C# Tutorial Create a side scrolling platform game in visual studio

n this tutorial we will show you how to create a simple yet efficient side scrolling game only using Visual Studio toolbox components using the c# programming language. We have create another c# platform game tutorial before where you played the game in a static level for this one, we thought we can do better by adding the side scrolling and platform elements together. The main objective of this game, you the player are in a level where there is a locked door, you need to collect the key from the other end of the level while collecting some of the coins and not dropping off the platforms. Once you collect the key you are able to open the door and complete the level. Premise is fairly simple and it's a lot of fun to make, let get started.

note – games development is a lot about trial and error, don't be afraid to of the errors or the game breaking, while doing this tutorial it seems a lot easier to follow through but if you want to add more functionalities to it, don't be afraid to try it out, if the worse happens this tutorial is here to help other than that you can try to make anything you want.

Lesson Objectives

1. Create a full side scrolling platform game in visual studio with c# programming language
2. Use several picture boxes and control them using timers
3. Use Loops to easily identify the platforms, coins, keys and doors
4. Create gravity and jump force in visual studio
5. Using Key down and Key up events to control the character
6. Smoothly scrolling the background, items and platforms as the player moves between left or right

Above is the background image, this is a large image with a width of 2000 pixels and height of 480 pixels. This is the background image for the game this will be used to make the game on top of and we will scroll this image left to right.


This is the coin image, this is an animated coin gif

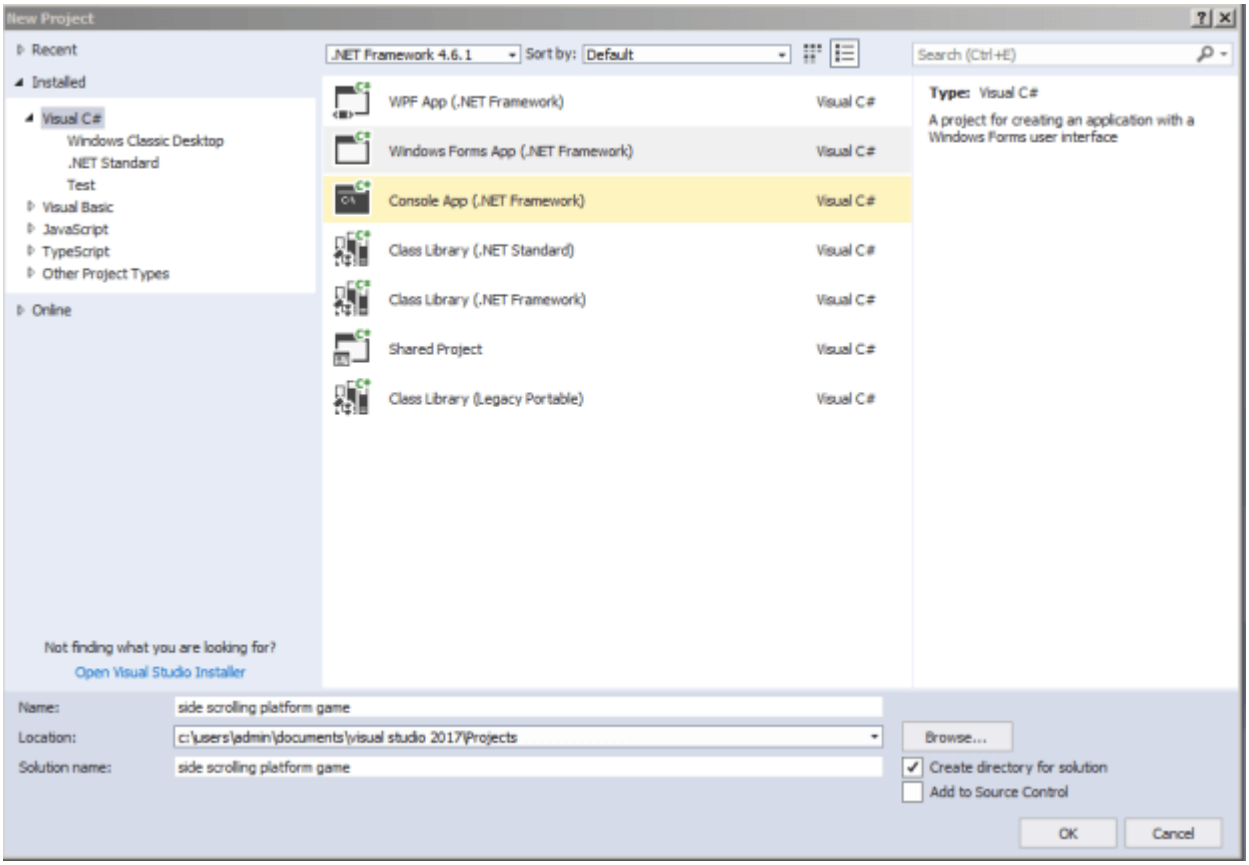

This is the door closed image, this will be default view of the door

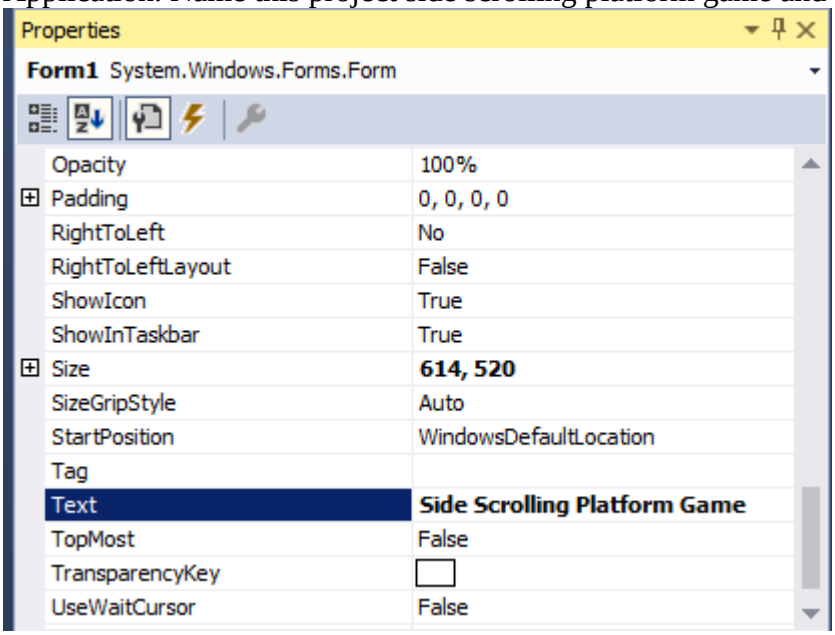

This is the door open image, when the player collects the key and collides with the door it will change to this image

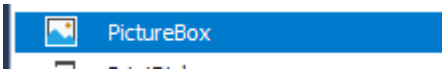

This is the key image.


This is the platform image. All the platforms in the game will have this as their background image.

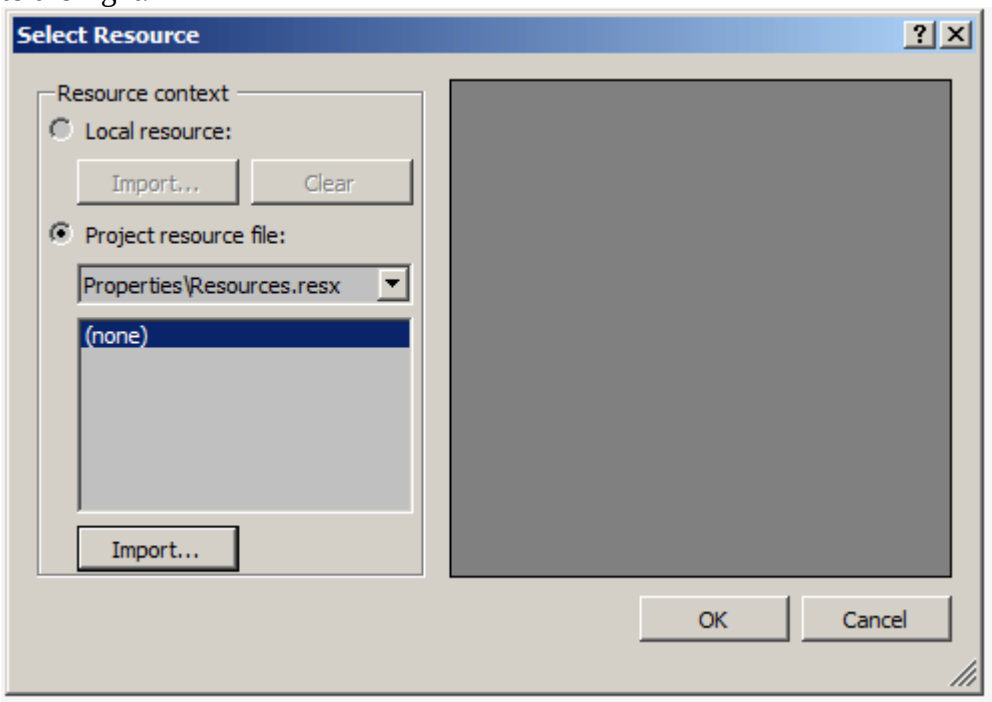
7. If you are new to games development, then let me let you in a secret that is games are illusions on a computer system, when you see Mario or sonic moving through a level they are not necessarily moving themselves the environment is moving towards them giving them an illusion of movement. We are going to do something similar to that in this tutorial. When the player will press left or right we will move the character little bit but we will move the environment more towards them which will give the player an ILLUSION of movement.

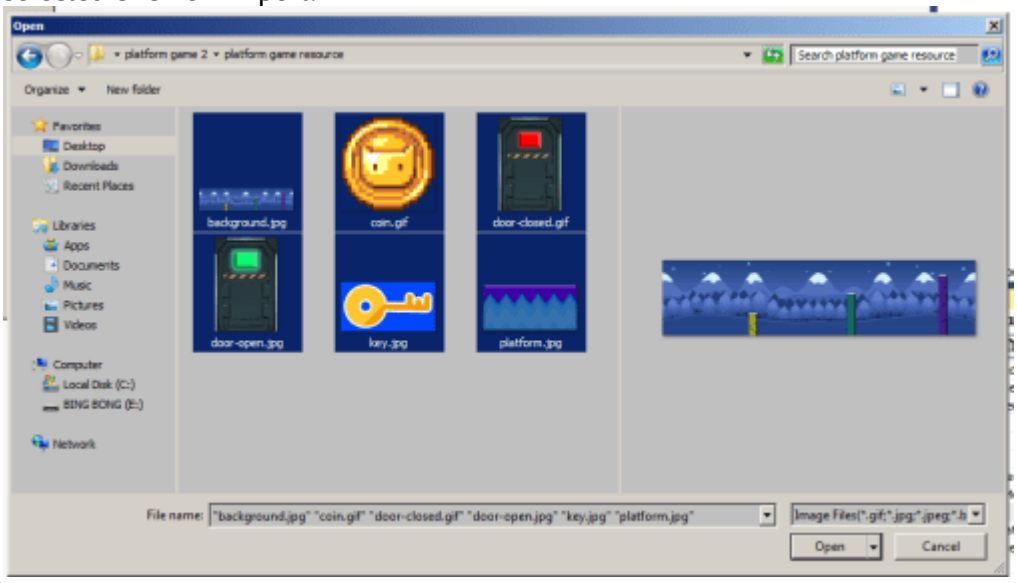
8. 

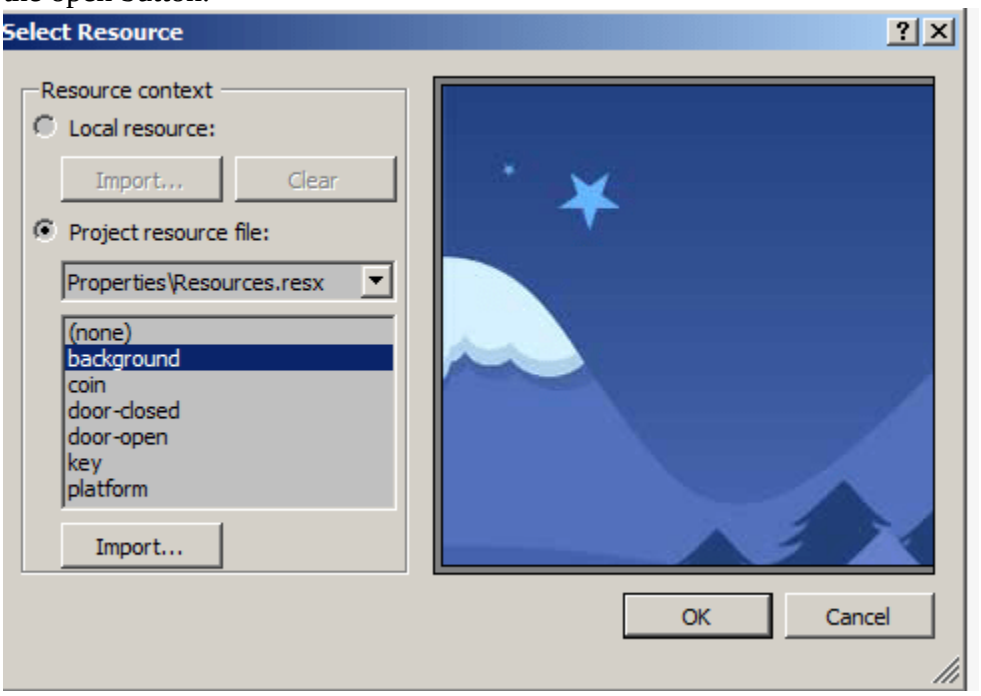
9. Create a new project in visual studio, Make sure its Visual C# and Windows Form Application. Name this project side scrolling platform game and click OK.
10. 

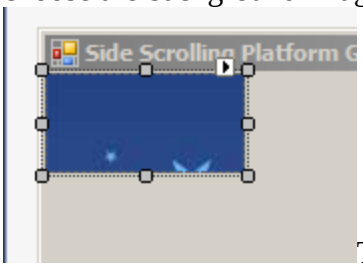
11. In the properties window change the size to 614, 520 and the text to “Side Scrolling Platform Game”.
12. 

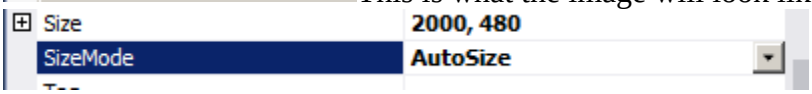
13. From the toolbox drag and drop a picture box to the form. This will be used as the main background for our side scrolling game. Once you dropped it on the form please change the following in its properties window.
14. **Name: background**
15. **Location: 0,0**
16. 

17. In the same properties window find the option for image and click on the 3 dotted button to the right.
18. 

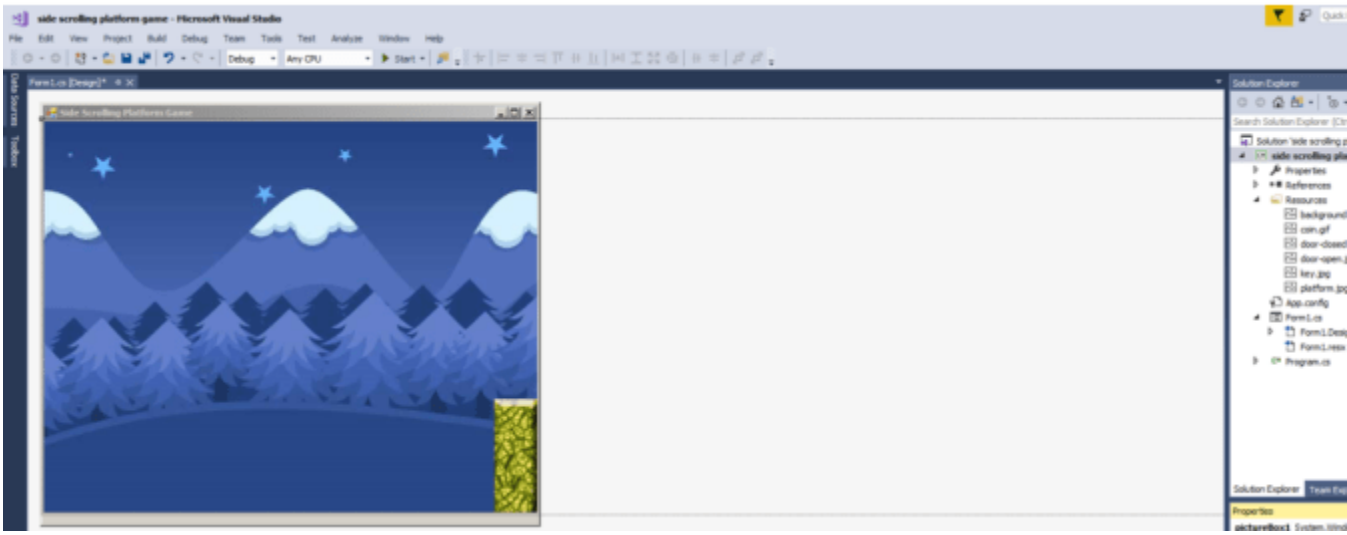
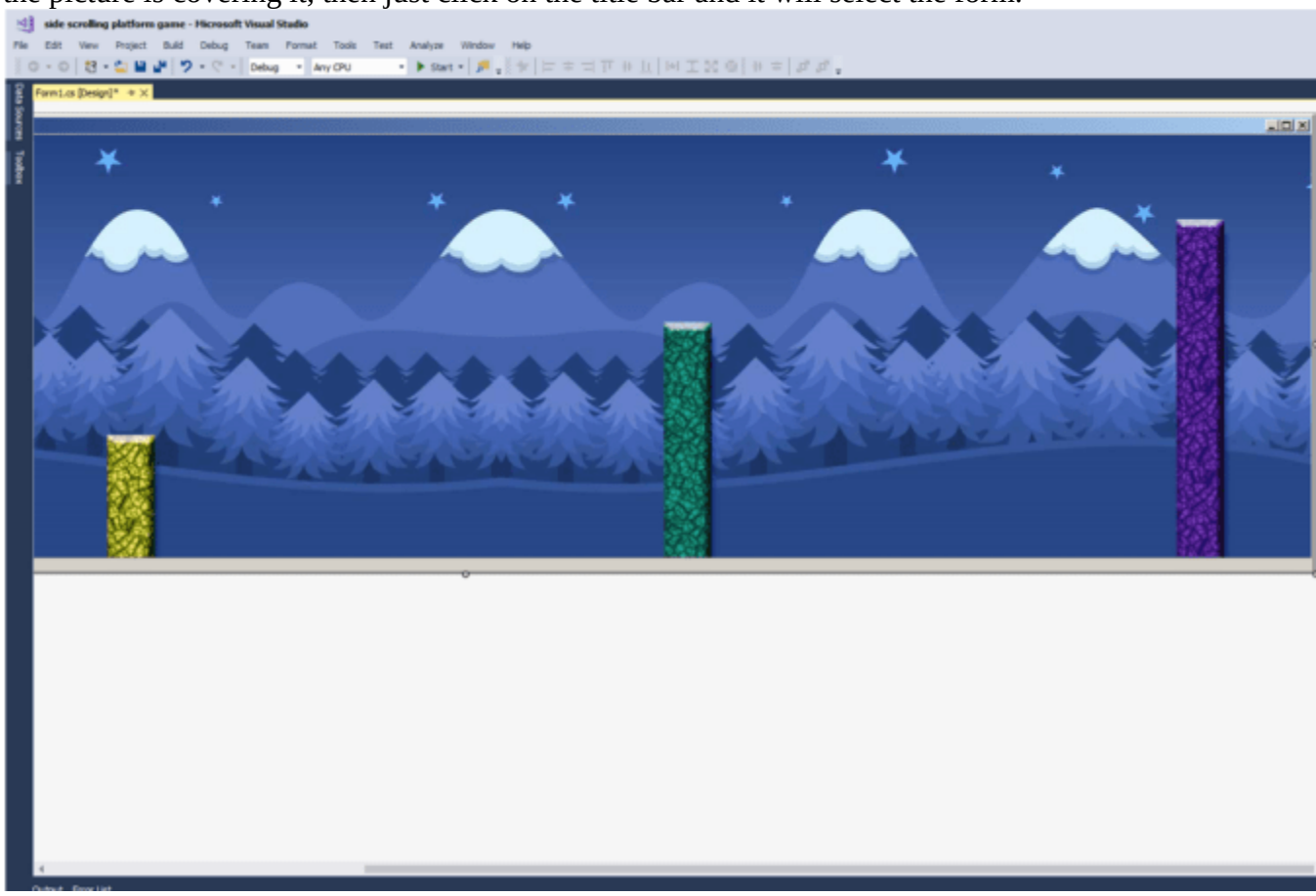
19. Once clicked you will see this window. This is the resource selection window it allows us to import images and other files to the project. While the Project Resource File option is selected click on import.
20. 

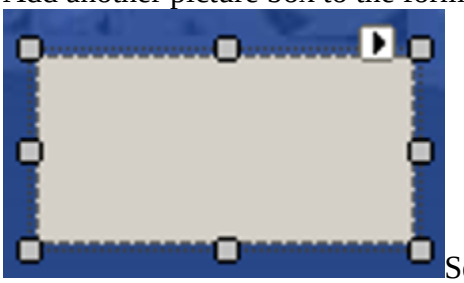
21. Navigate to the images you downloaded from MOOICT, select all the files and click on the open button.
22. 

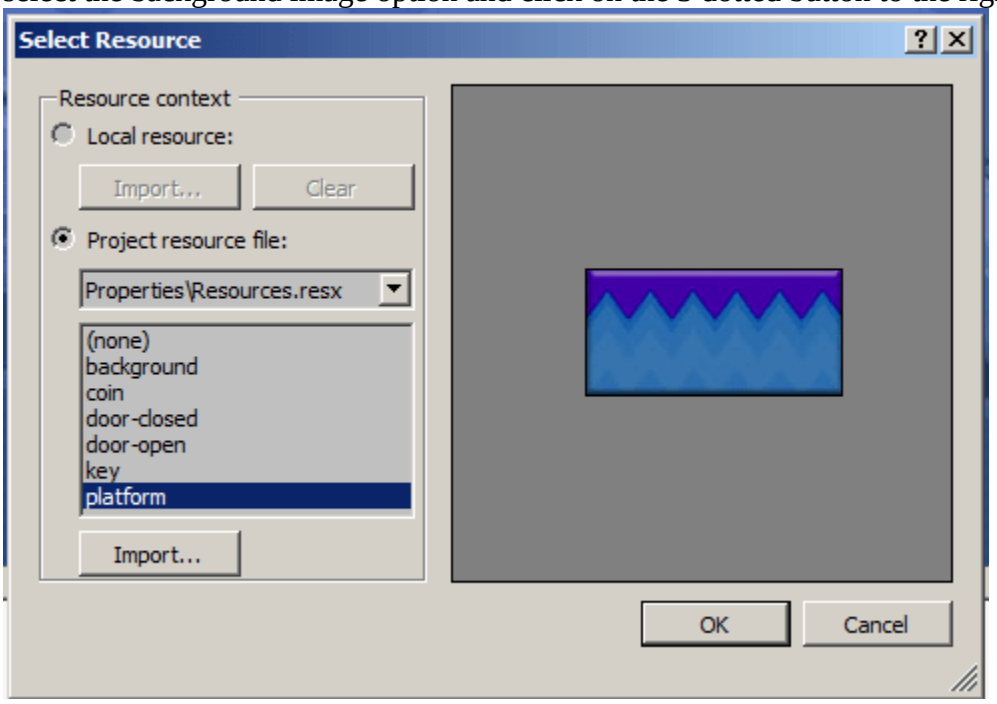
23. You will be taken back to the resource selection window and from the list of images choose the background image and click OK.
24. 

This is what the image will look like now.
25. 

26. While the image is selected go to the properties window, change the size mode to AutoSize this will change the size to 2000, 480 which is the native size of the image.

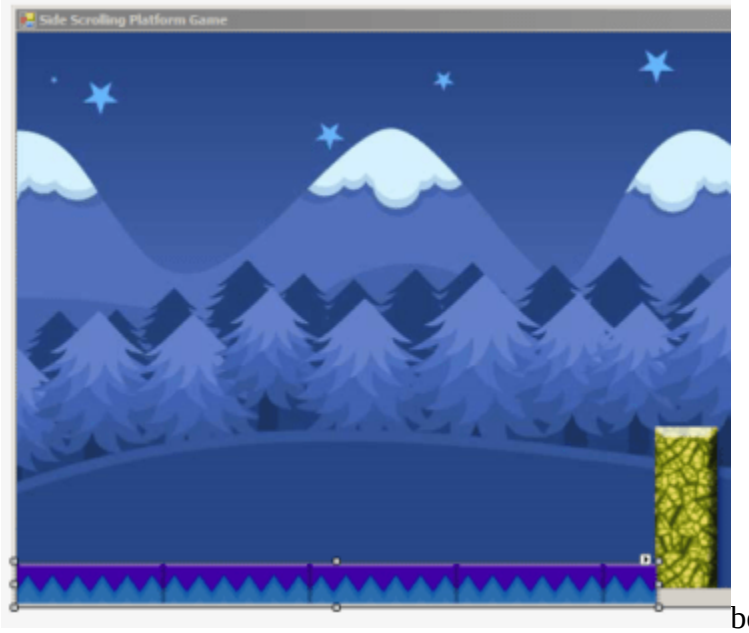
27. 
28. This is what the form looks like now, as you can see most of the image is not visible, This is the right size for the form but we need to some elements to the whole game not just the first part of it. When you click on the form you can extend the width as you require, extend it so you can see the whole picture. If you can't select the form because the picture is covering it, then just click on the title bar and it will select the form.
29. 
30. As you can see here we have extended the form to the full size now we can add platforms, coins, door and key on the level.

31. Add another picture box to the form.
32.  Select the picture box and change the following in its properties.
33. 

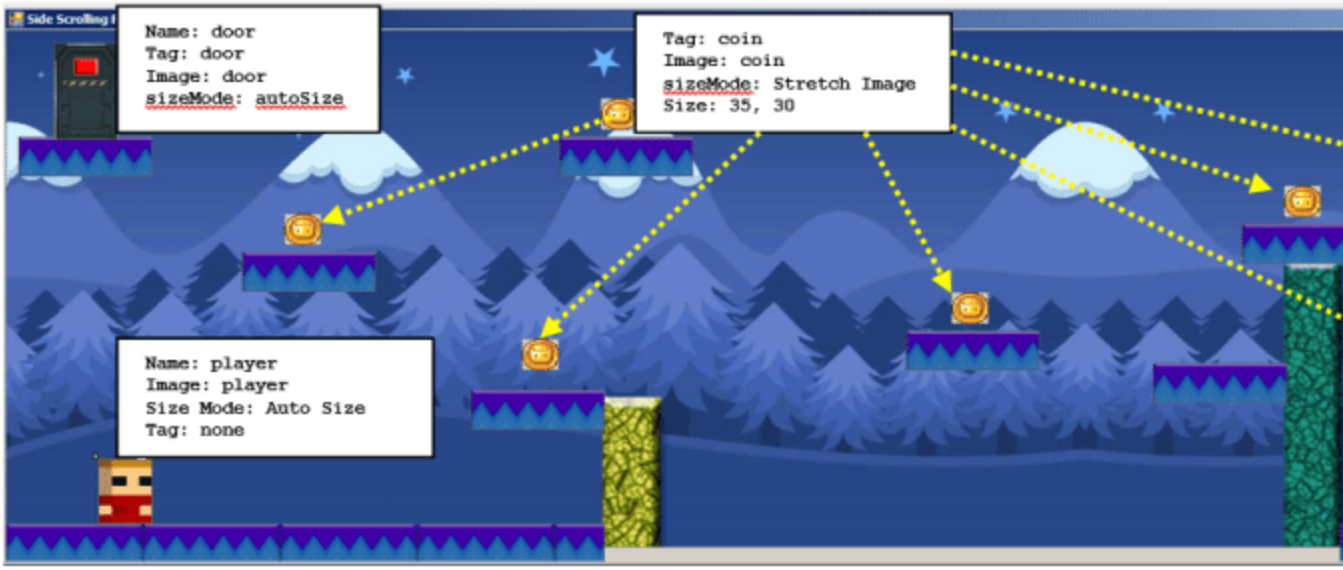
In the properties window select the background image option and click on the 3 dotted button to the right.
34. 

35. This will take you the resource selection window, from this window select the platform image and click OK.
36. 

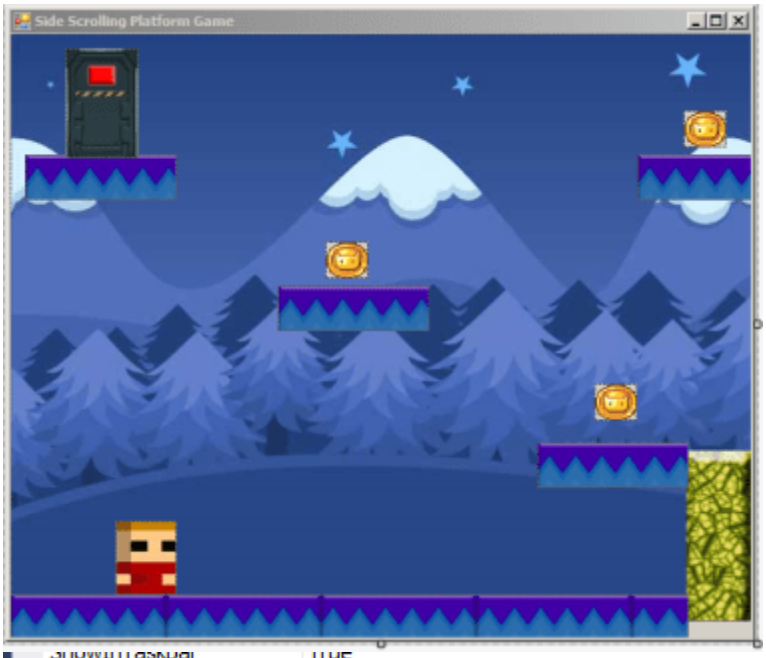
37. From the properties window add a tag to the picture box called “platform” all lower case. As we will have more than 1 platform in the game its best to give them a tag which will help organize it and identify them in the code.

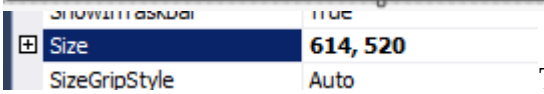
38. 

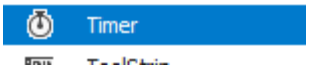
because we set the image as the background for this platform we can extend it and make it smaller the background will tile with it. Now you can feel free to add as many platforms as you want, the easiest way to do this is to simply copy and paste this platform multiple times, it will copy the background image and the tag with it.
39. 
40. Add another picture box to the form
41. **Name: door**
42. **Tag: door**
43. **Image: door**
44. **Size Mode: Auto Size**
45. **Location: you can place this door picture box anywhere in the level, we placed it on the top left over a platform see the level details below.**
46. 
47. Add another picture box to the form
48. **Name: player**
49. **Tag: none**
50. **Image: player**
51. **Size Mode: Auto Size**
52. **Location: 86, 398**

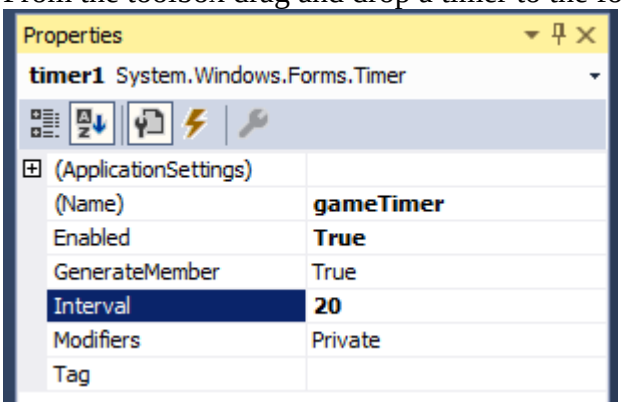
53. 
54. Add another picture box to the form
55. **Tag: coin**
56. **Image: coin**
57. **Size Mode: Stretch Image**
58. **Size: 35, 30**
59. **Location: place this object on top of the platforms, once you got one set up, you can copy and paste it many times.**
60. 
61. Add another picture box to the form
62. **Name: key**
63. **Tag: key**
64. **Image: key**
65. **Size Mode: Auto Size**
66. **Location: Place on the top right platform. See the level layout below.**
67. 

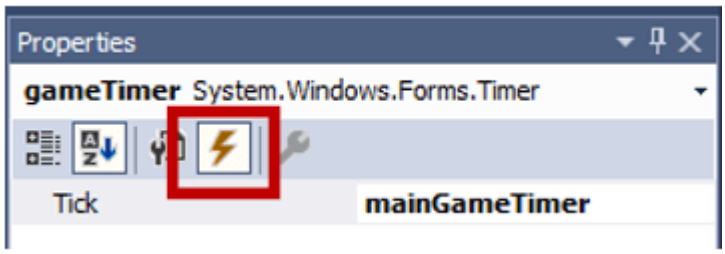
68. Above is the final level design. As you can see we have placed the door, coins, platforms and key to this level. For now you can follow the same level design and may be once you got the grip you can design your own with more details.
69. Click on the form title again, we need to set the forms size back to normal. Set the form size back to **614** in the size option in the properties window.

70. 



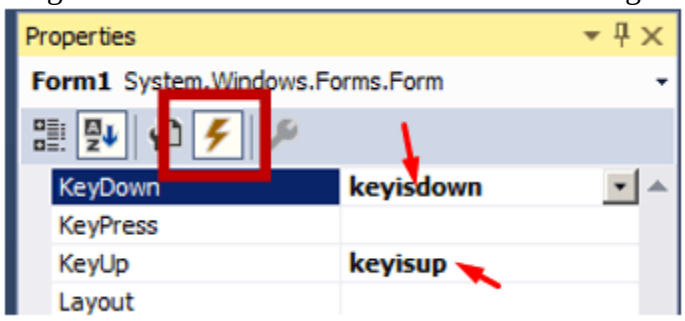
This is the form size you should apply now.
71. 

72. From the toolbox drag and drop a timer to the form.
73. 

74. In the timers properties window change the following. Set name to **gameTimer**, set enabled to **True** and Interval to **20**. Double check the image and see if it matches with yours.
75. 

76. Click on the little lightning bolt icon In the properties window, this will take you to the events manager window, for the timer there is only one event TICK. Type in

mainGameTimer and press enter. This will take you the code view, comeback to the design view we need to add 2 more events to the game.

77. 

78. Click on the form, make sure you have selected the form and see the events manager by clicking on the small lightning bolt icon. Find the key down event and type in **keyisdown** and press enter. Find the key up event type in **keyisup** and press enter.

79. Start Coding

80. This is the code view of the game so far. There are three empty events in this game we are going to start by adding variables first.

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9  using System.Windows.Forms;
10
11 namespace side_scrolling_platform_game
12 {
13     public partial class Form1 : Form
14     {
15         public Form1()
16         {
17             InitializeComponent();
```

```
18        }
19
20        private void mainGameTimer(object sender, EventArgs e)
21        {
22
23        }
24
25        private void keyisdown(object sender, KeyEventArgs e)
26        {
27
28        }
29
30        private void keyisup(object sender, KeyEventArgs e)
31        {
32
33        }
34    }
35 }
```

81. Add the highlighted code where you see in the code below. All the codes are commented to further your understanding of the process.

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.Linq;
7  using System.Text;
```

```
8   using System.Threading.Tasks;
9   using System.Windows.Forms;
10
11 namespace side_scrolling_platform_game
12 {
13     public partial class Form1 : Form
14     {
15
16         bool goleft = false; // boolean which will control players going left
17         bool goright = false; // boolean which will control players going right
18         bool jumping = false; // boolean to check if player is jumping or not
19         bool hasKey = false; // default value of whether the player has the key
20
21         int jumpSpeed = 10; // integer to set jump speed
22         int force = 8; // force of the jump in an integer
23         int score = 0; // default score integer set to 0
24
25         int playSpeed = 18; //this integer will set players speed to 18
26         int backLeft = 8; // this integer will set the background moving speed to 8
27
28         public Form1()
29         {
30             InitializeComponent();
31         }
32
33         private void mainGameTimer(object sender, EventArgs e)
34         {
35
```

```
36      }
37
38      private void keyisdown(object sender, KeyEventArgs e)
39      {
40
41      }
42
43      private void keyisup(object sender, KeyEventArgs e)
44      {
45
46      }
47  }
48 }
```

82. There are 4 Booleans first **goleft** and **goright** Booleans will be used to detect the players movement, **jumping** Boolean will be used to control how the player jumps in the game and lastly the **haskey** Booleans will be set true once the player collects the key.
83. After that we have 5 integers. **JumpSpeed** will control how fast the player jumps, **force** will be used to check how high the player can jump, **score** is obvious off course to keep score, **playSpeed** will control how fast the player moves left or right and finally **backLeft** will control the environment speed relative to the player.

84. Key is Down Event

85. This event will trigger when the player presses a key on the keyboard. We will map out three different keys for this game, left right and space key.

```
1       private void keyisdown(object sender, KeyEventArgs e)
2       {
3           //if the player pressed the left key AND the player is inside the panel
4           // then we set the car left boolean to true
5           if (e.KeyCode == Keys.Left)
6           {
7               goleft = true;
```

```
8           }
9           // if player pressed the right key and the player left plus player width is less then the panel1
    width
10
11
12          if (e.KeyCode == Keys.Right)
13          {
14             // then we set the player right to true
15             goright = true;
16          }
17
18          //if the player pressed the space key and jumping boolean is false
19
20          if (e.KeyCode == Keys.Space && !jumping)
21          {
22             // then we set jumping to true
23             jumping = true;
24          }
        }
```

```
//if the player pressed the left key AND the player is inside the panel
// then we set the car left boolean to true
if (e.KeyCode == Keys.Left)
{
goleft = true;
}
// if player pressed the right key and the player left plus player width is less then the
    panel1 width

if (e.KeyCode == Keys.Right)
{
// then we set the player right to true
goright = true;
}
```

The two if statements above will be waiting for the left or right key to be press and when they are we will set either the goleft or goright Boolean to true.

```
100.        //if the player pressed the space key and jumping boolean is false
101.
102.        if (e.KeyCode == Keys.Space && !jumping)
103.        {
104.        // then we set jumping to true
105.        jumping = true;
106.        }
```

107. Lastly in this event we are looking at the space key, in this if statement we have two different condition and they both have to be true in order for this event to trigger. We are looking for the player to press the space key AND the jumping Boolean needs to be false. In short hand code we can tell the code !jumping means jumping is not true. If both of these conditions are met then we set jumping back to true, this is a popular method to stop players from double, triple jumping in the game.

108. Key is up event

109. This is event similar to the key is down event, in simple terms we setting everything back to false once the represented keys are released. also notice we are not specifically calling the space key because we know the jumping will have to be true so if the keys are released then we set it false.

```
1   private void keyisup(object sender, KeyEventArgs e)
2   {
3       // if the LEFT key is up we set the car left to false
4       if (e.KeyCode == Keys.Left)
5       {
6           goleft = false;
7       }
8       // if the RIGHT key is up we set the car right to false
9       if (e.KeyCode == Keys.Right)
10      {
11          goright = false;
12      }
13      //when the keys are released we check if jumping is true
14      // if so we need to set it back to false so the player can jump again
15      if (jumping)
16      {
```

```
17              jumping = false;
18          }
19      }
```

110. Main Game Timer event

111. This event controls the whole game, from the movements of the player, to the environment and also removing objects from the form as they collide with each other. Follow the code in this event closely because its long and can get complicated, do one line at a time and if you get any errors check back with this tutorial.

```
1       private void mainGameTimer(object sender, EventArgs e)
2       {
3           // linking the jumpspeed integer with the player picture boxes to location
4           player.Top += jumpSpeed;
5
6           // refresh the player picture box consistently
7           player.Refresh();
8
9           // if jumping is true and force is less than 0
10          // then change jumping to false
11          if (jumping && force < 0)
12          {
13              jumping = false;
14          }
15
16          // if jumping is true
17          // then change jump speed to -12
18          // reduce force by 1
19          if (jumping)
20          {
21              jumpSpeed = -12;
```

```
22              force -= 1;
23          }
24          else
25          {
26              // else change the jump speed to 12
27              jumpSpeed = 12;
28          }
29
30          // if go left is true and players left is greater than 100 pixels
31          // only then move player towards left of the
32          if (goleft && player.Left > 100)
33          {
34              player.Left -= playSpeed;
35          }
36          // by doing the if statement above, the player picture will stop on the forms left
37
38
39          // if go right Boolean is true
40          // player left plus players width plus 100 is less than the forms width
41          // then we move the player towards the right by adding to the players left
42          if (goright && player.Left + (player.Width + 100) < this.ClientSize.Width)
43          {
44              player.Left += playSpeed;
45
46          }
47          // by doing the if statement above, the player picture will stop on the forms right
48
49
```

```
50          // if go right is true and the background picture left is greater 1352
51          // then we move the background picture towards the left
52          if (goright && background.Left > -1353)
53          {
54              background.Left -= backLeft;
55
56              // the for loop below is checking to see the platforms and coins in the level
57              // when they are found it will move them towards the left
58              foreach (Control x in this.Controls)
59              {
60                  if (x is PictureBox && x.Tag == "platform" || x is PictureBox && x.Tag == "coin" || x is
    PictureBox && x.Tag == "door" || x is PictureBox && x.Tag == "key")
61
62                  {
63                      x.Left -= backLeft;
64                  }
65              }
66
67          }
68
69          // if go left is true and the background pictures left is less than 2
70          // then we move the background picture towards the right
71          if (goleft && background.Left < 2)
72          {
73              background.Left += backLeft;
74
75              // below the is the for loop thats checking to see the platforms and coins in the level
76              // when they are found in the level it will move them all towards the right with the
    background
77              foreach (Control x in this.Controls)
```

```
78              {
79                  if (x is PictureBox && x.Tag == "platform" || x is PictureBox && x.Tag == "coin" || x is
    PictureBox && x.Tag == "door" || x is PictureBox && x.Tag == "key")
80
81                  {
82                      x.Left += backLeft;
83                  }
84              }
85          }
86
87
88          // below if the for loop thats checking for all of the controls in this form
89          foreach (Control x in this.Controls)
90          {
91              // is X is a picture box and it has a tag of platform
92              if (x is PictureBox && x.Tag == "platform")
93              {
94                  // then we are checking if the player is colliding with the platform
95                  // and jumping is set to false
96                  if (player.Bounds.IntersectsWith(x.Bounds) && !jumping)
97                  {
98                      // then we do the following
99                      force = 8; // set the force to 8
100                     player.Top = x.Top - player.Height; // also we place the player on top of the picture
    box
101
102                     jumpSpeed = 0; // set the jump speed to 0
103                 }
104             }
105             // if the picture box found has a tag of coin
                if (x is PictureBox && x.Tag == "coin")
```

```
106             {
107                 // now if the player collides with the coin picture box
108                 if (player.Bounds.IntersectsWith(x.Bounds))
109                 {
110                     this.Controls.Remove(x); // then we are going to remove the coin image
111                     score++; // add 1 to the score
112                 }
113             }
114         }
115
116         // if the player collides with the door and has key boolean is true
117
118         if (player.Bounds.IntersectsWith(door.Bounds) && hasKey)
119         {
120             // then we change the image of the door to open
121             door.Image = Properties.Resources.door_open;
122             // and we stop the timer
123             gameTimer.Stop();
124             MessageBox.Show("You Completed the level!!"); // show the message box
125         }
126
127         // if the player collides with the key picture box
128
129         if (player.Bounds.IntersectsWith(key.Bounds))
130         {
131
132             // then we remove the key from the game
133             this.Controls.Remove(key);
```

```
134             // change the has key boolean to true
135             hasKey = true;
136         }
137
138
139         // this is where the player dies
140         // if the player goes below the forms height then we will end the game
141         if (player.Top + player.Height > this.ClientSize.Height + 60)
142         {
143             gameTimer.Stop(); // stop the timer
144             MessageBox.Show("You Died!!!"); // show the message box
            }
        }
```

112.
113. // **linking the jumpspeed integer with the player picture boxes to location**
114. **player.Top += jumpSpeed;**
115. The code above is linking the jump speed with the player picture boxes top location. This will artificially add gravity to the player by including += meaning in every frame the player character will be pushed down according to the value of jump speed.
116. // **refresh the player picture box consistently**
117. **player.Refresh();**
118. Every picture box comes with several functions built in them, one of them is the refresh function. The picture box will flicker when you are playing the game so using this refresh function allows that to be reduced down a little bit.
119. // **if jumping is true and force is less than 0**
120. // **then change jumping to false**
121. **if (jumping && force < 0)**
122. **{**
123. **jumping = false;**
124. **}**
125. The if statement above means that if the jumping Boolean is true and force is less than 0 then we set jumping to false.
126. // **if jumping is true**
127. // **then change jump speed to -12**
128. // **reduce force by 1**
129. **if (jumping)**
130. **{**

```
jumpSpeed = -12;
force -= 1;
}
else
{
// else change the jump speed to 12
jumpSpeed = 12;
}
```
In the if statement above if jumping is true then we reverse the jump speed which will propel the player upwards and we reduce 1 from the force as the character jumps, else statement will trigger when the if statement condition becomes false. If the character is not jumping then we add force to the character in the jump speed.
```
// if go left is true and players left is greater than 100 pixels
// only then move player towards left of the
if (goleft && player.Left > 100)
{
player.Left -= playSpeed;
}
// by doing the if statement above, the player picture will stop on the forms left
```
In the if statement above if the go left Boolean is true and the player is further than 100 pixels from the left then we will allow the player to move the left. By doing an if statement like this we can stop the player from leaving the form from the left.
```
// if go right Boolean is true
// player left plus players width plus 100 is less than the forms width
// then we move the player towards the right by adding to the players left
if (goright && player.Left + (player.Width + 100) < this.ClientSize.Width)
{
player.Left += playSpeed;
}
```
In this if statement above we are looking if the go right Boolean is true AND players left position + player width + 100 pixels is less than the forms width meaning the right side of the form then we allow the player to move towards the right. if this condition is not met then the player will not move.
```
// if go right is true and the background picture left is greater 1352
// then we move the background picture towards the left
if (goright && background.Left > -1353)
{
background.Left -= backLeft;
// the for loop below is checking to see the platforms and coins in the level
// when they are found it will move them towards the left
foreach (Control x in this.Controls)
{
if (x is PictureBox && x.Tag == "platform" || x is PictureBox && x.Tag == "coin" || x is PictureBox && x.Tag == "door" || x is PictureBox && x.Tag == "key")
{
```

x.Left -= backLeft;
}
}
}

In the if statement above you will see that we have a for each loop inside it. Lets break this down and explain it further.

IF GORIGHT AND BACKGROUND IMAGES LEFT POSITION IS GREATER THAN -1353 (this number is not random I tested it couple of times and this is the number that fits the best with the game.)

If those conditions are true then we move the background towards the left with background.Left -= backLeft; So if the player is moving right the background should move left.

Now the next part should make you aware of why we used tags, in visual studio two objects cannot share the same name however they can share the same tag. So we use tags to identify multiple objects. Because there are so many different platforms, coins, door and key on the level we need to move them all with the background at one speed giving the illusion of movement. We are using this symbol || in the if statement to differentiate the conditions, the is the way the program reads this statement

IF X IS A PICTURE BOX AND IT HAS THE TAG PLATFORM **OR** X IS A PICTURE BOX AND IT HAS THE TAG COIN **OR** IF X IS A PICTURE BOX AND IT HAS THE TAG DOOR **OR** IF X IS A PICTURE BOX AND IT HAS THE TAG KEY.

```
// the for loop below is checking to see the platforms and coins in the level
// when they are found it will move them towards the left
foreach (Control x in this.Controls)
{
if (x is PictureBox && x.Tag == “platform” || x is PictureBox && x.Tag == “coin” || x is PictureBox && x.Tag == “door” || x is PictureBox && x.Tag == “key”)
{
x.Left -= backLeft;
}
}
```

This for look inside that if statement simply states that we loop through every control component in this.Controls meaning this form. Then we identify if those controls are a picture box AND they have the given tags then we move them towards the left with the backLeft speed. Each of those controls will be linked in that x variable in the loop and they will be process to move towards the left.

```
// if go left is true and the background pictures left is less than 2
// then we move the background picture towards the right
if (goleft && background.Left < 2)
{
background.Left += backLeft;
// below the is the for loop thats checking to see the platforms and coins in the level
// when they are found in the level it will move them all towards the right with the background
foreach (Control x in this.Controls)
```

```
{
if (x is PictureBox && x.Tag == “platform” || x is PictureBox && x.Tag == “coin” || x is PictureBox && x.Tag == “door” || x is PictureBox && x.Tag == “key”)
{
x.Left += backLeft;
}
}
}
```

The above if statement now controlling the background from moving to the right.

IF GOLEFT IS TRUE AND BACKGROUND LEFT POSITION IS LESS THAN 2

THEN WE MOVE THE BACKGROUND TO THE LEFT USING += BACKLEFT VALUE.

We’ve done the similar thing to move the background, platforms, coins, door and key towards the right of the screen.

```
// below if the for loop thats checking for all of the controls in this form
foreach (Control x in this.Controls)
{
// is X is a picture box and it has a tag of platform
if (x is PictureBox && x.Tag == “platform”)
{
// then we are checking if the player is colliding with the platform
// and jumping is set to false
if (player.Bounds.IntersectsWith(x.Bounds) && !jumping)
{
// then we do the following
force = 8; // set the force to 8
player.Top = x.Top – player.Height; // also we place the player on top of the picture box
jumpSpeed = 0; // set the jump speed to 0
}
}
// if the picture box found has a tag of coin
if (x is PictureBox && x.Tag == “coin”)
{
// now if the player collides with the coin picture box
if (player.Bounds.IntersectsWith(x.Bounds))
{
this.Controls.Remove(x); // then we are going to remove the coin image
score++; // add 1 to the score
}
}
}
```

After that we are running another loop in the timer event, this one will check when we jump on top of the platform and collide with the coins.

233. The first if statement in the loop is checking if X control is a picture box and it has a tag of platform then we are also checking if the player intersects with it and player is not jumping if this is true then, we set the force back to 8, we place the player on top of the given platform by using player.top = x.top + player.Height. then we are setting the jumpSpeed to 0.

234. After the platform calculations we move on to the coin, now we don't want the player to jump on top of the coin, we want the coin to disappear from the scene and we want to add 1 to the score integer. So we are once again checking the if X is a picture box and it has a tag of coin, then we are also checking if the player intersects with the coin then we remove the coin from the game and add 1 to the score integer, it's as simple as this. Now we don't usually do this but I have some homework for you regarding the score. Remember this section because it will be relating to this later in the tutorial.

```
235.    // if the player collides with the door and has key boolean is true
236.    if (player.Bounds.IntersectsWith(door.Bounds) && hasKey)
237.    {
238.    // then we change the image of the door to open
239.    door.Image = Properties.Resources.door_open;
240.    // and we stop the timer
241.    gameTimer.Stop();
242.    MessageBox.Show("You Completed the level!!"); // show the message box
243.    }
```

244. In the is statement above we are looking at the collision between the player and the door. In this if statement the player will have to intersects with the door AND the hasKey Boolean must be true, if so then we change the door image to the door open image, stop the game timer and show the level complete message.

```
245.    // if the player collides with the key picture box
246.    if (player.Bounds.IntersectsWith(key.Bounds))
247.    {
248.    // then we remove the key from the game
249.    this.Controls.Remove(key);
250.    // change the has key boolean to true
251.    hasKey = true;
252.    }
```

253. In the if statement above we are checking if the player intersects with the key then we will remove the key and change the hasKey Boolean to true. Remember without the hasKey Boolean being true the door will not open thus the game is not going to end.

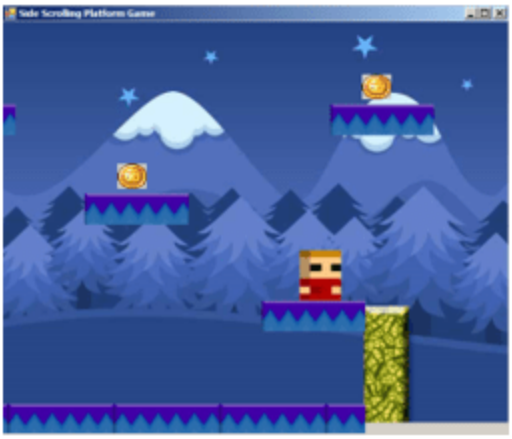
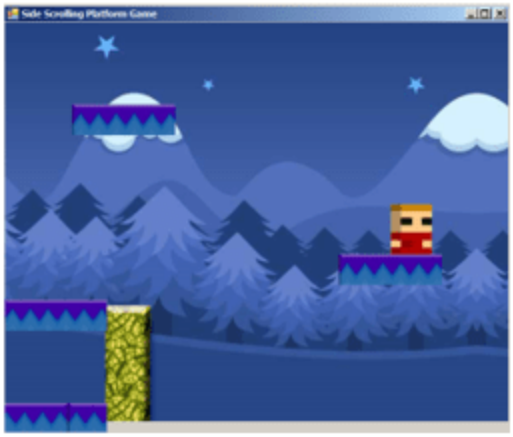
```
254.    // this is where the player dies
255.    // if the player goes below the forms height then we will end the game
256.    if (player.Top + player.Height > this.ClientSize.Height + 60)
257.    {
258.    gameTimer.Stop(); // stop the timer
259.    MessageBox.Show("You Died!!!"); // show the message box
260.    }
```

261. Above is the last if statement in this program, this one is checking if the player has dropped off the form from the bottom then we stop the timer and show a message that states you died, in the nicest way possible off course. The way to read this if statement is

IF THE PLAYERS TOP LOCATION + PLAYERS HEIGHT IS GREATER THAN THE FORMS HEIGHT + 60 PIXELS then we follow the instructions inside the if statement.

262. Let's try to run the game – 

Click on the start button on the top tool bar.

263.   

264. Homework –

265. Remember when I mentioned that score integer and we will come back to it, at the moment you have the score being added as you collect the coins however nothing on the screen or the end screen message is showing the score. Can you fix That? Add something to the code or to the form that allows you to see how many coins you collected.