

---

Subject: Calling windows not in main.cpp

Posted by [Mystery Smith](#) on Sun, 22 Nov 2009 09:44:13 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

Hi all,

I've run across yet another problem (learning C++/U++ is more difficult than I first imagined.) Because my project is quite large and I'm such a novice, I thought it would be a good idea to separate my code into separate files with intuitive names. I decided to try doing it with my Splash Screen and the code looks like this.

main.cpp

```
/* BRAIN BOX MAIN.CPP */
```

```
#include "BrainBox.h"
```

```
....
```

```
/* Run the program Loop */
```

```
GUI_APP_MAIN
```

```
{  
    splashScreen().Run();  
    Ctrl::ProcessEvent();
```

```
    // Open the mainWindow  
    mainWindow().Run();  
}
```

BrainBox.h

```
#ifndef _BrainBox_BrainBox_h_  
#define _BrainBox_BrainBox_h_
```

```
#include <CtrlLib/CtrlLib.h>  
#include <Core/Core.h>  
#include <stdio.h>
```

```
using namespace Upp;
```

```
//Load Layouts
```

```
#define LAYOUTFILE <BrainBox/Layouts.lay>  
#include <CtrlCore/lay.h>
```

```
// Load Images
```

```
#define IMAGECLASS Images
#define IMAGEFILE <BrainBox/images.iml>
#include <Draw/iml.h>
```

```
// Declarations
//Guessing some sort of declaration for the function needs to be made here?
```

```
#endif
```

```
SplashScreen.cpp
/* Splash Screen File */
#include <CtrlLib/CtrlLib.h>
```

```
using namespace Upp;
```

```
struct splashScreen : TopWindow
{
    splashScreen()
    {
        Title("Splash Screen");
        SetRect(0, 0, 200, 300);
    }
};
```

I can't link these files together and I'm assuming it has something to do with the Header File. Can anybody help? Sorry I'm so bad at this.

Thanks so much, in advance

---

Subject: Re: Calling windows not in main.cpp  
Posted by [mirek](#) on Sun, 22 Nov 2009 22:08:15 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Mystery Smith wrote on Sun, 22 November 2009 04:44  
/\* Splash Screen File \*/  
#include "BrainBox.h" // <<<----- HERE!

```
using namespace Upp;
```

```
struct splashScreen : TopWindow
{
    splashScreen()
    {
```

```
Title("Splash Screen");
SetRect(0, 0, 200, 300);
}
};
```

---

---

Subject: Re: Calling windows not in main.cpp  
Posted by [Mystery Smith](#) on Sun, 22 Nov 2009 23:16:01 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Hey thanks for your help, I fixed that up but I'm still getting the same error.

```
C:\MyApps\BrainBox\main.cpp(119) : error C2228: left of '.PopUp' must have class/struct/union
type is "unknown-type"
C:\MyApps\BrainBox\main.cpp(119) : error C3861: 'splashScreen': identifier not found
main.cpp
```

Do you know what is causing this?

---

---

Subject: Re: Calling windows not in main.cpp  
Posted by [mr\\_ped](#) on Mon, 23 Nov 2009 08:56:19 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

You have to have already defined what is "splashScreen" here:  
main.cpp: "splashScreen().Run();"

You can use Build / Preprocess (Alt+F7) to see what the compiler is trying to compile after he resolves the includes, you will see there's nothing about splashScreen definition in BrainBox.h.

Common C++ way of doing things is to have: (I'm writing it by hand, so sorry for any typos) (ignore all the "bar" things firstly before you get the idea how foo works, then re-read and try to resolve "bar")

foo.h:

```
class bar; //a trick how to use class "bar" in other header without it's actual definition, see a bit
down
```

```
class foo
{
protected: //next definitions will be visible only inside the class and derived classes
    int x; //variable definition
    bar *pointer_to_class_bar; //you can use pointer even without actual definition of "bar" being
```

known here

```
public:    //next definitions will be visible outside of the class
foo() : pointer_to_class_bar(NULL)
{
    //this is constructor - with (empty) code defined HERE */

    //pointer_to_class_bar->DoSomething();
    //^^ this would produce error, because compiler doesn't know here the bar has method
    "DoSomething"
}

int fwd_foo( int ); //this is forward definition of function, without actual code
};
```

foo.cpp:

```
#include "bar.h" //in this file we will need to know bar class,
                //so include it's full definition (similar file to foo.h)
#include "foo.h" //include public definitions of foo class

//here comes the rest of foo implementation, like the missing body of int fwd_foo( int ) method

int foo::fwd_foo( int ) //notice how the name of method is preceded by class name
{
    if ( pointer_to_class_bar != NULL ) pointer_to_class_bar->SomeMethodOfBar();
    //^^ here it will work
    return 7;
}
```

other.cpp:

```
#include "foo.h" //we want to use foo class

... inside some function like main for example ...
{
    foo Foo; //create instance of class foo at function stack memory
    int y = Foo.fwd_foo( 0 );
    //^^ compiler knows how to call this method, because it was forward-defined in the "foo.h", and
    we did include that file
    //int my_x = Foo.x;
    //^^ error: while compiler knows the Foo has "x", you can't access it directly like this,
    //it's protected only for usage from withing Foo class family

    //bar Bar; //this will create error, because compiler doesn't know full definition of class bar
    bar * another_bar_pointer; //but this will work, because it was also forward-defined in foo.h,
```

```
//but the usage is very limited, like only passing around the pointer value
/* notice we don't need to include full bar.h to do that,
   so we will save compile time and when you edit bar.h,
   this file does not need to recompile, only files which do include full bar.h,
   like foo.cpp
*/
}
```

The basic idea is, that C++ compiler has to know EVERYTHING you are using in the file which is being compiled. It will not look at other files (unless you direct him by #include).

This is different from Java, where you only write the code parts (like .cpp files in C++), and you then do "import foo" and the compiler will take a look at foo and figure it out.

C++ is dumber in this aspect, the compiler has only single file during compilation (the one which you can see after Alt+F7), so what you use there, must be defined already there.

In your code you define the splashscreen in SplashScreen.cpp, but the main.cpp has no idea what you are talking about, it would have to include the .cpp file to see it (of course that's not the proper solution, the proper solution is to define the needed bits in some header file, and this header file include in both main.cpp and SplashScreen.cpp, and everywhere else where you want to work with splashscreen struct (as you are deriving it from TopWindow, I would rather suggest to make it "class", unless you have very good reasons to have it as struct (I can't think of any)).

Try to search some C++ language tutorial about definitions of classes (.h files) and implementation (.cpp) to make sure you fully understand that concept. It's not difficult at all, once you get the idea, it's very natural. (natural in "computer-dumb" way, the Java is natural in "human-smart" way in this aspect)

---