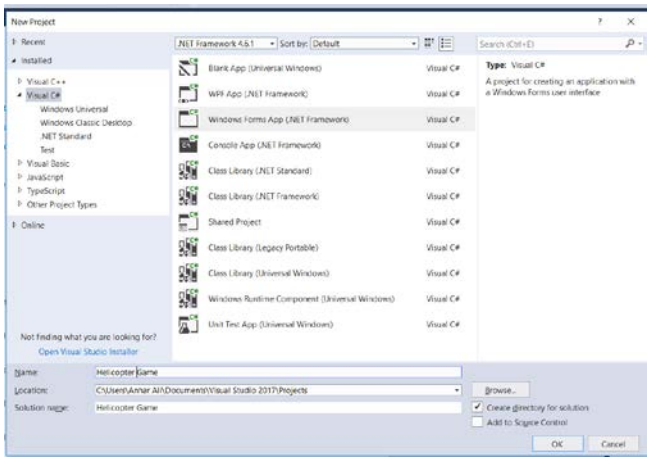
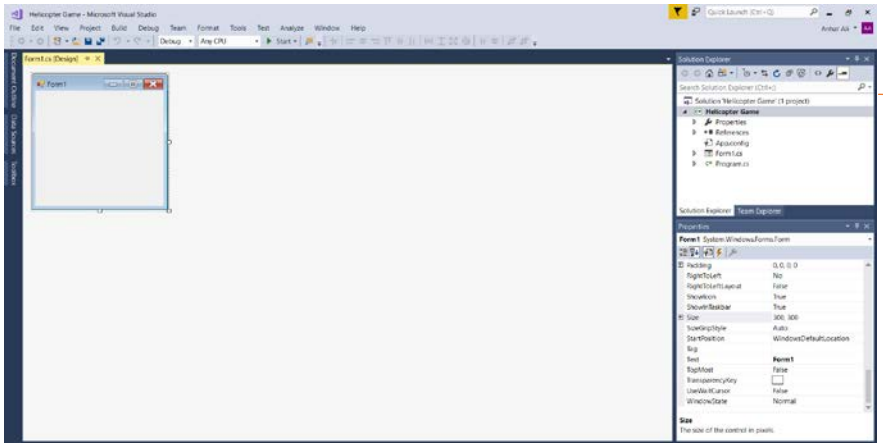
C# Tutorial – Create a helicopter flying and shooting game in visual studio

In this tutorial we will create a fun little helicopter game in visual studio. You will be flying the helicopter which can shoot at UFO's while dodging obstacles. You can control the helicopter by pressing up and down to shoot press the space bar. We will create this full project using Visual Studio and C# programming language. All the game assets are available below so make sure you download it to follow along this tutorial.

Start Visual Studio, create a new project called Helicopter Game and press OK





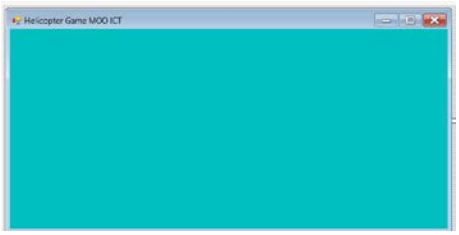
In this form Properties window change the following for this game

Back Colour – 0, 192, 192

Size – 813, 411

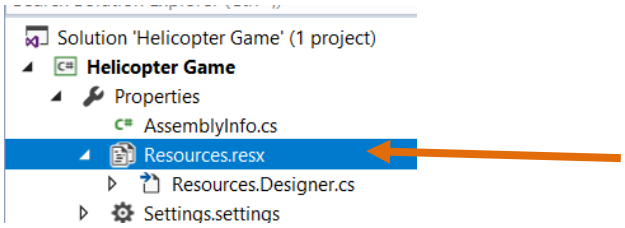
Text – Helicopter Game

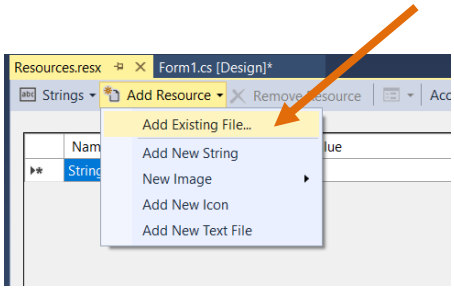
The result will be the following



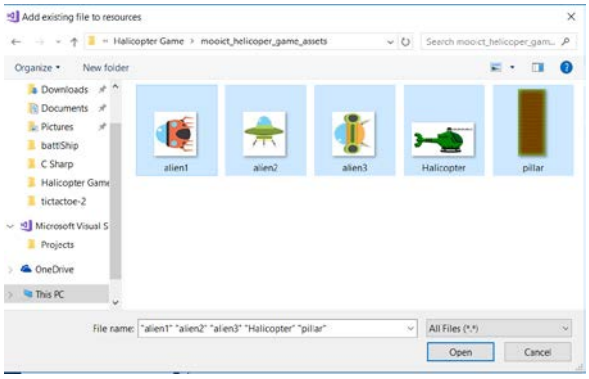
Now import the resources from MOOICT to the game.

Under the properties menu in the solutions explorer, double click on the **resources.resx** file

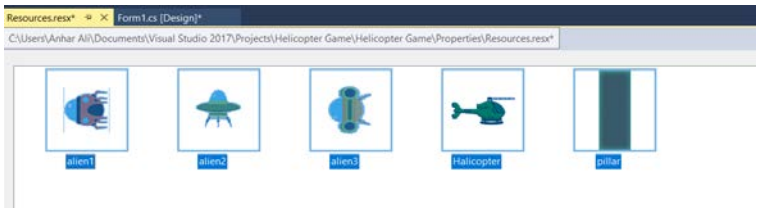




Click on the Add resources drop down menu and click on Add existing file

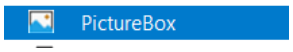


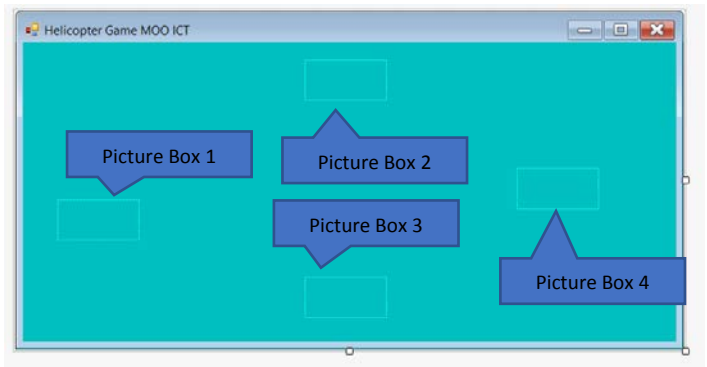
Highlight all of the images from the folder and click OPEN



This will add them all to the game. Now save the files [either press the save all button or press CTRL + S] and go back to the design view.

Now from the tool box add 4 picture boxes to the form

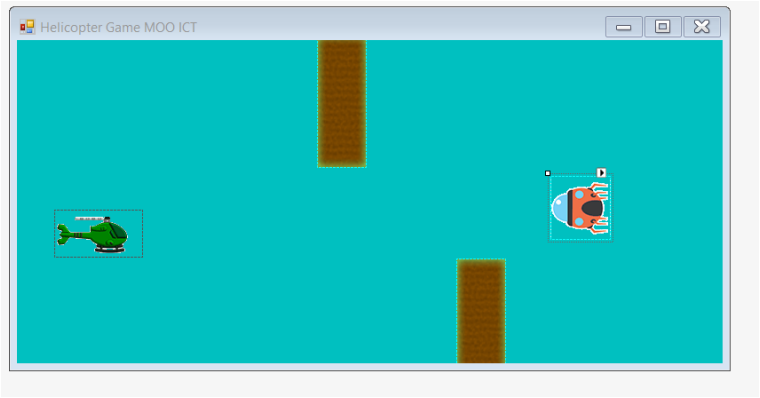




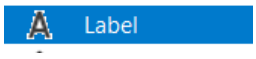
We will need the make the following changes to these picture boxes property menu.

PictureBox1 **Name – player** **Back Colour – transparent** **Image – helicopter image** **Size Mode – Auto Size**
PictureBox2 **Name – pillar1** **Image – pillar image** **Location – 338, -6** **Size – 56, 150** **Size Mode – Stretch Image**
PictureBox 3 **Name – pillar2** **Image – pullar image** **Location – 495, 246** **Size – 56, 146** **Size Mode – Stretch Image**
PictureBox 4 **Name – ufo** **Back Colour – Transparent** **Image – alien1 image** **Size Mode – Auto Size**

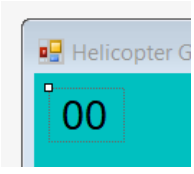
Final result



Now we need to add a label to the left top corner of the screen to keep score of the game.

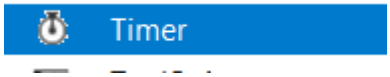


Drag and drop a label component to the form



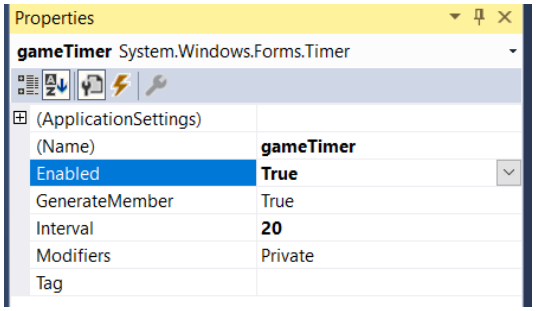
Change the text to 00 and change the font option in the properties window for the label to size 14.

Now lets add the final component, a Timer



Drag and drop the timer to the form

Make the following changes to the timer properties.

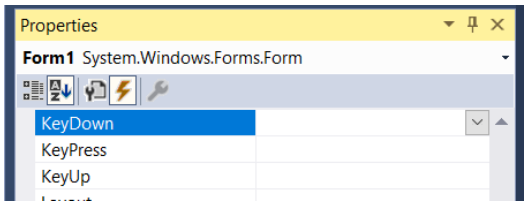


Change the name to **gameTimer** (one word), Enabled True, Interval 20.

Adding events to the game –

We need key down, key up and time tick event.

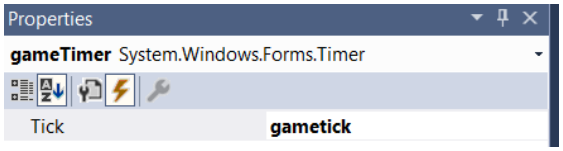
Click on the Form and in the properties window click on that little lightning bolt icon which will take you to the events window.



Find the **KeyDown** option type **keyisdown** and press enter. This will take you the code view, come back to the design view.

Find the **KeyUp** option type **keyisup** and press enter. This will take you to the code view, come back to the design view.

Click on the timer and go to the events window, in the **Tick** option type **gametick** and press enter.



This is the game code so far.

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

namespace Helicopter_Game
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
```

```
        }

        private void keyisdown(object sender, KeyEventArgs e)
        {

        }

        private void keyisup(object sender, KeyEventArgs e)
        {

        }

        private void gametick(object sender, EventArgs e)
        {

        }
    }
}
```

So far we have the above empty events. So we are going to start adding the variables and some custom functions for the game.

Add the highlighted code below to 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.Threading.Tasks;
using System.Windows.Forms;

namespace Helicopter_Game
{
    public partial class Form1 : Form
    {

        // enter the variables
        bool goup; // this is a boolean to allow player to go up
        bool godown; // this is a boolean to allow player to go down
        bool shot = false; // this will check if the player has shot any bullets
        int score = 0; // this is a integer for player to keep score
        int speed = 8; // this is the speed of obstacles and ufos
        Random rand = new Random(); // this is the random class to generate a random number
        int playerSpeed = 7; // this interger will control how fast the player moves
        int index; // this is a empty integer which will be used to change the UFO images

        public Form1()
        {
            InitializeComponent();
        }

        private void keyisdown(object sender, KeyEventArgs e)
        {

        }

        private void keyisup(object sender, KeyEventArgs e)
        {

        }

        private void gametick(object sender, EventArgs e)
        {

        }
```

```
        private void changeUFO()
        {

        }

        private void makeBullet()
        {

        }
    }
}
```

In this highlighted code above we have added the necessary variables to the game and also in the bottom of the code we have created a custom function called change UFO and make bullet. Lets explore what we have done

```
        // enter the variables
        bool goup; // this is a boolean to allow player to go up
        bool godown; // this is a boolean to allow player to go down
        bool shot = false; // this will check if the player has shot any bullets
        int score = 0; // this is a integer for player to keep score
        int speed = 8; // this is the speed of obstacles and ufos
        Random rand = new Random(); // this is the random class to generate a random number
        int playerSpeed = 7; // this interger will control how fast the player moves
        int index; // this is a empty integer which will be used to change the UFO images
```

```
// The green text you see next to the lines are comments, these are used to enhance deeper understanding for the code written it also helps to find what you are looking for in the code. It's a good practice to always comment your code.
```

All of the variables above are global variables meaning they can be accessed from any function in the game and we can change their values.

Bool is short for Boolean, we have two of them one is go up and one is go down. We will be using them to control the players movement in the game. Since they can only have two values true and false its all we need for this process. Shot Boolean will be used so the player doesn't just hold down the shoot button and automatically shoot down all of the aliens, we require them to press the button and only shoot a single bullet. By using a shot Boolean we will be able to set and reset the bullet numbers.

Int is short for integer. We have several integers in this game such as score, speed, player speed and index. Score will be increased each time you kill a UFO, speed will be used to determine how fast the UFO and on screen obstacles are moving, player speed will control the player movement and index we will use in the change UFO function to change the UFO images when they are shot down in the game.

Random rand is a the instance of Random Class which helps generate a random number between two values for examples in this game we will need to spawn the UFO in different places in the game so Random class can help us make this game more unpredictable. When the player shoots an UFO it will spawn back of the form in a different location than before in which the player must manoeuvre to shoot it again. If we had to hard code it ourselves the game will end up being boring.

```
        private void changeUFO()
        {

        }
```

This function is declared as private because we don't need to access it from any where else. We will populate it later on when we have done some of our crucial parts in this game . Please make sure you pay extra attention to the CURLY brackets because they are important and without them it will throw an error in this game.

```
        private void makeBullet()
        {

        }
```

This empty function will be used when the player is shooting the bullets, we will dynamically create the bullets and populate the screen with them. The timer object will then animate the bullet and speed them across the screen. By creating different function to carry out different instructions helps us keep the code organised and it's a very good practise to learn and implement into your own projects.

Adding Code to the key down event

The code is explained in the comments –

```
        private void keyisdown(object sender, KeyEventArgs e)
        {
            if (e.KeyCode == Keys.Up)
            {
                // if the player has pressed down the up key
                // we change the go up to true
                goup = true;
            }
            if(e.KeyCode == Keys.Down)
            {
                // if the player has pressed down the down key
                // we change the go down to true
                godown = true;
            }

            if(e.KeyCode == Keys.Space && shot == false)
            {
                // if the player has pressed down space key
                // AND shot boolean is false when they did
                // then we run the make bullet function
                // and change the shot from false to true
                makeBullet();
                shot = true;
            }
        }
```

Key up event

Code is explained in the comments –

```
        private void keyisup(object sender, KeyEventArgs e)
        {
            if (e.KeyCode == Keys.Up)
            {
                // if the player has left the up key
                // change go up to false
                goup = false;
            }
            if (e.KeyCode == Keys.Down)
            {
                // if the player has left the down key
                // change go down to false
                godown = false;
            }

            if (shot == true)
            {
                //if shot variable is true
                // we change it false so the player will have to shoot again
                // for more bullet.
                shot = false;
            }
        }
```

Change UFO function –

Code is explained with comments –

```
        private void changeUFO()
        {
            index += 1; // increase index by 1

            if (index > 3)
            {
                // if indexes value is greater than 3
                // set it back to 1
                index = 1;
            }

            // we will use the switch statement to switch between alien images
            // by using the number in index we can switch them effectively
            // when the numbers in index change this switch statement will follow
            switch(index)
            {
                // if the number in index is 1
                // then we will show the alien 1 skin on UFO picture Box
                case 1:
                    ufo.Image = Properties.Resources.alien1;
                    break;
                // if the number in index is 2
                // then we will show the alien 2 skin on UFO picture Box
                case 2:
                    ufo.Image = Properties.Resources.alien2;
                    break;
                // if the number in index is 3
                // then we will show the alien 3 skin on UFO picture Box
                case 3:
                    ufo.Image = Properties.Resources.alien3;
                    break;
            }
        }
```

Make bullet function –

Code is explained in comments

```
    private void makeBullet()
        {
            PictureBox bullet = new PictureBox();
            // create a new picture box class to the form

            bullet.BackColor = System.Drawing.Color.DarkOrange;
            // set the colour of the bullet to dark organge

            bullet.Height = 5;
            // set bullet height to 5 pixels

            bullet.Width = 10;
            // set bullet width to 10 pixels

            bullet.Left = player.Left + player.Width;
            // bullet will place in front of player object

            bullet.Top = player.Top + player.Height / 2;
            // bullet will be middle of player object

            bullet.Tag = "bullet";
            // set the tag for the object to bullet

            this.Controls.Add(bullet);
```

```
            // finally adding the picture box bullet to the scene
        }
```

Game time function

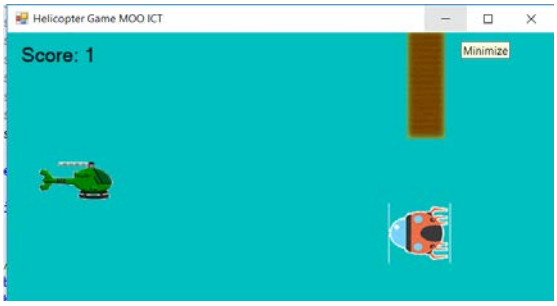
Code is explained in the comments –

```
private void gametick(object sender, EventArgs e)
        {
            // move pillar 1 towards the left of the screen
            pillar1.Left -= speed;

            // move pillar 2 towards the left of the screen
            pillar2.Left -= speed;

            // move ufo towards the left of the screen
            ufo.Left -= speed;

            // show the score on label 1
            label1.Text = "Score: " + score;


            if (goup)
            {
                // if go up is true then move the player up the screen
                // notice its minus equals means it will deduct from the top location
                // thus moving the player upwards
                player.Top -= playerSpeed;
            }

            if(godown)
            {
                // if go down is true then move the player down the screen
                // notice its plus equals means it will add to the top location
                // thus moving the player downwards
                player.Top += playerSpeed;
            }


            if(pillar1.Left < -150)
            {
                // if pillar 1 has gone past -150 which is off the screen
                // then move it to 900 pixels to the right of the screen
                // it will appear to have a continuous motion from right to left
                pillar1.Left = 900;
            }

            if(pillar2.Left < -150)
            {
                // if pillar 2 has gone past -150 which is off the screen
                // then move it to 1000 pixels to the right of the screen
                // it will appear to have a continuous motion from right to left
                pillar2.Left = 1000;
            }

            // the two || symbols represent the OR option in If statements
            // the below if statement is logically checking the following
            // if UFO has left the screen towards the left
            // OR
            // player has collided with the UFO object on screen
            // OR
            // player has collided with pillar 1 object
            // OR
            // player has collided with pillar 2 object
            // then follow the instructions inside the statement
            // we are able to check multiple conditions at the if statement
```

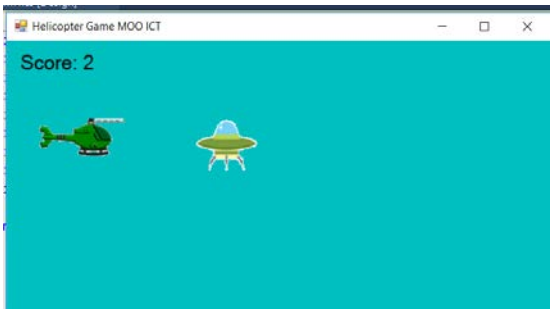
```
        if(ufo.Left < -5 ||
            player.Bounds.IntersectsWith(ufo.Bounds) ||
            player.Bounds.IntersectsWith(pillar1.Bounds) ||
            player.Bounds.IntersectsWith(pillar2.Bounds)
            )
        {
            // if one of the above is true then we stop the timer
            gameTimer.Stop();
            // the game will show the final score to the player in a message box
            MessageBox.Show("You failed the mission, you Killed " + score + " Ufo's");
        }

        // below is a for loop thats checking the components in this form
        // first we created a valiable called X in this form
        // x will be linked to the bullet object
        // it will find out if the bullet object exist

        foreach(Control X in this.Controls)
            {
            // if X is a picture box object AND it has a tag of bullet
            // then we will follow the instructions within

            if(X is PictureBox && X.Tag == "bullet")
            {
                // move x towards the right of the screen
                X.Left += 15;

                // if x has left the screen towards the right
                // x's location is greater than 900 pixels from the screen
                if(X.Left > 900)
                {
                    // then remove x from display
                    this.Controls.Remove(X);

                    // dispose the x from the application
                    // we use the dispose method so it doesn't leak memory later on
                    X.Dispose();
                }

                // below we will check if X collides with the UFO object
                if(X.Bounds.IntersectsWith(ufo.Bounds))
                {
                    // is X collides with the UFO object

                    // add 1 to the score
                    score += 1;

                    // remove the bullet from the screen
                    this.Controls.Remove(X);

                    // dispose the bullet from the program
                    X.Dispose();

                    // move the UFO object 1000 pixels off the screen
                    ufo.Left = 1000;

                    // generate a random vertical location for the UFO
                    ufo.Top = rand.Next(5, 330) - ufo.Height;

                    // run the change UFO function it appears like a different UFO
                    changeUFO();
                }
            }
        }
    }
```

That’s the game so far. Now lets try to debug it and see if everything works

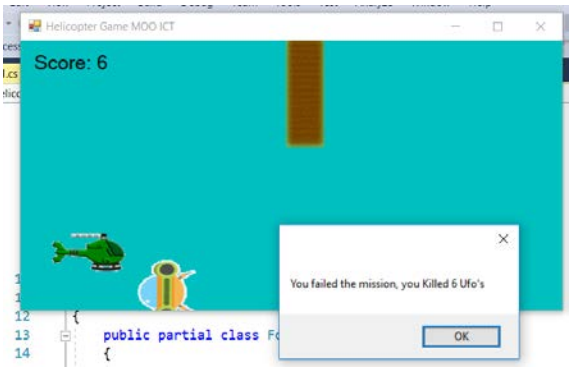
You can click on this start button on the tool bar or you can press F5 to debug the game.



The starts and the objects are moving towards the left. I can shoot the UFO’s on screen and its keeping track of my score.



When I shoot the UFO it respwns and changes its image as programmed in the changeCPU function.



Finally when I bump into the UFO the game ends and it shows the message box.

Well done getting this far in to the game. The idea behind these tutorials we do at MOOICT isn’t so you can get good at following, now you should change this game to your own ideas see if you can add more enemies, even more obstacles, add a restart button. All of these tasks are achieveable.

Most importantly have fun doing them. Don't get discouraged if an error shows up, thats how we learn. If there is an error go back to the code and check against this tutorial.

Moo Out