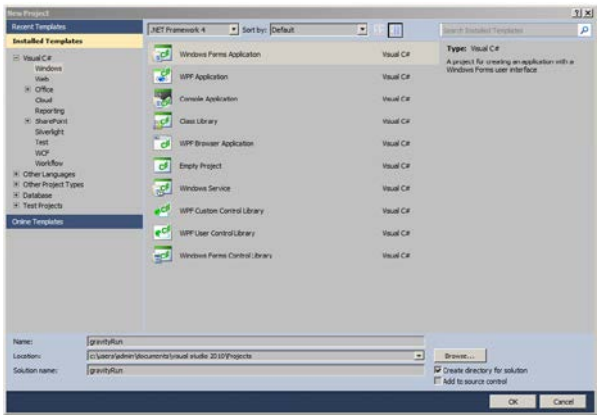
C# Tutorial - Create a Batman Gravity Run Game

Start a new project in visual studio and call it gravityRun, to follow along this tutorial please download the image assets for the game from http://www.mooict.com

It should be a windows form application with C#

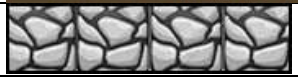


Click OK

Change the size of the to 800,300 and TEXT to Batman Gravity Run

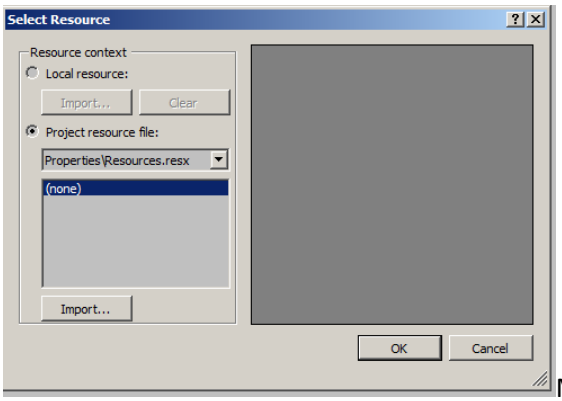
We are going to use some graphics and found a nice little batman running GIF, off course you can use any image you want for the project.

Lets take a look at the assets of the game.

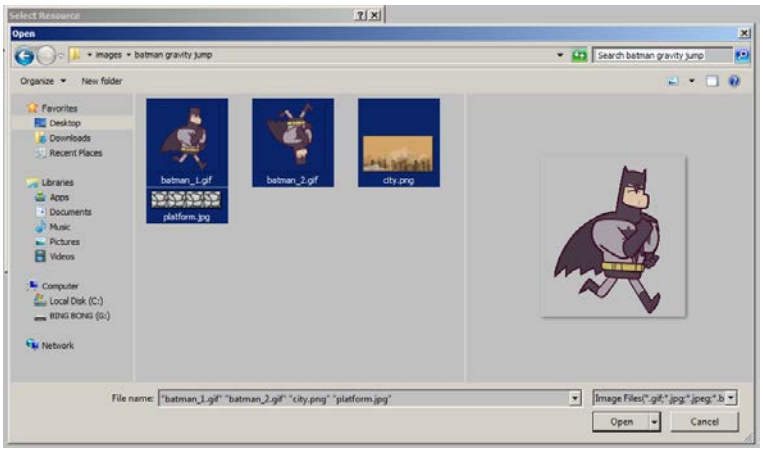
	This is the batman running GIF we have its an animated gif this will be used when batman is running under normal gravity.
	This upside down batman GIF will be used when batman is running under reverse gravity
	This is the city back ground for the form
	This is the platform image for the platform which the player can run on.

Now lets look back at the properties of the form and change the back ground of it.

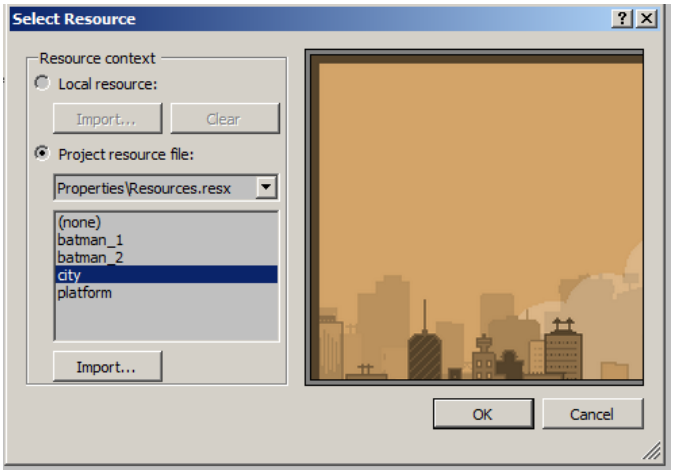
Click on the three dots (...)



Make sure the project resource files are selected and click on import.

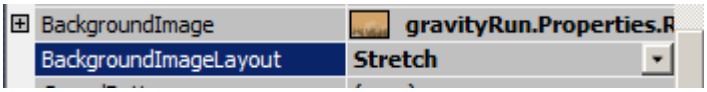


Navigate to the folder where you saved all of the images in. Select all and click open.



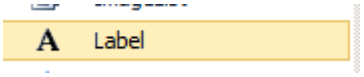
Select city from the list and click OK.

We don't want the background to repeat on the form. We want to stretch it to the forms size.



Change the background image layout option in the properties menu to Stretch.

Add a Label to the form.



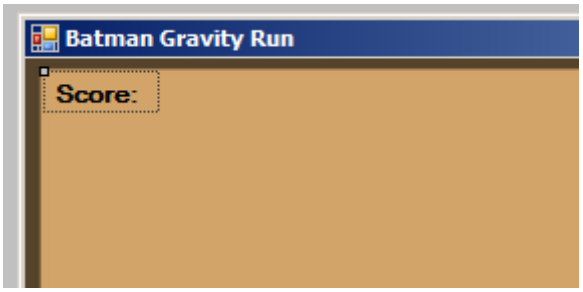
Change the properties of the label to

BackColor - Transparent

Font - 9 point BOLD

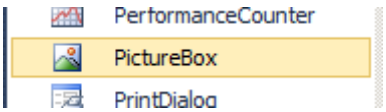
Text - Score:

Location - 12,9



It will be placed in the top left corner of the screen.

Add a picture box to the form.

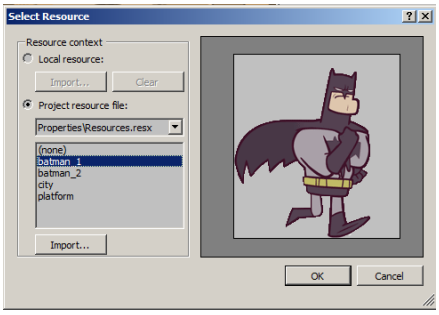


Change the following in the properties window

Name - player

BackColor - Transparent

Click on the three dots (...) to open the image dialog box.

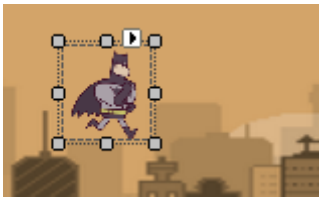


Click on Batman 1 and click OK

Location - 119, 128

Size - 45, 48

SizeMode - Stretch Image



Add another picture box

This will be used for the platforms

name - **p1**

BackColor - Transparent

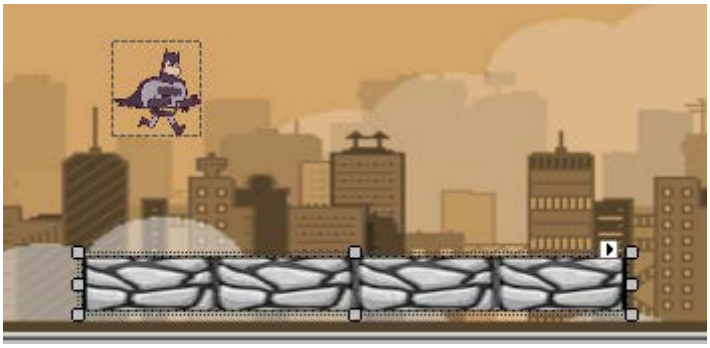
Image - Choose the **platform picture** from the resources window as we did for batman earlier

Location - 104, 235

Size - 273, 28

SizeMode - Stretch Image

Tag - Platform



Now copy and paste this picture box

Apply the following the properties of this one.

Name - p2

BackColor - Transparent

Image - Platform Image you imported earlier

Location - 491, 235

Size - 273, 28 (Same)

SizeMode - Stretch Image

Tag - Platform

Lets copy and paste it again for the 3rd platform and apply the following properties to it.

Name - p3

BackColor - Transparent

Image - Platform Image you imported earlier

Location - 304,26

Size - 273, 28 (Same)

SizeMode - Stretch Image

Tag - Platform

Now for the final platform, once again and copy and paste the final platform to the form.

Name - p4

BackColor - Transparent

Image - Platform Image you imported earlier

Location - 705,26

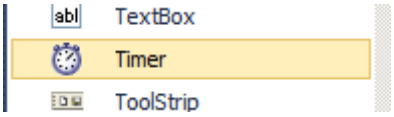
Size - 273, 28 (Same)

SizeMode - Stretch Image

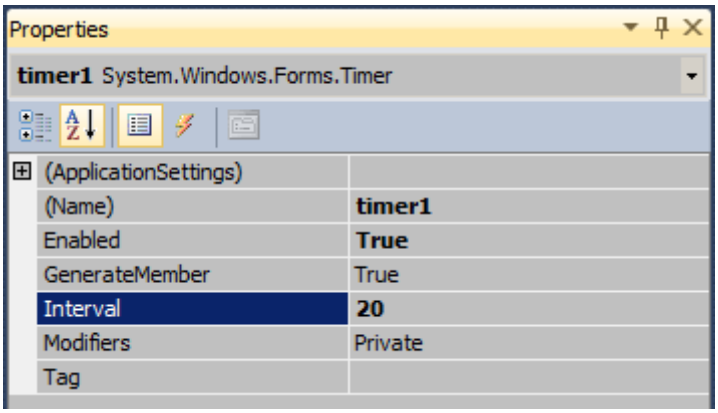
Tag - Platform

Now all of the platforms are done.

We need one final ingredient for this game to work, it's the timer.

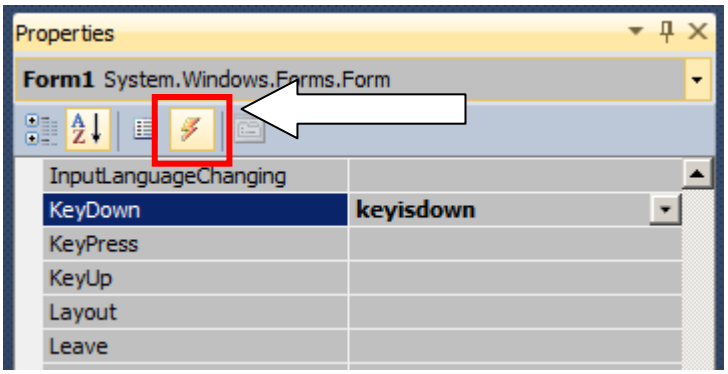


Add the following to the timer.



Enabled - True Interval - 20

Click on the form and check the properties window, click on the lightning button.



Find the key down option and type **keyisdown** and press enter. When you do so it will take you to the code view, come back to the design view we need one more event.

Double click on the timer icon bottom of the form.

<-- double click this.

```
bool landed = false;
int gravity = 5;
int score = 0;
int platformSpeed = 10;
Random rnd = new Random();
```

```
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 gravityRun
{
    public partial class Form1 : Form
    {
        bool landed = false;
        int gravity = 5;
        int score = 0;
        int platformSpeed = 10;
        Random rnd = new Random();

        public Form1()
        {
            InitializeComponent();
        }

        private void keyisdown(object sender, KeyEventArgs e)
        {

        }

        private void timer1_Tick(object sender, EventArgs e)
        {

        }
    }
}
```

All the following are global variables. Enter them before the form1() line.

bool landed is a Boolean which we will be turned true when the player has landed on a platform. We will set it back to false when we have jumped off the platform again.

int gravity is set to 5 this will be used for to create a simple gravity for our player.

int score will keep track of how far our player has come in the game.

int platformSpeed this is the speed for our platforms.

Random rnd = new Random() this will be used to get a random number, this will increase the challenge for our game.

Add the following code in the **keyisdown** function.

```
private void keyisdown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Space && landed == true)
    {
        gravity = -gravity;
        landed = false;
        // do work
    }

    if (e.KeyCode == Keys.R)
    {
        reset();
    }
}
```

In the key down event we will check which keys are pressed and take action.
If the player presses space AND landed is true then we change the gravity from positive to negative and change the landing to false.

We are also checking if the player presses R we will run the reset function.

This is the reset function

```
private void reset()
{
    score = 0;
    p1.Width = 273;
    p2.Width = 273;
    p3.Width = 273;
    p4.Width = 273;
```

This is the reset function. This is an independent function to rest of the events. This function will reset each of the components on the screen for example it will reset the width of the platforms to 273 which is default.

```
            player.Left = 147;
            player.Top = 131;

            p1.Left = 104;
            p1.Top = 235;

            p2.Left = 491;
            p2.Top = 235;

            p3.Left = 304;
            p3.Top = 26;

            p4.Left = 702;
            p4.Top = 26;

            timer1.Start();
        }
```

It will reset the player back to its original place LEFT 147 and TOP 131.

IT will reset P1, P2, P3 and P4 picture boxes to their original place.

Finally it will start the timer again.

When the player presses R on the keyboard it will restart the game and it will be default value of it.

This is the timer event, add the following code into the function.

```
private void timer1_Tick(object sender, EventArgs e)
        {
            player.Top += gravity;
            player.Left = 80;
            label1.Text = "Score: " + score;

            foreach (Control x in this.Controls)
            {
                if ( x is PictureBox && x.Tag == "platform")
                {
                x.Left -= platformSpeed;

                if (x.Left < -500)
                {
                    x.Width = rnd.Next(100, 300);
                    x.Left = 500;
                    score++;
                }
            }

            if (player.Bounds.IntersectsWith(p2.Bounds) ||
                player.Bounds.IntersectsWith(p1.Bounds))
            {
                landed = true;
                player.Top = p2.Top - player.Height;
                player.Image = Properties.Resources.batman_1;
            }

            if (player.Bounds.IntersectsWith(p3.Bounds) ||
                player.Bounds.IntersectsWith(p4.Bounds))
            {
                landed = true;
                player.Top = p3.Top + p3.Height;
                player.Image = Properties.Resources.batman_2;
            }

 if(player.Top < -40 || player.Top > ClientSize.Height)
            {
                timer1.Stop();
                label1.Text += " -- Press R to reset";
            }

}
```

As you can see all of important things are happening in the timer event. Lets go line by line

player.Top += gravity; when the game starts the player object will move downwards. += gravity will add 5 pixels to each time the timer runs.
player.Left = 80; This line will keep the player 80 pixels to the left.
label1.Text = "Score: " + score; This line is showing the score on the label 1.

foreach (Control x in this.Controls) This line is starting the loop. We are creating a x variable with the type of control in this form.
For loops have open and closed curly brackets { } don't miss them.

if (x is PictureBox && x.Tag == "platform") we created a if statement inside the loop. This will check is X is a type of picture box and it has a tag of platform. Which if you followed the tutorial thus far, it's the tag we inserted in the platform properties. don't forget the open and curly brackets of the if statement { }

x.Left -= platformSpeed; This will move the platforms towards the left of the screen.

if (x.Left < -500) If the platforms reach -500
Then we change the width of X to a random number between 100 and 300 pixels. This will make the game more challenging.
Then we change the X.Left to 500 and increase the score by 1.

After this we start checking if the player is touching the platforms p1 p2 p3 or p4

if (player.Bounds.IntersectsWith(p2.Bounds) ||
player.Bounds.IntersectsWith(p1.Bounds))
This is an example of an if statement which is checking multiple conditions the way it works if

If(condition 1 OR condition 2) In this case we are checking if player is hitting p1 or p2.

landed = true; we change the landed to true
player.Top = p2.Top - player.Height; Now we keep the player on TOP of the p1 or p2. Since they both are bottom we want the player to on top of them.

player.Image = Properties.Resources.batman_1;
This is the batman 1 image with the normal batman running.

if (player.Bounds.IntersectsWith(p3.Bounds) ||
player.Bounds.IntersectsWith(p4.Bounds))
In this if statement we are checking if the player is touching either p3 or p4 platform.

If player is then we change the landing to true.
player.Top = p3.Top + p3.Height; Since this is the top platform we don't want to be on top of it we want to be on bottom of it.
player.Image = Properties.Resources.batman_2; Now we change it to the upside down batman image.

if(player.Top < -40 || player.Top > ClientSize.Height)
Now we are checking if the player has gone passed the screen either through the bottom or top. Player.TOP < -40 is player has left the screen from the top as in -40 pixels from the top. and if player.TOP > clietSize.Height meaning from bottom of the screen then we can run the following instruction.

timer1.Stop();
This will stop the timer

label1.Text += " -- Press R to reset";
Then we will add the last part of the game, where the game score is showing once the game ends, we will add the following to the label -- Press R to reset.

Common mistakes we make in programming are to forget to open or close curly brackets after loops or if statement, pay extra attention to these and make sure you have them all matching.

Now run the game

Batman is running on the bottom platforms

Batman is running upside down on the top platforms.

Full code of the game

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

namespace gravityRun
{
    public partial class Form1 : Form
    {

        bool landed = false;
        int gravity = 5;
        int score = 0;
        int platformSpeed = 10;
        Random rnd = new Random();

        public Form1()
        {
            InitializeComponent();
        }

        private void keyisdown(object sender, KeyEventArgs e)
        {
            if (e.KeyCode == Keys.Space && landed == true)
            {
                gravity = -gravity;
                landed = false;
                // do work
            }

            if (e.KeyCode == Keys.R)
            {
                reset();
            }
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            player.Top += gravity;
            player.Left = 80;
            label1.Text = "Score: " + score;

            foreach (Control x in this.Controls)
            {
```

```
                if ( x is PictureBox && x.Tag == "platform")

                x.Left -= platformSpeed;

                if (x.Left < -500)
                {
                    x.Left = 500;
                    x.Width = rnd.Next(100, 300);
                    score++;
                }
            }

            if (player.Bounds.IntersectsWith(p2.Bounds) || player.Bounds.IntersectsWith(p1.Bounds))
            {
                landed = true;
                player.Top = p2.Top - player.Height;
                player.Image = Properties.Resources.batman_1;
            }

            if (player.Bounds.IntersectsWith(p3.Bounds) || player.Bounds.IntersectsWith(p4.Bounds))
            {
                landed = true;
                player.Top = p3.Top + p3.Height;
                player.Image = Properties.Resources.batman_2;
            }

            if(player.Top < -40 || player.Top > ClientSize.Height)
            {
                timer1.Stop();
                label1.Text += " -- Press R to reset";
            }

        }


        private void reset()
        {
            score = 0;
            p1.Width = 273;
            p2.Width = 273;
            p3.Width = 273;
            p4.Width = 273;
            player.Left = 147;
            player.Top = 131;

            p1.Left = 104;
            p1.Top = 235;

            p2.Left = 491;
            p2.Top = 235;

            p3.Left = 304;
            p3.Top = 26;

            p4.Left = 702;
            p4.Top = 26;

            timer1.Start();
        }
    }
}
```