CS 2630: Data Structures

Fall 2021

Lab 04 (10 points)

Due: Thursday 10/28 before 11:00pm

Objectives

After this lab assignment, students should be able to:

- Create a Visual Studio GUI Project using CLR

Instructions

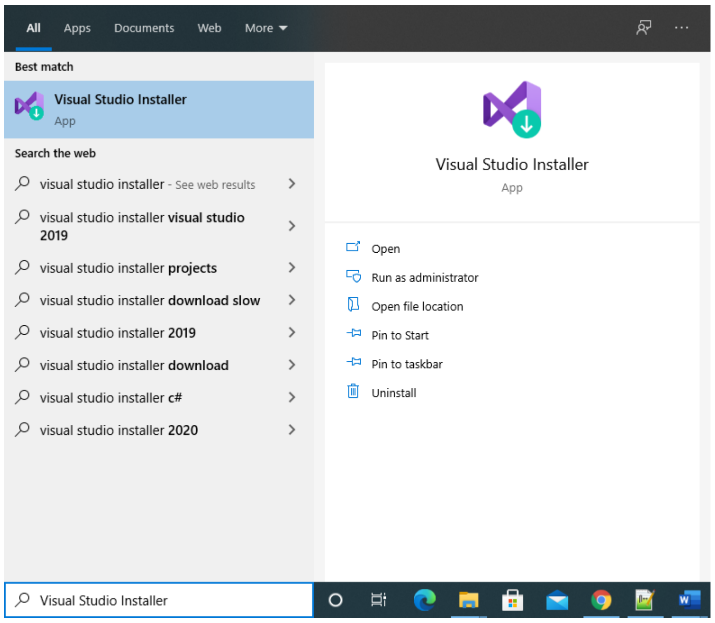
For Lab 4, there are four tasks for you to complete:

- A tutorial that shows you how to create an empty CLR Project in Visual Studio
- Three tasks that introduce you to adding and controlling various elements on a Windows Form
    - Task 1 - Adding a "ball" graphic
    - Task 2 - Animating the "ball" graphic
    - Task 3 - Adding a stop condition for the "ball" animation

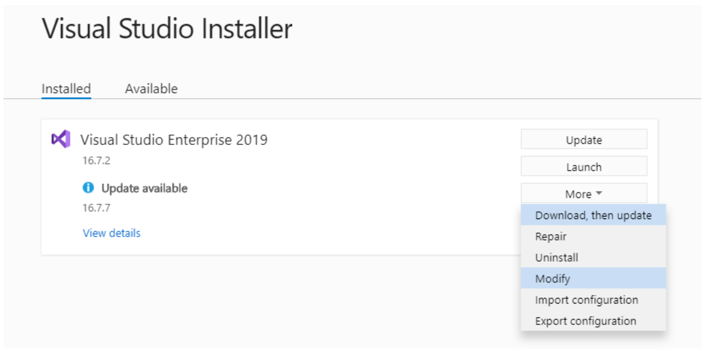
Complete all four tasks and then submit your project according to the Submission instructions at the end of this document.

Visual Studio - Creating an Empty CLR Project

Your version of Visual Studio most likely has not been installed with CLI support. So, you must modify your installation to include it. Launch the Visual Studio Installer –search for Visual Studio in the windows search bar and it should appear in the list of results:


All
Apps
Documents
Web
More
Best match
Visual Studio Installer
App
Search the web
visual studio installer - See web results
visual studio installer visual studio 2019
visual studio installer projects
visual studio installer download slow
visual studio installer 2019
visual studio installer download
visual studio installer c#
visual studio installer 2020
Visual Studio Installer
App
Open
Run as administrator
Open file location
Pin to Start
Pin to taskbar
Uninstall
Visual Studio Installer

Once the installer launches, select "Modify" to add component to Visual Studio. This option may be under the "More" selection, as shown below:



Visual Studio Installer

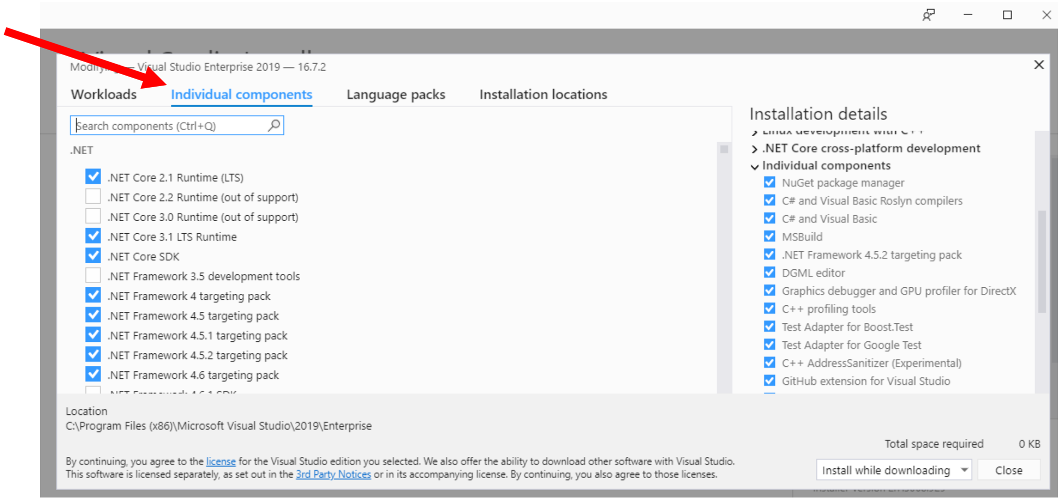
Installed Available

Visual Studio Enterprise 2019
16.7.2

Update available
16.7.7

View details

You should now see a dialog allowing you to add installation components. Select "Individual Components."



Modifying — Visual Studio Enterprise 2019 — 16.7.2

.NET

- [✓] .NET Core 2.1 Runtime (LTS)
- [ ] .NET Core 2.2 Runtime (out of support)
- [ ] .NET Core 3.0 Runtime (out of support)
- [✓] .NET Core 3.1 LTS Runtime
- [✓] .NET Core SDK
- [ ] .NET Framework 3.5 development tools
- [✓] .NET Framework 4 targeting pack
- [✓] .NET Framework 4.5 targeting pack
- [✓] .NET Framework 4.5.1 targeting pack
- [✓] .NET Framework 4.5.2 targeting pack
- [✓] .NET Framework 4.6 targeting pack

Installation details

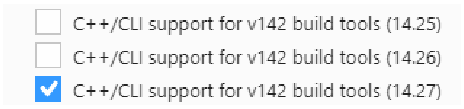
- Linux development with C++
- .NET Core cross-platform development
- Individual components
  - [✓] NuGet package manager
  - [✓] C# and Visual Basic Roslyn compilers
  - [✓] C# and Visual Basic
  - [✓] MSBuild
  - [✓] .NET Framework 4.5.2 targeting pack
  - [✓] DGML editor
  - [✓] Graphics debugger and GPU profiler for DirectX
  - [✓] C++ profiling tools
  - [✓] Test Adapter for Boost.Test
  - [✓] Test Adapter for Google Test
  - [✓] C++ AddressSanitizer (Experimental)
  - [✓] GitHub extension for Visual Studio

Location
C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise

By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

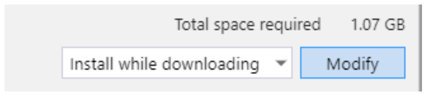
Total space required 0 KB

Check the box to the left of the bottom-most "C++/CLI support for v142 build tools"



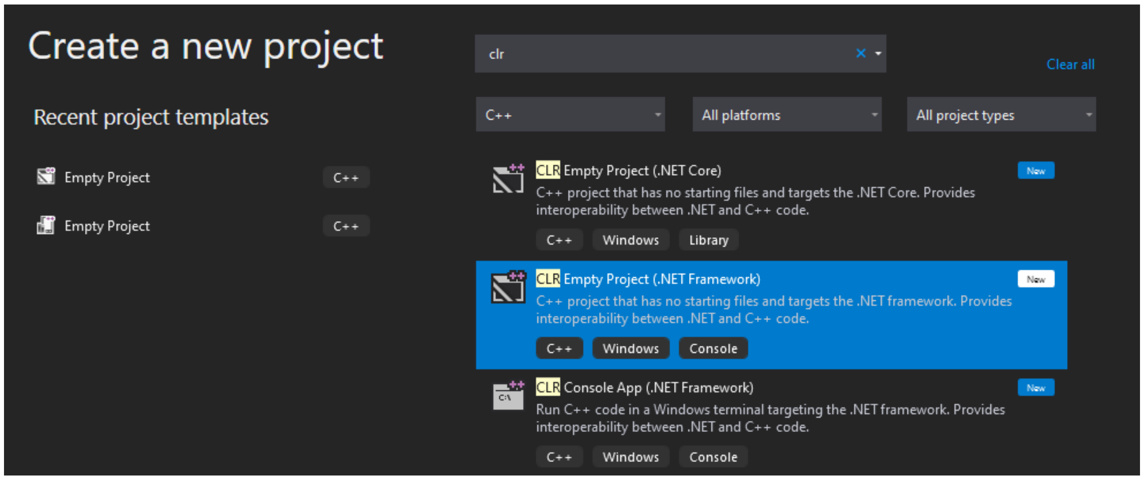
- [ ] C++/CLI support for v142 build tools (14.25)
- [ ] C++/CLI support for v142 build tools (14.26)
- [✓] C++/CLI support for v142 build tools (14.27)

Now click "Modify" on the lower-right corner of the dialog:



Total space required 1.07 GB

The installation may take some time, depending on your environment. Once the installation is finished, launch Visual Studio. You may now create a new project that allows you to work with the .NET framework. So, create a new project and make sure your screen looks like the image below:



Create a new project

clr

Recent project templates

Empty Project C++
Empty Project C++

CLR Empty Project (.NET Core) New
C++ project that has no starting files and targets the .NET Core. Provides interoperability between .NET and C++ code.
C++ Windows Library

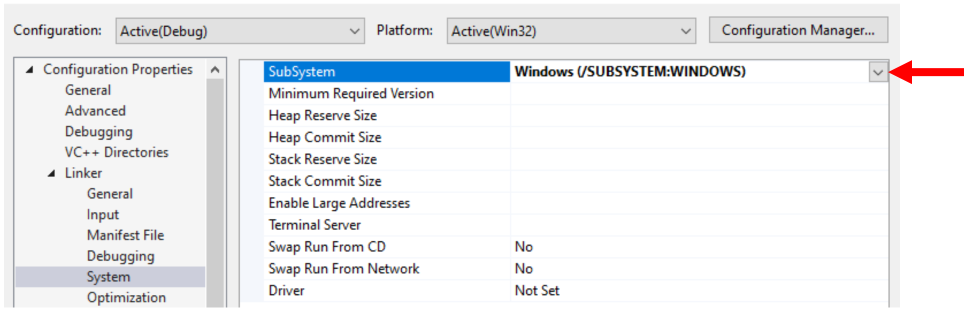
CLR Empty Project (.NET Framework) New
C++ project that has no starting files and targets the .NET framework. Provides interoperability between .NET and C++ code.
C++ Windows Console

CLR Console App (.NET Framework) New
Run C++ code in a Windows terminal targeting the .NET framework. Provides interoperability between .NET and C++ code.
C++ Windows Console

Specifically, select "C++" as the language and type "CLR" into the search box at the top. You will get several options for CLR projects. Select "CLR Empty Project (.NET Framework)." Depending on your version of Visual Studio, you may just see "CLR Empty Project." – this is fine, select that option if it's all you see. **You MUST name your project with a compatible C++ name, such as "Lab4." Don't add any spaces, don't start with a number, and don't use any characters except for: alphanumeric characters and underscores.**

Once your project has been created, there are several additional configuration parameters we need to change:

1. Navigate to Project→Properties→Linker→System→SubSystem
   Change this value to "Windows" – after doing this, you can select the down arrow to the right and specifically select "Windows (/SUBSYSTEM:WINDOWS)"

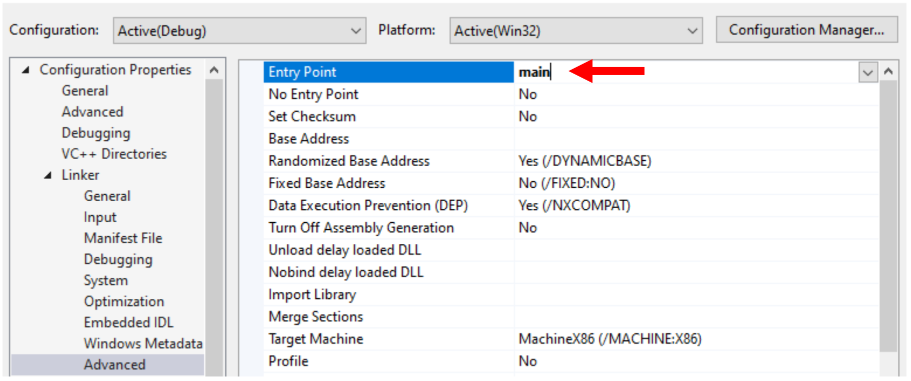


Configuration: Active(Debug) Platform: Active(Win32) Configuration Manager...

- Configuration Properties
  - General
  - Advanced
  - Debugging
  - VC++ Directories
  - Linker
    - General
    - Input
    - Manifest File
    - Debugging
    - System
    - Optimization

SubSystem	Windows (/SUBSYSTEM:WINDOWS)
Minimum Required Version	
Heap Reserve Size	
Heap Commit Size	
Stack Reserve Size	
Stack Commit Size	
Enable Large Addresses	
Terminal Server	
Swap Run From CD	No
Swap Run From Network	No
Driver	Not Set

2. Now, navigate to Project→Properties→Linker→Advanced→Entry Point
   Change this value to "main"



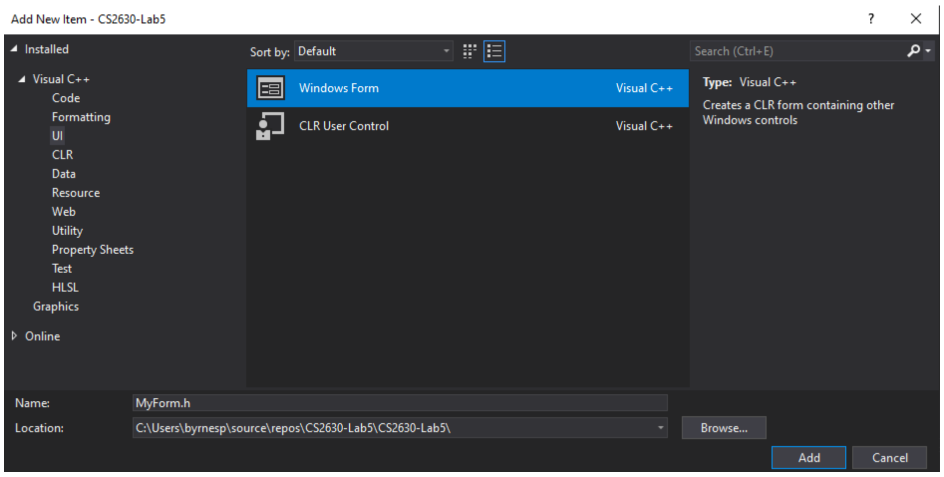
Configuration: Active(Debug) Platform: Active(Win32) Configuration Manager...

- Configuration Properties
  - General
  - Advanced
  - Debugging
  - VC++ Directories
  - Linker
    - General
    - Input
    - Manifest File
    - Debugging
    - System
    - Optimization
    - Embedded IDL
    - Windows Metadata
    - Advanced

Entry Point	main
No Entry Point	No
Set Checksum	No
Base Address	
Randomized Base Address	Yes (/DYNAMICBASE)
Fixed Base Address	No (/FIXED:NO)
Data Execution Prevention (DEP)	Yes (/NXCOMPAT)
Turn Off Assembly Generation	No
Unload delay loaded DLL	
Nobind delay loaded DLL	
Import Library	
Merge Sections	
Target Machine	MachineX86 (/MACHINE:X86)
Profile	No

3. Once these two parameters have been changed appropriately, select "Apply" and "OK."

Now that the project has been properly configured, let's add a Windows Form to the project. Navigate to Project→Add New Item…→UI→Windows Form→Add



Add New Item - CS2630-Lab5

- Installed
  - Visual C++
    - Code
    - Formatting
    - UI
    - CLR
    - Data
    - Resource
    - Web
    - Utility
    - Property Sheets
    - Test
    - HLSL
  - Graphics
- Online

Sort by: Default

Windows Form	Visual C++
CLR User Control	Visual C++

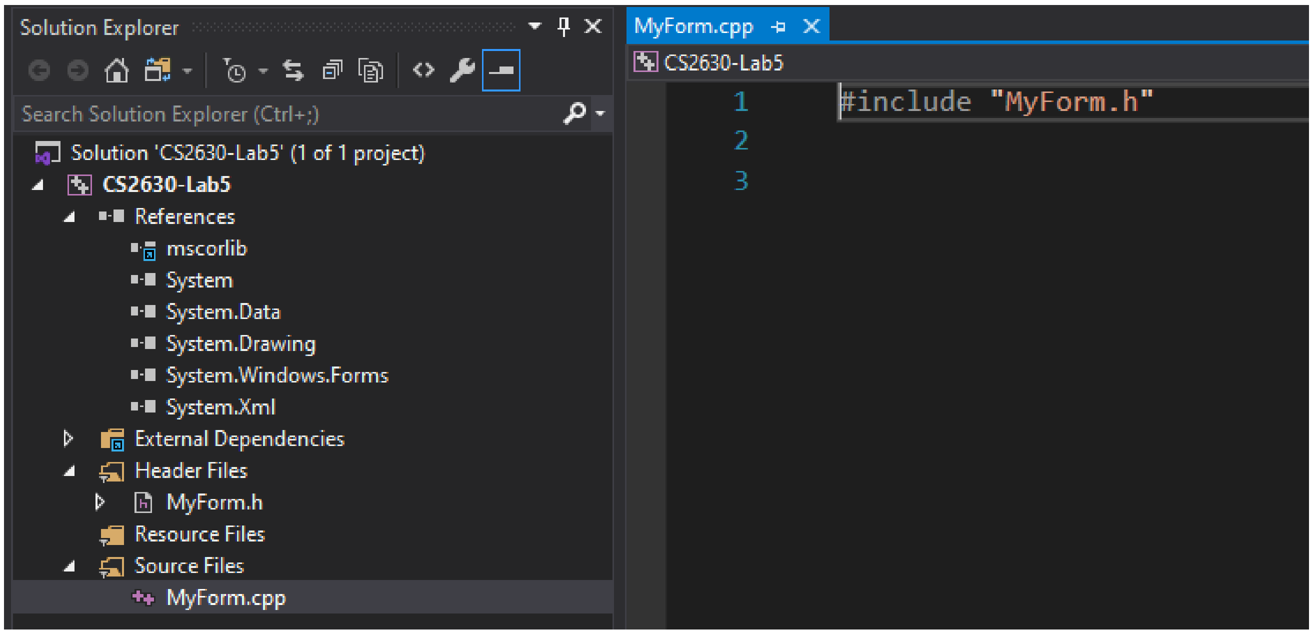
Search (Ctrl+E)

Type: Visual C++
Creates a CLR form containing other Windows controls

Name: MyForm.h
Location: C:\Users\byrnesp\source\repos\CS2630-Lab5\CS2630-Lab5\

After you click “Add” you will see an error in your form design view – **ignore the error**. You should notice that `MyForm.h` and `MyForm.cpp` have been added to your project.

Open MyForm.cpp. For now, it only contains one line and looks like the following screenshot:



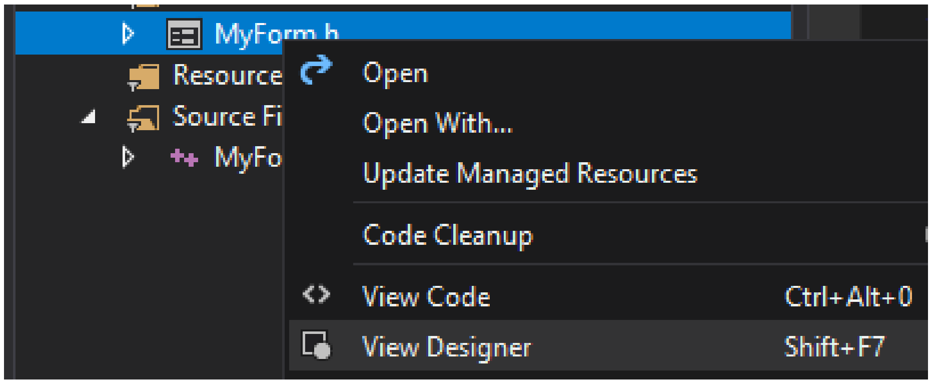
```
#include "MyForm.h"
```

Copy the following code after line `1`:

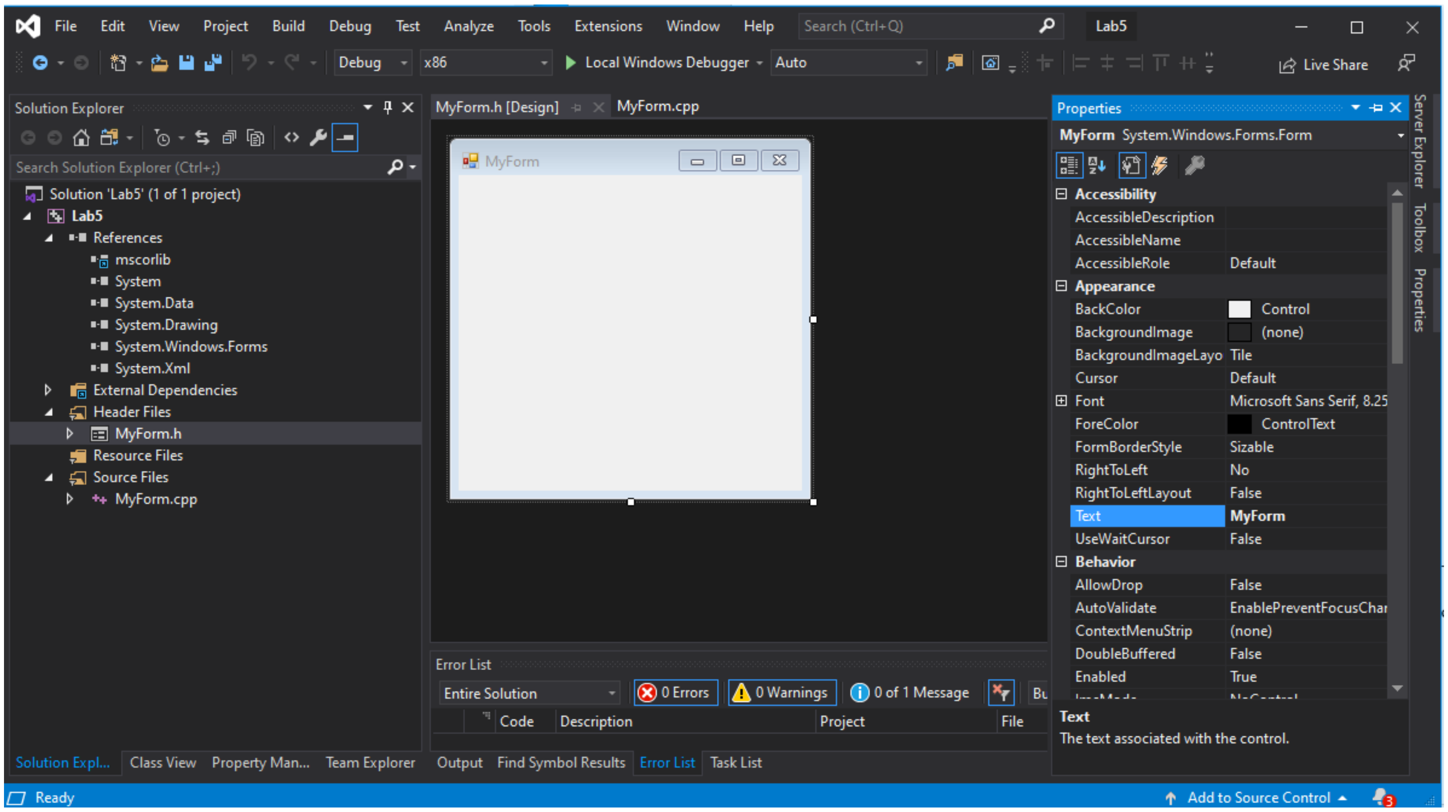
```
using namespace System;
using namespace System::Windows::Forms;

[STAThread]
int main( array<String^>^ args ) {
    Application::EnableVisualStyles();
    Application::SetCompatibleTextRenderingDefault(false);

    Lab4::MyForm form;
    Application::Run(%form);
}
```

Note that you will need to replace “Lab4” with the name of YOUR project. For your future projects, you will also replace “Lab4” with your project names. **Also, as a reminder, you should name your project with a compatible C++ variable name so that you don’t run into non-compatible object name problems!!!!** If your form name is not “`MyForm`”, you also need to change “`MyForm`” to whatever name you gave to your form.

Finally, save everything in your project. Close your project and reopen it. Right click “MyForm.h” in the solution explorer and select “View Designer”. (This is a known bug in VS 2017 and unfortunately, Microsoft decided it is a low priority bug to work on…)



You should now see the “MyForm.h [Design]” tab in Visual Studio:



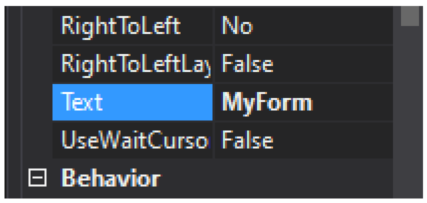
```
Solution 'Lab5' (1 of 1 project)
  Lab5
    References
      mscorlib
      System
      System.Data
      System.Drawing
      System.Windows.Forms
      System.Xml
    External Dependencies
    Header Files
      MyForm.h
    Resource Files
    Source Files
      MyForm.cpp
```

MyForm System.Windows.Forms.Form

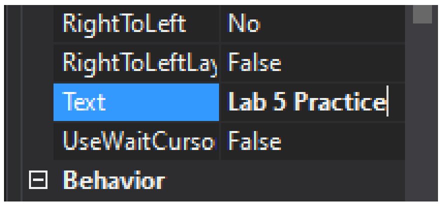
Accessibility	
AccessibleDescription	
AccessibleName	
AccessibleRole	Default
Appearance	
BackColor	Control
BackgroundImage	(none)
BackgroundImageLayo	Tile
Cursor	Default
Font	Microsoft Sans Serif, 8.25
ForeColor	ControlText
FormBorderStyle	Sizable
RightToLeft	No
RightToLeftLayout	False
Text	MyForm
UseWaitCursor	False
Behavior	
AllowDrop	False
AutoValidate	EnablePreventFocusChar
ContextMenuStrip	(none)
DoubleBuffered	False
Enabled	True

Text
The text associated with the control.

If you don’t see the **Properties** window (right side of figure), then you can right-click on the form and select “Properties.” In the Properties window, change the "**Text**" property to “**Lab4 Practice**”. If the Properties window keeps auto hiding, you can right click on it and choose “Dock”.

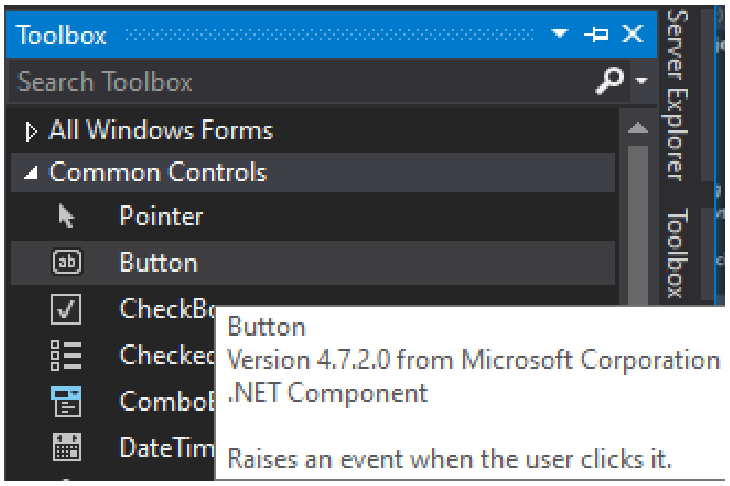


RightToLeft	No
RightToLeftLay	False
Text	MyForm
UseWaitCurso	False
Behavior	



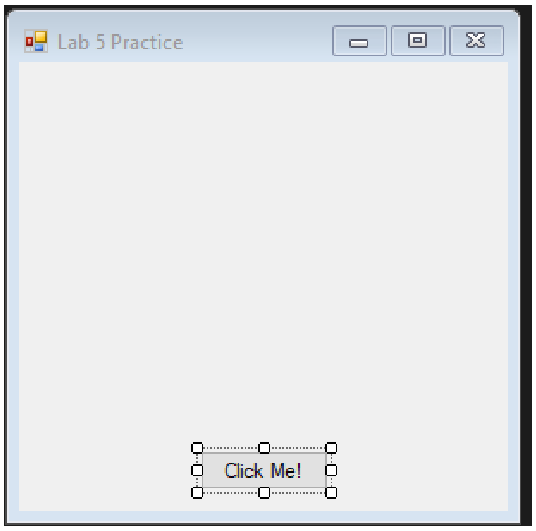
RightToLeft	No
RightToLeftLay	False
Text	Lab 5 Practice
UseWaitCurso	False
Behavior	

If you don't see the **ToolBox**, use the "View" menu to open it. Dock the **ToolBox** to the location you prefer. Select Common Controls→Button, then click on the form. The button will appear on the form. Move it to the bottom and use the Properties Window to change its text to: **Click Me!**



Button
Version 4.7.2.0 from Microsoft Corporation
.NET Component

Raises an event when the user clicks it.



Double click on the button on the form and you will be taken to an "**Event Handler**" for the button. This is the function that will run anytime a user of your program clicks on this button!

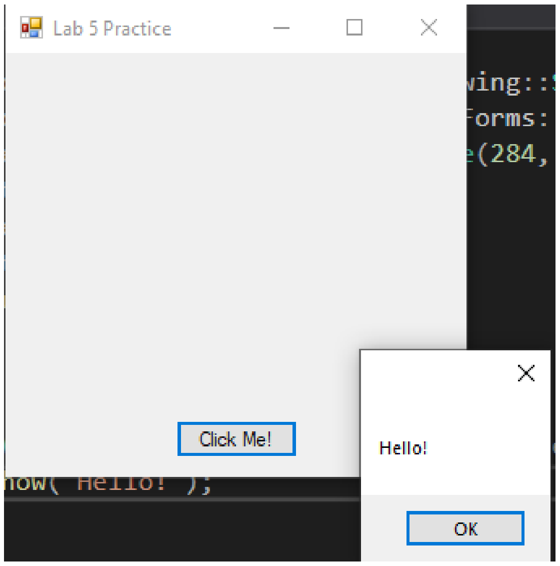
```
77  #pragma endregion
78      private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {
79      }
80      };
81  }
```

This is where you type the code you want to execute when the button is clicked. You may notice that all the code is being put in the .H file for the form. Microsoft apparently decided to make their C++ look at lot more like Java!

Add the following code to the Event Handler for the button: `MessageBox::Show( "Hello!" );`

```
77  #pragma endregion
78      private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {
79          MessageBox::Show("Hello!");
80      }
81      };
```

We now have a basic CLR project that we can build and run! So, build your project and then select Debug→Start Without Debugging. Your program should launch and present you with the Form you just created. Click on the button and make sure you see something like the screenshot below.



Hello!

Task 1 – Adding a "Ball" Element to a Windows Form

Now that we have a basic CRL project, we can start adding new elements. In the `MyForm.h` file, find the following line (around line 37):

```
37      private: System::Windows::Forms::Button^ button1;
```

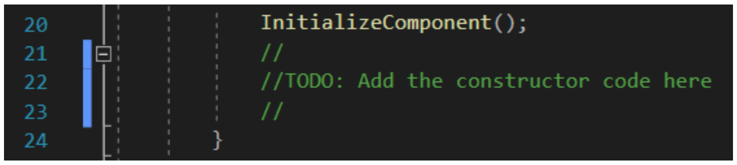
Type the following code after this line:

```
38      private:
39          int ball_x, ball_y;
40          static const int BALLSIZE = 40;
```

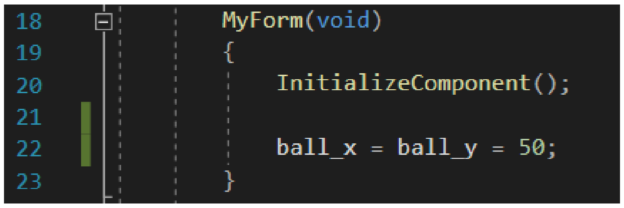
The observant student will note the "^". This is an extension to C++ supported only by Visual Studio.NET. Unlike standard C++, Studio.NET supports "managed" pointers – pointers that are garbage collected automatically just like the ones you were used to in Java. GUI development in Studio.NET uses such pointers. You can still use "->" to access elements (just like pointers declared using "*"), but you use `gcnew` to create objects rather than `new`.

Locate the `MyForm` default constructor and under the line: `InitializeComponent();`

Add the following statement: `ball_x = ball_y = 50;`

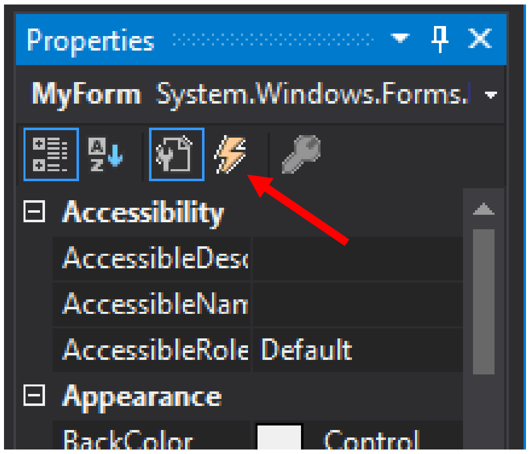


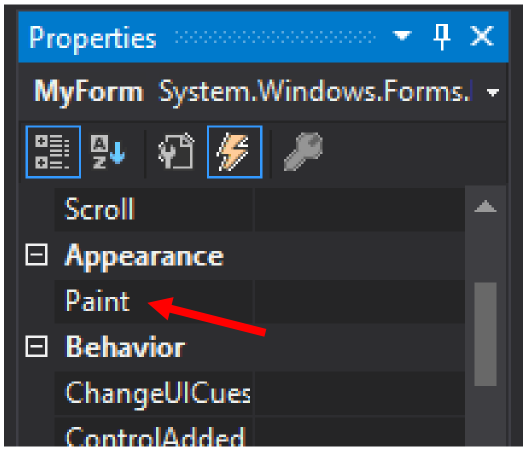
```
            InitializeComponent();
            //
            //TODO: Add the constructor code here
            //
        }
```



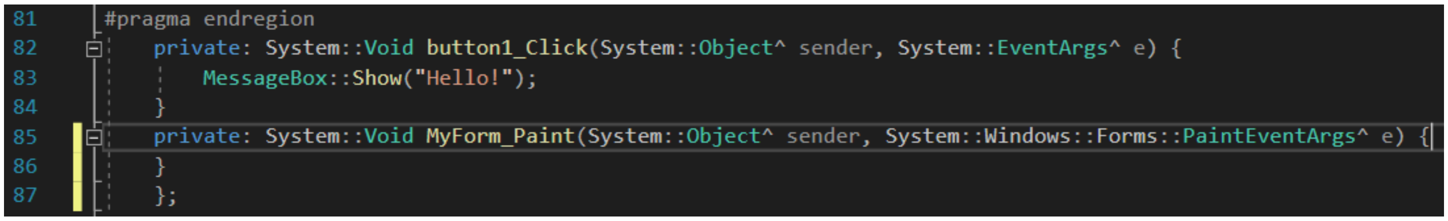
```
        MyForm(void)
        {
            InitializeComponent();

            ball_x = ball_y = 50;
        }
```

Go back to the design view of `MyForm`, make sure the whole form is "selected", then go to the Properties Window, and click on the "lightning bolt" to bring up events. You are going to add an event handler for the Paint event. Scroll until you see the Paint event (under Appearance) in the Properties Window, then double-click on the white part to the right of it.





You are going to add an event handler for the Paint event. Scroll until you see the Paint event (under Appearance) in the Properties Window, then double-click on the word "Paint". This will make an event handler for the paint method and throw you to the code again.

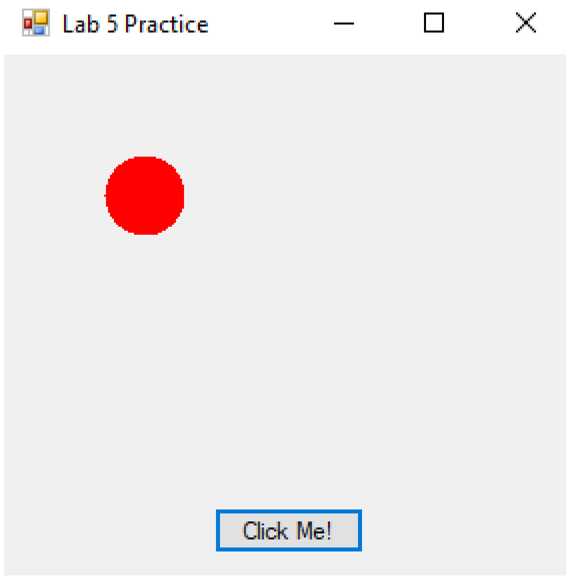


```
#pragma endregion
    private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {
        MessageBox::Show("Hello!");
    }
    private: System::Void MyForm_Paint(System::Object^ sender, System::Windows::Forms::PaintEventArgs^ e) {
    }
    };
```

The code for the paint method will be very similar to what we did in the paint method in Java, except that in Java, a "Graphics" object was passed as a parameter. For .NET, we need to get it from the `PaintEventArgs` parameter. So, put in the following code to draw a red ball:

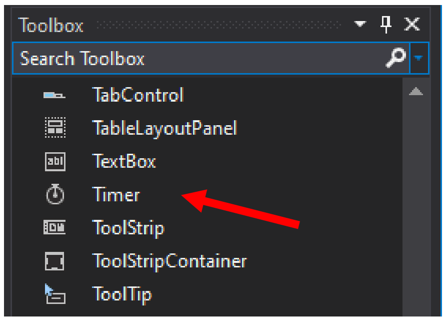
```
private: System::Void MyForm_Paint(System::Object^ sender, System::Windows::Forms::PaintEventArgs^ e) {
    Graphics^ g = e->Graphics;
    g->FillEllipse(gcnew SolidBrush(Color::Red), ball_x, ball_y,
        BALLSIZE, BALLSIZE);
}
```

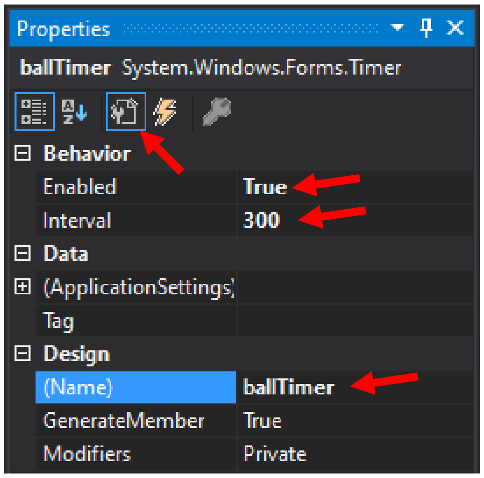
Build and run your project and make sure you can now see a red ball.



Task 2 – Animating the "Ball" Graphic

Switch back to the design view of `MyForm`. Take a Timer from the Toolbox (search under "All Window Forms") and put it on the form. In the Properties Window, with the timer highlighted, switch back to the Properties (click the little icon to the left of the lightning bolt). Set the "Enabled" property to be true and set the "Interval" to be 300 (which is 300 milliseconds). Change the "Name" property to `ballTimer`.



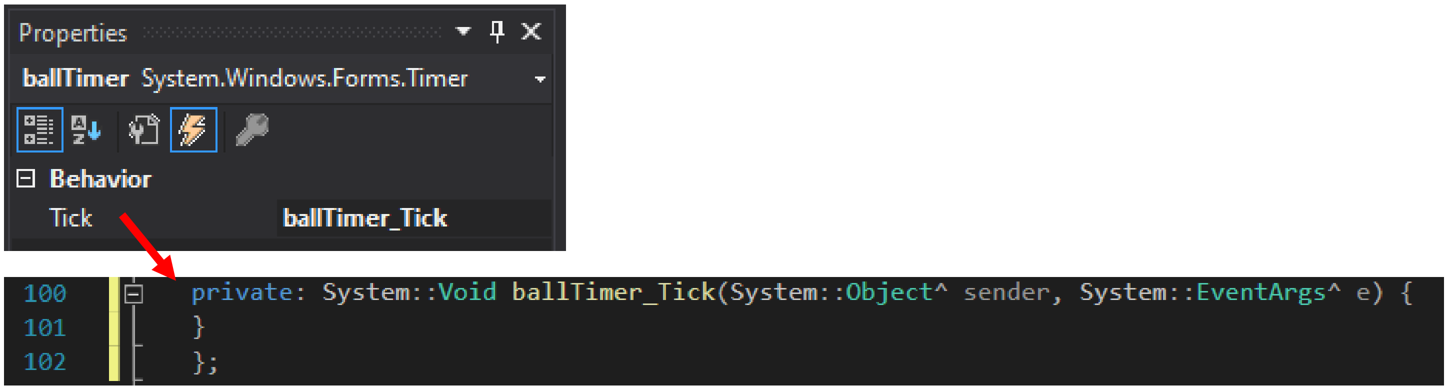


Properties

ballTimer System.Windows.Forms.Timer

Behavior	
Enabled	**True**
Interval	**300**
Data	
(ApplicationSettings)	
Tag	
Design	
(Name)	**ballTimer**
GenerateMember	True
Modifiers	Private

In the Properties Window, switch back to the Events. Double-click on "Tick". You are going to write a handler for when the timer "goes off".



Properties

ballTimer System.Windows.Forms.Timer

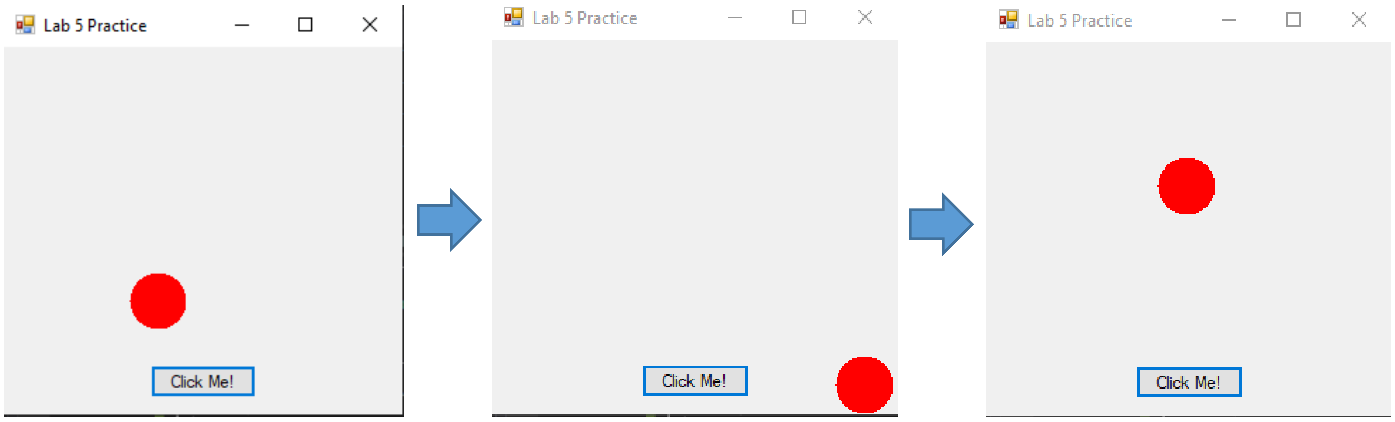
Behavior	
Tick	**ballTimer_Tick**

```
private: System::Void ballTimer_Tick(System::Object^ sender, System::EventArgs^ e) {
}
};
```

Inside of the `ballTimer_Tick()` handler function, add the following code – you may play around with these values and experiment to see what happens.

```
    private: System::Void ballTimer_Tick(System::Object^ sender, System::EventArgs^ e) {
        ball_x += 30;
        ball_y += 20;
        if (ball_x + BALLSIZE > this->Width)
            ball_x = 0;
        if (ball_y + BALLSIZE > Height)
            ball_y = 0;
        Invalidate(); // Cause a repaint.
    }
```

Build and run your project and make sure you can now see the red ball moving.



Task 3 – Adding a Stop Condition for the "Ball" Animation

Switch back to the design view of the form. For this task, it's up to you to figure out how to follow each of the steps! The steps are as follows:

1. In the Toolbox window, expand the "All Windows Forms" items, find `TrackBar`, and add one to the Form.

2. Change the following properties:

• TickFrequency :	100
• Maximum:	800
• Minimum:	20
• Value:	300
• Name:	timerSpeedTrackBar

3. Add an Event Handler for the "Value Changed" event. We want this to specify how fast the timer goes off. Add the following code:

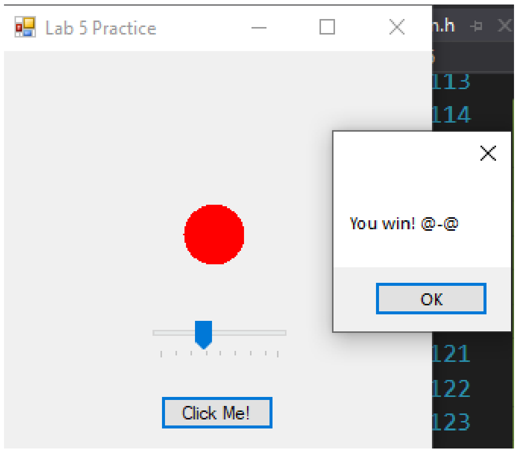
```
private: System::Void timerSpeedTrackBar_ValueChanged(System::Object^ sender, System::EventArgs^ e) {
    ballTimer->Interval = timerSpeedTrackBar->Value;
}
```

4. Finally we'll add code to stop the program when the user clicks on the ball. Select the Form in the forms designer, add an event handler for `MouseDown`, and add code to the mouse handler using the pseudocode:

if the click is on the ball, execute

```
        ballTimer->Enabled = false;
        MessageBox::Show("You win! @-@");
        this->Close();
```

This will turn off the timer (so the ball stops moving), display a message, and then close the application. The "right" way to determine if a mouse click is "on the ball" is to compute the distance between the mouse and the ball's center, noting that (ball_x, ball_y) is the coordinate of the upper-left corner. A simpler way is if the mouse's X coordinate is within BALLSIZE pixels of ball_x and the Y coordinate is within BALLSIZE pixels of ball_y – this method treats the ball as a square. Use whichever solution you prefer. Within the event handler, the mouse's X and Y coordinates are given by e->X and e->Y.



You win! @-@

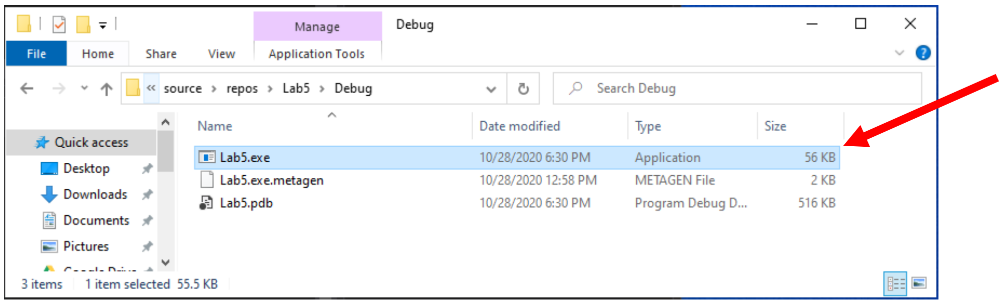
Grading

You will be graded according to the following Rubric:

- (10 points) Does your program work and can I stop the ball from moving?

Submission

When you are done building / testing your program, navigate to the debug directory and locate the executable file for your program. The Debug folder in the root directory for you project is typically where the executable file is located:



Name	Date modified	Type	Size
Lab5.exe	10/28/2020 6:30 PM	Application	56 KB
Lab5.exe.metagen	10/28/2020 12:58 PM	METAGEN File	2 KB
Lab5.pdb	10/28/2020 6:30 PM	Program Debug D...	516 KB

3 items 1 item selected 55.5 KB

You must submit this executable file to Canvas before the due date.