

How to build splash screen in windows forms application?

Asked 13 years, 1 month ago Modified 1 year, 10 months ago

Viewed 170k times



77

I need to show splash screen on my application start for few seconds. Does anybody know how to implement this?



Will be much appreciate for the help.



c#

winforms

splash-screen



Share

Improve this question

Follow

edited Dec 10, 2013 at 21:43



Jim G.

15.4k ● 22 ● 108 ● 176


asked Oct 31, 2011 at 15:16



mironych

2,968 ● 2 ● 29 ● 38

14 +1 because there is no REAL example of a good splash so far. I've got into a messy implementation from diferent blogs and websites but a good one just came up after a lot of trouble. The question is really good. – [Anderson Matos](#) Nov 1, 2011 at 3:48

There is another option that does not require you to use timers or keep a reference of your splash form in your main form or vice versa: stackoverflow.com/questions/7963275
– Veldmuys Nov 1, 2011 at 7:25 

possible duplicate of [Splash Screen Example](#) – Jim G. Dec 10, 2013 at 21:43

The sample below from Telerik uses a ShapedForm control however change that to a normal Windows form. This is by far the easiest and best way I've seen.
telerik.com/support/kb/winforms/forms-and-dialogs/details/...
– driverobject Oct 13, 2014 at 5:49

[Show Splash Screen during Loading the Main Form](#)
– Reza Aghaei Apr 7, 2018 at 11:22

13 Answers

Sorted by:

Highest score (default)



101



First, create your splash screen as a borderless, immovable form with your image on it, set to initially display at the center of the screen, colored the way you want. All of this can be set from within the designer; specifically, you want to:



- Set the form's ControlBox, MaximizeBox, MinimizeBox and ShowIcon properties to "False"
- Set the StartPosition property to "CenterScreen"
- Set the FormBorderStyle property to "None"
- Set the form's MinimumSize and MaximumSize to be the same as its initial Size.

Then, you need to decide where to show it and where to dismiss it. These two tasks need to occur on opposite sides of the main startup logic of your program. This could be in your application's `main()` routine, or possibly in your main application form's Load handler; wherever you're creating large expensive objects, reading settings from the hard drive, and generally taking a long time to do stuff behind the scenes before the main application screen displays.

Then, all you have to do is create an instance of your form, `Show()` it, and keep a reference to it while you do your startup initialization. Once your main form has loaded, `Close()` it.

If your splash screen will have an animated image on it, the window will need to be "double-buffered" as well, and you will need to be absolutely sure that all initialization logic happens outside the GUI thread (meaning you cannot have your main loading logic in the mainform's Load handler; you'll have to create a `BackgroundWorker` or some other threaded routine.

[Share](#) [Improve this answer](#)

[edited Oct 31, 2011 at 16:24](#)

[Follow](#)

answered Oct 31, 2011 at 15:29



[KeithS](#)

71.5k ● 23 ● 116 ● 165

8 The splash screen should also run on a separate thread to have a process run in the background so that your application can do work while being displayed (e.g. connect to your database, load files). It tells the user that your application is running and it does not waste their time by just staying there for X number of seconds. – [Scott Wylie](#) Nov 1, 2011 at 0:40

1 +1 for details. I've created a splash here using a new application loop (Application.Run) for it and a singleton approach so I can create the splash at the very start and close it just before the login form being shown.
– [Anderson Matos](#) Nov 1, 2011 at 3:47 ✎

@ALMMa - I did practically the same thing building my splash screen implementation, including the ability to explicitly hide or "duck" it either by command or if certain types of windows (ike Windows Security login dialogs) were detected. – [KeithS](#) Feb 12, 2015 at 23:08

1 An example of a working script is missing – [JinSnow](#) Dec 28, 2020 at 15:52

@JinSnow - "How do I do this" is not always best answered with "Like this:" followed by a code block. Case in point, the best implementation of a splash screen can vary significantly based on the loading behavior of the app it's plugged into. We also discourage "gimme t3h c0d3z" here. My answer (and many others) give you the steps to follow (including specific properties to set) to create the form, and various caveats to watch out for when plugging it in to initialization logic. Questions seeking more specific help are expected to show what they've got so far. – [KeithS](#) Jun 14, 2021 at 23:06



Here are some guideline steps...

5



1. Create a borderless form (this will be your splash screen)
2. On application start, start a timer (with a few seconds interval)
3. Show your Splash Form
4. On Timer.Tick event, stop timer and close Splash form - then show your main application form

Give this a go and if you get stuck then come back and ask more specific questions relating to your problems

Share Improve this answer

answered Oct 31, 2011 at 15:20

Follow



[musefan](#)

48.4k ● 21 ● 140 ● 191



5



simple and easy solution to create splash screen

1. open new form use name "SPLASH"
2. change background image whatever you want
3. select progress bar
4. select timer

now set timer tick in timer:

```
private void timer1_Tick(object sender, EventArgs e)
{
    progressBar1.Increment(1);
    if (progressBar1.Value == 100) timer1.Stop();
}
```

add new form use name "FORM-1" and use following command in FORM 1.

note: Splash form works before opening your form1

1. add this library

```
using System.Threading;
```

2. create function

```
public void splash()  
{  
    Application.Run(new splash());  
}
```

3. use following command in initialization like below.

```
public partial class login : Form  
{  
    public login()  
    {  
        Thread t = new Thread(new ThreadStart(spla  
t.Start();  
        Thread.Sleep(15625);  
  
        InitializeComponent();  
  
        enter code here  
  
        t.Abort();  
    }  
}
```

<http://solutions.musanitech.com/c-create-splash-screen/>

Follow



Steve

9,551 ● 10 ● 53 ● 84

answered Sep 9, 2015 at 1:31



Ha Hani Mehdi

Mehd 195 ● 4 ● 9

1 This is the simplest and best example I have found. Worked perfectly. – [Ben Winding](#) Jan 13, 2016 at 5:03

1 @TylerDurden, *simplest* and *best*? Check [this one](#) instead (and I'd carefully call it *good enough* myself). – [Sinatr](#) Sep 6, 2016 at 12:47

Thanks @Sinatr, I would probably use that implementation next time, it looks to be much more flexible. This solution was just simpler when I was picking up c#, I was originally looking for a simple solution, which didn't require delegates and too much threading. I've also learnt that the *best solution* changes as you learn new methods ;) Thanks for sharing – [Ben Winding](#) Sep 7, 2016 at 11:18

1 how to get rid of exception thrown by t.abort() ? – [Vilas Joshi](#) Sep 4, 2020 at 6:25

Sorry, what is `new splash()` ? – [Alex S.](#) Mar 10, 2022 at 5:34



3



I wanted a splash screen that would display until the main program form was ready to be displayed, so timers etc were no use to me. I also wanted to keep it as simple as possible. My application starts with (abbreviated):

```
static void Main()  
{  
    Splash frmSplash = new Splash();
```



```
frmSplash.Show();  
Application.Run(new ReportExplorer(frmSplash));  
}
```

Then, ReportExplorer has the following:

```
public ReportExplorer(Splash frmSplash)  
{  
    this.frmSplash = frmSplash;  
    InitializeComponent();  
}
```

Finally, after all the initialisation is complete:

```
if (frmSplash != null)  
{  
    frmSplash.Close();  
    frmSplash = null;  
}
```

Maybe I'm missing something, but this seems a lot easier than mucking about with threads and timers.

Share Improve this answer

answered Sep 6, 2018 at 9:45

Follow



AndrewB

59 ● 8

good for simple cases, where threading is not important or necessary. Great for scenarios where the initial form is a Login and subsequent form is the "Main" form but there is a lot of data loads in between Login and Main form

– [GoldBishop](#) Feb 18, 2019 at 19:20

Only problem is when you have some animation on the splash. For exemple, a progress bar with Marquee style won't



Here's my **2023** take on a **2011** question.

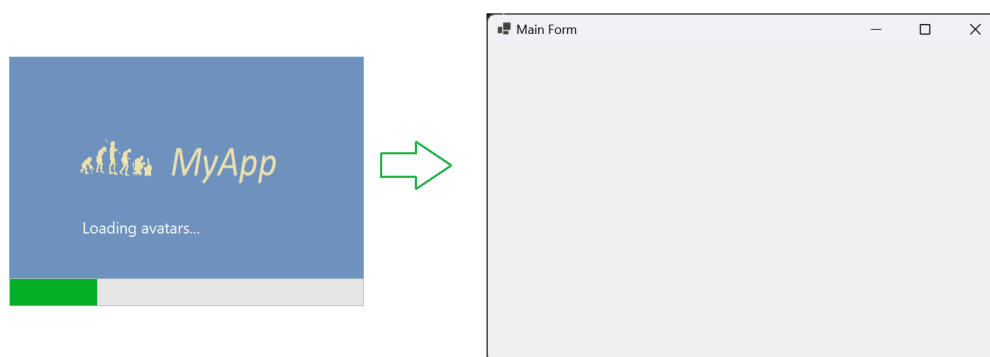
3



Over time, I've done this many times in many ways. The approach that currently use:



- Force the main form `Handle` creation so that the message that creates the splash can be posted into the main form's message queue using `BeginInvoke`. This allows the main form ctor to return. Ordinarily the handle (the native `hwnd`) doesn't come into existence until it's shown. Therefore, it needs to be coerced while it's still hidden.
- Override the `SetVisibleCore()` preventing the main window from becoming visible until the Splash has finished processing.



Main form is hidden until splash screen closes.

```
public partial class MainForm : Form
{
    public MainForm()
    {
```

```

        InitializeComponent();

        Debug.Assert(!IsHandleCreated, "Expecting handle to be created");
        // Ordinarily we don't get the handle until the window is shown. But we want it now.
        _ = Handle;
        Debug.Assert(IsHandleCreated, "Expecting handle to be created");

        // Call BeginInvoke on the new handle so as not to block the main thread
        BeginInvoke(new Action(() => execSplashFlow()))
    }
    protected override void SetVisibleCore(bool value)
    {
        base.SetVisibleCore(value && _initialized);
    }

    bool _initialized = false;

    private void execSplashFlow()
    {
        using (var splash = new SplashForm())
        {
            splash.ShowDialog();
        }
        _initialized = true;
        WindowState = FormWindowState.Maximized;
        Show();
    }
}

```

Splash Example

The async initialization can be performed in the Splash class itself *or* it can fire events causing the main app to do things. Either way, when it closes itself the main form will set the `_initialized` bool to `true` and it is now capable of becoming visible.

```

public partial class SplashForm : Form
{
    public SplashForm()
    {

```

```

    {
        InitializeComponent();
        StartPosition = FormStartPosition.CenterScreen;
        FormBorderStyle = FormBorderStyle.None;
    }
    protected async override void OnVisibleChanged(EventArgs e)
    {
        base.OnVisibleChanged(e);
        if (Visible)
        {
            labelProgress.Text = "Updating installation";
            progressBar.Value = 5;
            await Task.Delay(1000);
            progressBar.Value = 25;

            // SIMULATED background task like making a
            // database (long-running task that doesn't
            labelProgress.Text = "Loading avatars...";
            await Task.Delay(1000);

            labelProgress.Text = "Fetching game history";
            progressBar.Value = 50;
            await Task.Delay(1000);
            labelProgress.Text = "Initializing scenario";
            progressBar.Value = 75;
            await Task.Delay(1000);
            labelProgress.Text = "Success!";
            progressBar.Value = 100;
            await Task.Delay(1000);
            DialogResult = DialogResult.OK;
        }
    }
}

```

Share Improve this answer

edited Feb 22, 2023 at 17:10

Follow



LarsTech

81.5k ● 14 ● 158 ● 231

answered Feb 22, 2023 at 14:28



IV.

8,197 ● 2 ● 14 ● 27

2 [Clone](#) demo code. – IV. Feb 22, 2023 at 14:29



create splash

2



```
private void timer1_Tick(object sender, EventArgs e)
{
    counter++;
    progressBar1.Value = counter * 5;
    // label2.Text = (5*counter).ToString();
    if (counter == 20)
    {
        timer1.Stop();
        this.Close();
    }
}

this.AutoScaleDimensions = new System.Drawing.SizeF(6F
this.AutoScaleMode = System.Windows.Forms.AutoScaleModeMod
this.BackColor = System.Drawing.SystemColors.GradientI
this.ClientSize = new System.Drawing.Size(397, 283);
this.ControlBox = false;
this.Controls.Add(this.label2);
this.Controls.Add(this.progressBar1);
this.Controls.Add(this.label1);
this.ForeColor = System.Drawing.SystemColors.ControlLi
this.FormBorderStyle = System.Windows.Forms.FormBorderStyle
this.Name = "Splash";
this.ShowIcon = false;
this.ShowInTaskbar = false;
this.StartPosition = System.Windows.Forms.FormStartPosition
this.ResumeLayout(false);
this.PerformLayout();
```

Then in your application

```
sp = new Splash();  
sp.ShowDialog();
```

Share Improve this answer

Follow

edited Sep 21, 2012 at 21:46



Andrew Barber

40.1k ● 20 ● 96 ● 124

answered Sep 11, 2012 at 6:27



Ashekur Rahman Molla
Asik

93 ● 1 ● 7



2



The other answers here cover this well, but it is worth knowing that there is built in functionality for splash screens in Visual Studio: If you open the project properties for the windows form app and look at the Application tab, there is a "Splash screen:" option at the bottom. You simply pick which form in your app you want to display as the splash screen and it will take care of showing it when the app starts and hiding it once your main form is displayed.

You still need to set up your form as described above (with the correct borders, positioning, sizing etc.)

Share Improve this answer

Follow


answered May 6, 2014 at 7:27



tomRedox

30.3k ● 29 ● 124 ● 165

This sounds ideal, but I can't see this option (in VS 2013).
The only references I can find online are only for Visual

Basic, so perhaps it's only supported for that? – [Giles](#) Jul 25, 2016 at 16:26 

- 1 Hi @Giles, that could well be the case, I've only used it on VB apps – [tomRedox](#) Jul 25, 2016 at 23:11



None of the other answers gave me exactly what I was looking for. Read on for my solution to the problem.

2



I want a splash screen to fade in from 0% opacity to 100% opacity while things boot up, with a minimum display time of 2000ms (to allow the full fade in effect to show). Once everything is ready, I want the splash screen to display for a further 500ms while the main screen displays behind the splash screen. Then I want the splash screen to go away, leaving the main screen running.



Note that I use the MVP pattern for winforms. If you don't use MVP, you will need to simplify the below example a little.

Long story short, you need to create an `AppContext` class that inherits from `ApplicationContext`. I have put this in my `Program.cs` as below:

```
static class Program
{
    /// <summary>
    /// The main entry point for the application.
    /// </summary>
    [STAThread]
    static void Main()
    {
        Application.SetHighDpiMode(HighDpiMode.SystemA
```

```

        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(
            false);
        Application.Run(new ApplicationContext());
    }
}

public class ApplicationContext : ApplicationContext
{
    private IMainPresenter _mainPresenter;
    private bool _ready;

    public ApplicationContext()
    {
        _ready = false;

        using (ISplashPresenter splashPresenter = new
            SplashScreenView())
        {
            Stopwatch sw = Stopwatch.StartNew();

            _mainPresenter = new MainPresenter(new Mai
            _mainPresenter.Closed += MainPresenter_Clo

            new Thread(() =>
            {
                // !!! Do work here !!!

                if (sw.ElapsedMilliseconds < 2000)
                    Thread.Sleep(2000 - (int)sw.El

                _ready = true;
            })
            .Start();

            while (!_ready)
            {
                Application.DoEvents();
                Thread.Sleep(1);
            }

            _mainPresenter.Show();

            _ready = false;
        }
    }
}

```

```

        new Thread(() =>
        {
            Thread.Sleep(500);

            _ready = true;
        })
        .Start();

        while (!_ready)
        {
            Application.DoEvents();
            Thread.Sleep(1);
        }
    }

    private void MainPresenter_Closed(object sender, EventArgs e)
    {
        ExitThread();
    }
}

```

There are several implementation specific details that I haven't gone into here, such as `ISplashPresenter` implementing `IDisposable` and exactly how the fade in is managed; if enough people request it I will edit this answer to include a complete example.

Share Improve this answer

answered Jul 13, 2021 at 13:36

Follow



user5151179

585 ● 2 ● 10



First you should create a form with or without Border (border-less is preferred for these things)

1



```
public class SplashForm : Form
{
    Form _Parent;
    BackgroundWorker worker;
    public SplashForm(Form parent)
    {
        InitializeComponent();
        BackgroundWorker worker = new BackgroundWorker();
        this.worker.DoWork += new
        System.ComponentModel.DoWorkEventHandler(this.worker _
        backgroundWorker1.RunWorkerAsync());
        _Parent = parent;
    }
    private void worker _DoWork(object sender, DoWorkE
    {
        Thread.sleep(500);
        this.hide();
        _Parent.show();
    }
}
```

At Main you should use that

```
static class Program
{
    [STAThread]
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRendering
        Application.Run(new SplashForm());
    }
}
```

Share Improve this answer

answered Oct 31, 2011 at 15:30

Follow



Rosmarine Popcorn

11k ● 11 ● 61 ● 90

-
- 2 This is bad. This will make your SplashScreen, the Main Form of your application. This means that when you close it, the application will stop – [Matias Cicero](#) Jun 19, 2014 at 15:16
-



1

Maybe a bit late to answer but i would like to share my way. I found an easy way with threads in the main program for a winform application.



Lets say you have your form "splashscreen" with an animation, and your "main" which has all your application code.



```
[STAThread]
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(
        Thread mythread;
        mythread = new Thread(new ThreadStart(ThreadLo
        mythread.Start());
        Application.Run(new MainForm(mythread));
    }

    public static void ThreadLoop()
    {
        Application.Run(new SplashScreenForm());
    }
}
```

In your main form in the constructor:

```
public MainForm(Thread splashscreenthread)
{
    InitializeComponent();

    //add your constructor code
}
```

```
splashscreenthread.Abort();  
}
```

This way the splashscreen will last just the time for your main form to load.

Your splashscreen form should have his own way to animate/display information. In my project my splashscreen start a new thread, and every x milliseconds it changes his main picture to another which is a slightly different gear, giving the illusion of a rotation.

example of my splashscreen:

```
int status = 0;  
private bool IsRunning = false;  
public Form1()  
{  
    InitializeComponent();  
    StartAnimation();  
}  
  
public void StartAnimation()  
{  
    backgroundWorker1.WorkerReportsProgress = false;  
    backgroundWorker1.WorkerSupportsCancellation = true;  
    IsRunning = true;  
    backgroundWorker1.RunWorkerAsync();  
}  
  
public void StopAnimation()  
{  
    backgroundWorker1.CancelAsync();  
}  
  
delegate void UpdatingThreadAnimation();  
public void UpdateAnimationFromThread()
```

```

{
    try
    {
        if (label1.InvokeRequired == false)
        {
            UpdateAnimation();
        }
        else
        {
            UpdatingThreadAnimation d = new
UpdatingThreadAnimation(UpdateAnimationFromThread);
            this.Invoke(d, new object[] { });
        }
    }
    catch(Exception e)
    {
    }
}

private void UpdateAnimation()
{
    if(status ==0)
    {
        // mypicture.image = image1
    }else if(status ==1)
    {
        // mypicture.image = image2
    }
    //doing as much as needed

    status++;
    if(status>1) //change here if you have more im
cycle of images
    {
        status = 0;
    }
    this.Refresh();
}

private void backgroundWorker1_DoWork(object sender, D
{
    BackgroundWorker worker = sender as Background

```

```
while (IsRunning == true)
{
    System.Threading.Thread.Sleep(100);
    UpdateAnimationFromThread();
}
```

Hope this will help some people. Sorry if i have made some mistakes. English is not my first language.

Share Improve this answer

edited Dec 3, 2018 at 15:39

Follow

answered Dec 3, 2018 at 15:30



Ornithorix

11 ● 2

"splashscreenthread.Abort();" doesn't seems to always abort the thread. – [Maxter](#) Dec 16, 2019 at 17:03



Here is the easiest way of creating a splash screen:

0

First of all, add the following line of code before the namespace in Form1.cs code:



using System.Threading;



Now, follow the following steps:



1. Add a new form in you application
2. Name this new form as FormSplashScreen

3. In the BackgroundImage property, choose an image from one of your folders
4. Add a progressBar
5. In the Dock property, set it as Bottom
6. In MarksAnimationSpeed property, set as 50
7. In your main form, named as Form1.cs by default, create the following method:

```
private void StartSplashScreen()  
{  
    Application.Run(new Forms.FormSplashScreen())  
}
```

8. In the constructor method of Form1.cs, add the following code:

```
public Form1()  
{  
    Thread t = new Thread(new ThreadStart(StartSp  
t.Start();  
    Thread.Sleep(5000);  
  
    InitializeComponent();//This code is automati  
Studio  
  
    t.Abort();  
}
```

9. Now, just run the application, it is going to work perfectly.

Share Improve this answer

edited Feb 13, 2021 at 17:33

Follow

answered Feb 13, 2021 at 15:40



RogerAguiar

1 ● 1



Try this code

-2



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

    private void timer1_Tick(object sender, EventArgs e)
    {
        progressBar1.Increment(1);
        if (progressBar1.Value == 100)
        {
            timer1.Stop();
            this.Hide();
            Form frm = new login();
            frm.Show();
        }
    }
}
```

Share Improve this answer

edited May 10, 2016 at 10:42

Follow



Mostafiz

7,349 ● 3 ● 29 ● 42

answered Mar 17, 2016 at 10:22



kuldeep kanaujia

7

4 This answer could use some explanation. – Raidri Mar 17, 2016 at 10:48



Try This:

-4



```
namespace SplashScreen
{
    public partial class frmSplashScreen : Form
    {
        public frmSplashScreen()
        {
            InitializeComponent();
        }

        public int LeftTime { get; set; }

        private void frmSplashScreen_Load(object sender
        {
            LeftTime = 20;
            timer1.Start();
        }

        private void timer1_Tick(object sender, EventArgs
        {
            if (LeftTime > 0)
            {
                LeftTime--;
            }
            else
            {
                timer1.Stop();
                new frmHomeScreen().Show();
                this.Hide();
            }
        }
    }
}
```

Share Improve this answer

Follow

edited Aug 20, 2018 at 1:19



Rob ♦

27.3k ● 16 ● 87 ● 102

answered Aug 16, 2018 at 13:20



JIGS SOKI

JIGS 1 ● 1

