timer
                label1.Text += "  Game Over! -  Press Enter to retry"; // show game over text on the label
                gameOver = true; // change the game over boolean to true
            }
        }
```

This event will trigger each time the bomb is clicked. Firstly once again we check if the game is not over. then we change the image of the bomb picture box to the boom image in the resources. Then we stop the game timer which runs the

entire game. We will change the label1 text to Game over and change the game over boolean to true. This will stop any unwanted to event to fire when the game is not running.

```
        private void gameEngine(object sender, EventArgs e)
        {
            label1.Text = "Score: " + score; // show the score on the label

            // below is the for each loop that will check all of the picture boxes in this form
            foreach (Control X in this.Controls)
            {
                if (X is PictureBox)
                {
                    // move the picture boxes to the top
                    // each frame will trigger the picture boxes 5 pixels to the top
                    X.Top -= speed;

                    // if the picture box reaches the top
                    if (X.Top + X.Height < 0)
                    {
                        // the do the following
                        X.Top = rand.Next(700, 1000); // change the top to  random location between 700 to 1000 pixels
                        X.Left = rand.Next(5, 500); // change the left to a random location between 5 to 400 pixels
                    }

                    // if the picture box has a tag of Baloon AND it reaches the top of -50 pixels
                    if (X.Tag == "Balloon" && X.Top < -50)
                    {
                        // then do the following
                        gameTimer.Stop(); // stop the timer
                        label1.Text += "  Game Over! -  Press Enter to retry"; // show game over text on the label
                        gameOver = true; // change the game over boolean to true
                    }

                }
            }

            /**
             * Below is the if statement that will determine when to speed up
             * the balloons in the game.
             * If the score is EQUALS to or GREATER than 10 then change the speed to 8
             * If the score is EQUALS to or GREATER than 20 then change the speed to 16
             * if the score is EQUALS to or GREATER than 35 then change the speed to 20
             * This will increase the challenge in the game
             **/

            if (score >= 10)
            {
                speed = 8;
            }
            if (score >= 20)
            {
                speed = 16;
            }
            if (score >= 35)
            {
                speed = 20;
            }
        }
```

```
label1.Text = "Score: " + score; // show the score on the label
```

This line will show the score variable on the label 1 text. Since the timer is running every 20 milliseconds this will an accurate way to show the score of the game.

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

This line is starting the loop. We will check every component presented on the form. We are looking for some specific components in the form.

```
if (X is PictureBox)
```

When the loop is checking the components and finds picture boxes in the form it will execute the following instructions

```
X.Top -= speed;
```

Move the picture boxes towards the top. If we use -= means it will deduct the position of the picture box for the amount of the speed variable and move it closer towards 0. So if the picture was on 400 pixels from the top then every tick it will remove 5 from 400, this will give the illusion of the picture box or the balloon floating towards top of the screen.

```
// if the picture box reaches the top
                    if (X.Top + X.Height < 0)
                    {
                        // the do the following
                        X.Top = rand.Next(700, 1000); // change the top to  random location between 700 to 1000 pixels
                        X.Left = rand.Next(5, 500); // change the left to a random location between 5 to 400 pixels
```

```
            }
```

In this if statement we are checking if the picture box makes it to the top. Inside the if statement we are checking X.TOP which is always 0 plus X.Height so this will give us a value to reset the picture box to the bottom of the screen. Once that happens we are then moving the X.TOP to a random number between 700 and a 1000 pixels. We don't want all of the picture boxes to appear at the same time. We are doing the same for the X.LEFT we are randomizing the location of the X axis of the picture box to make this game more challenging otherwise the player can always anticipate where the balloons going to pop out from.

```
            // if the picture box has a tag of Balloon AND it reaches the top of -50 pixels
            if (X.Tag == "Balloon" && X.Top < -50)
            {
                // then do the following
                gameTimer.Stop(); // stop the timer
                label1.Text += "  Game Over! -  Press Enter to retry"; // show game over text on the label
                gameOver = true; // change the game over boolean to true
            }
```

Now we need to check if the balloons make it to the top of the screen. This if statement we are checking if the picture box has a TAG [remember we added this earlier to the picture boxes inside the properties window] of Balloon and its top if less than 50 then we stop the game, show the game over text on label 1 and change the game over Boolean to true.

If you are wondering why do the previous if statement for picture to change location it reached the top then the answer is we have a bomb image which goes to the top and has to reset, this if statement applies to the balloons but the once before applies to the bomb image because we need to make it appear randomly with the balloons.

```
            /**
             * Below is the if statement that will determine when to speed up
             * the balloons in the game.
             * If the score is EQUALS to or GREATER than 10 then change the speed to 8
             * If the score is EQUALS to or GREATER than 20 then change the speed to 16
             * if the score is EQUALS to or GREATER than 35 then change the speed to 20
             * This will increase the challenge in the game
             **/

            if (score >= 10)
            {
                speed = 8;
            }
            if (score >= 20)
            {
                speed = 16;
            }
            if (score >= 35)
            {
                speed = 20;
            }
```

In the code above we are making the game a little more challenging. This will happen through checking how much the player scored thus far in the game. First if the player scored equals to or greater than 10 then we change the speed from 5 to 8. Remember the speed is what's determining how much the picture boxes move up by. Then we check if they score more than 20 then we change the speed to 16 and finally if they've scored more than 35 then we change the speed to 20.

You can set your own speed limits and score limits in the game to make it more interesting.

```
        private void resetGame()
        {
            // below is the for each loop to check for all of the picture boxes in this form
            // once it is found it will randomly assign their places below the the form
            // so it looks like its coming up from bottom to the top
            foreach (Control X in this.Controls)
            {
                if (X is PictureBox)
                {
                    X.Top = rand.Next(700, 1000);
                    // x being the picture box we are changing the the top from anywhere betwene 700 to 1000 pixels
                    X.Left = rand.Next(5, 500);
                    // x being the picture box we are changing the left side from anywhere between 5 to 400 pixles
                }
            }

            // reset the variables for the game
            bomb.Image = Properties.Resources.bomb; // change the picture to the boom picture
```

```
            speed = 5; // reset the speed back to 5
            score = 0; // reset the score to 0
            gameOver = false; // change game over to false
            label1.Text = "Score: " + score; // show the score on the label
            gameTimer.Start(); // start the time so the game can begin
        }
```

This is the game reset function. You will notice that the code is somewhat same to the once in the game engine function. The main difference here is that this function only runs once it's called. We call in the beginning of the code and we will also call it when the player decides to play again to beat their old score.

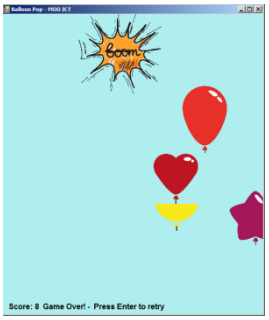
First we are running the same for each loop for all of the components in the form, if we find the picture boxes then we randomize their locations on the screen.

Then we reset each of the variables and start the game timer. Notice we are also changing the bomb image to the main bomb image because if the player accidentally clicked that image it stay in the boom image. So we need to reset this too.

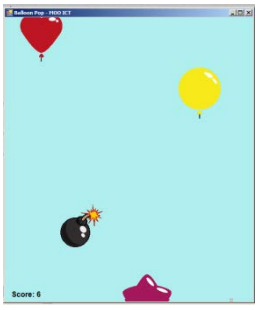
```
        private void keyisdown(object sender, KeyEventArgs e)
        {
            // if the player presses enter key and the game is over then we can run the reset function
            if(e.KeyCode == Keys.Enter && gameOver == true)
            {
                resetGame();
            }
        }
```

This is the key is down function. In this function we checking the if statement for two separate conditions. We want to check if the player has pressed the enter key on the keyboard and the game over boolean is true. We did this because otherwise in mid game player can press the enter key and lose their progress. So to only invoke the reset function when the game is truly over we are using two conditional if statement. If both of them return true then we can invoke the reset function and start again. Simple right.

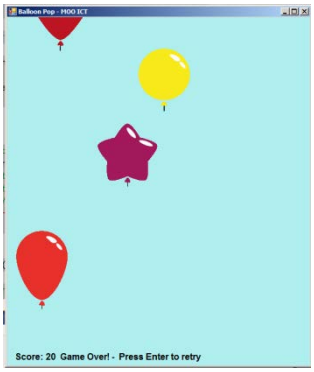
Now double check the code and test out the game. Make sure you haven't missed any curly brackets or made any spelling errors during this time. If you have it come up on the error console and you can check the specific line this error has happen. If not lets debug and make the magic happen.



Balloons are floating up and I also clicked the bomb picture and it changed to BOOM. Nice



The game has successfully reset when I pressed enter.



The game definitely speeds up when you score up, almost gave me anxiety scoring to 20. If you followed it thus far well done and MOO Out.

Full Source Code for the Balloon Pop 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 BalloonPop_MooICT
{
    public partial class Form1 : Form
    {

        int speed = 5; // this integer holds the value 5
        int score = 0; // this integer holds the value 0
        Random rand = new Random(); // this will create a new instance of the random class called rand
        bool gameOver = false; // this boolean will be used to check if the game is over or not

        public Form1()
        {
            InitializeComponent();
            resetGame(); // invoke the reset game function.
        }

        private void gameEngine(object sender, EventArgs e)
        {
            label1.Text = "Score: " + score; // show the score on the label

            // below is the for each loop that will check all of the picture boxes in this form
            foreach (Control X in this.Controls)
            {
                if (X is PictureBox)
                {
                    // move the picture boxes to the top
                    // each frame will trigger the picture boxes 5 pixels to the top
                    X.Top -= speed;

                    // if the picture box reaches the top
                    if (X.Top + X.Height < 0)
                    {
                        // the do the following
                        X.Top = rand.Next(700, 1000); // change the top to  random location between 700 to 1000 pixels
                        X.Left = rand.Next(5, 500); // change the left to a random location between 5 to 400 pixels
                    }

                    // if the picture box has a tag of Balloon AND it reaches the top of -50 pixels
                    if (X.Tag == "Balloon" && X.Top < -50)
                    {
                        // then do the following
                        gameTimer.Stop(); // stop the timer
                        label1.Text += "  Game Over! -  Press Enter to retry"; // show game over text on the label
                        gameOver = true; // change the game over boolean to true
                    }

                }
            }

            /**
             * Below is the if statement that will determine when to speed up
             * the balloons in the game.
             * If the score is EQUALS to or GREATER than 10 then change the speed to 8
             * If the score is EQUALS to or GREATER than 20 then change the speed to 16
             * if the score is EQUALS to or GREATER than 35 then change the speed to 20
             * This will increase the challenge in the game
             **/

            if (score >= 10)
            {
                speed = 8;
```

```
            }
            if (score >= 20)
            {
                speed = 16;
            }
            if (score >= 35)
            {
                speed = 20;
            }
        }

        private void popBalloon(object sender, EventArgs e)
        {
            // is game over is false
            if (gameOver == false)
            {
                // then do the follow

                var baloon = (PictureBox)sender; // verify the picture box that sends the event

                baloon.Top = rand.Next(700, 1000); // change the top on a random location between 700 and 1000 pixels

                baloon.Left = rand.Next(5, 500); // change the left on a random location between 5 to 400 pixels

                score++; // increase the score by 1
            }
        }

        private void popBomb(object sender, EventArgs e)
        {
            // this event will trigger when the bomb picture box is clicked in the game
            // if game over is false
            if (gameOver == false)
            {
                // then do the following
                bomb.Image = Properties.Resources.boom; // change the picture to the boom picture
                gameTimer.Stop(); // stop the timer
                label1.Text += "  Game Over! -  Press Enter to retry"; // show game over text on the label
                gameOver = true; // change the game over boolean to true
            }
        }

        private void resetGame()
        {
            // below is the for each loop to check for all of the picture boxes in this form
            // once it is found it will randomly assign their places below the the form
            // so it looks like its coming up from bottom to the top
            foreach (Control X in this.Controls)
            {
                if (X is PictureBox)
                {
                    X.Top = rand.Next(700, 1000);
                    // x being the picture box we are changing the the top from anywhere betwene 700 to 1000 pixels
                    X.Left = rand.Next(5, 500);
                    // x being the picture box we are changing the left side from anywhere between 5 to 400 pixles
                }
            }

            // reset the variables for the game
            bomb.Image = Properties.Resources.bomb; // change the picture to the boom picture
            speed = 5; // reset the speed back to 5
            score = 0; // reset the score to 0
            gameOver = false; // change game over to false
            label1.Text = "Score: " + score; // show the score on the label
            gameTimer.Start(); // start the time so the game can begin
        }

        private void keyisdown(object sender, KeyEventArgs e)
        {
            // if the player presses enter key and the game is over then we can run the reset function
            if (e.KeyCode == Keys.Enter && gameOver == true)
            {
                resetGame();
            }
        }
    }
}
```