Week 2: Practical

An Introduction to Visual Studio and C#

This practical will introduce you to techniques in designing Visual Studio applications and documenting projects. You will also learn how to graphically draw lines on a form using C#. Read the entire practical before you begin creating your application.

Reading

Before starting this practical, you should have read Chapters 1 and 2 of Gaddis. You should also have read Appendix D: Introduction to Algorithms and the pseudo-code guide at:

`http://users.csc.calpoly.edu/~jdalbey/SWE/pdl_std.html`

Getting Started

In your documents folder create a new folder called **101**. You should save all of your applications for this course in this folder. In your **101** folder create another folder called **Week 1** which will hold your work for the practical for Week 1.

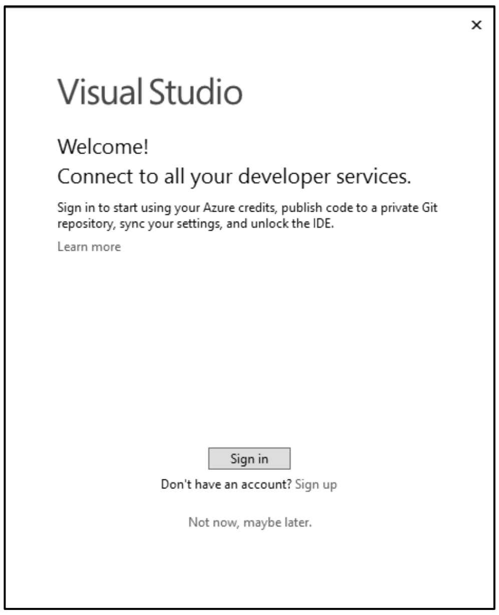
Each practical has set exercises to complete and to get full marks you must complete all exercises to the satisfaction of the demonstrator. Everyone must attempt Exercise 1 of each practical. Many practicals will have options for Exercise 2 and you must complete one of the options. Chose an option that interests you or is relevant to the degree that you are doing. You may complete more than one option but you can only have one option marked. The textbook also has many exercises that you can attempt for more practice. The more practice you do the better you will become at creating applications in C# .NET.

Create a paper prototype for your user interface for Exercise 3 and show this to the demonstrator when you get this practical verified. The user interface should be designed using the interface design practice and guidelines you have seen so far on the course and in the textbook.

Starting Visual Studio

From the Start (Windows) menu, select **Visual Studio 2022** (the program, not the folder).

You may see a window which looks similar to this:



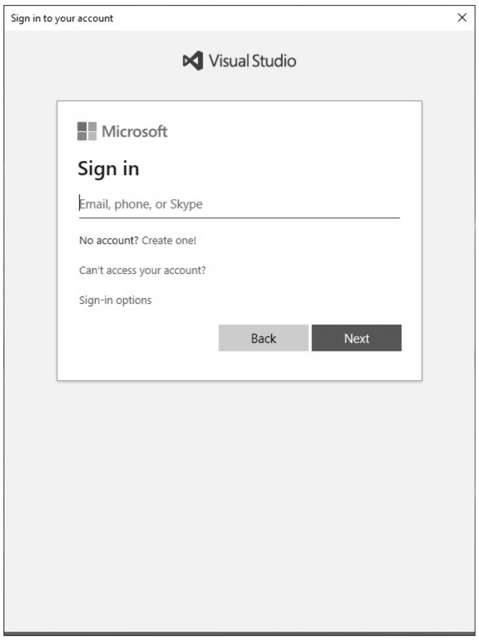
Visual Studio

Welcome!
Connect to all your developer services.

Sign in to start using your Azure credits, publish code to a private Git repository, sync your settings, and unlock the IDE.

You will then need to sign in using your university credentials otherwise Visual Studio will stop working after 30 days. Click the **Sign In** button.

You will then see:

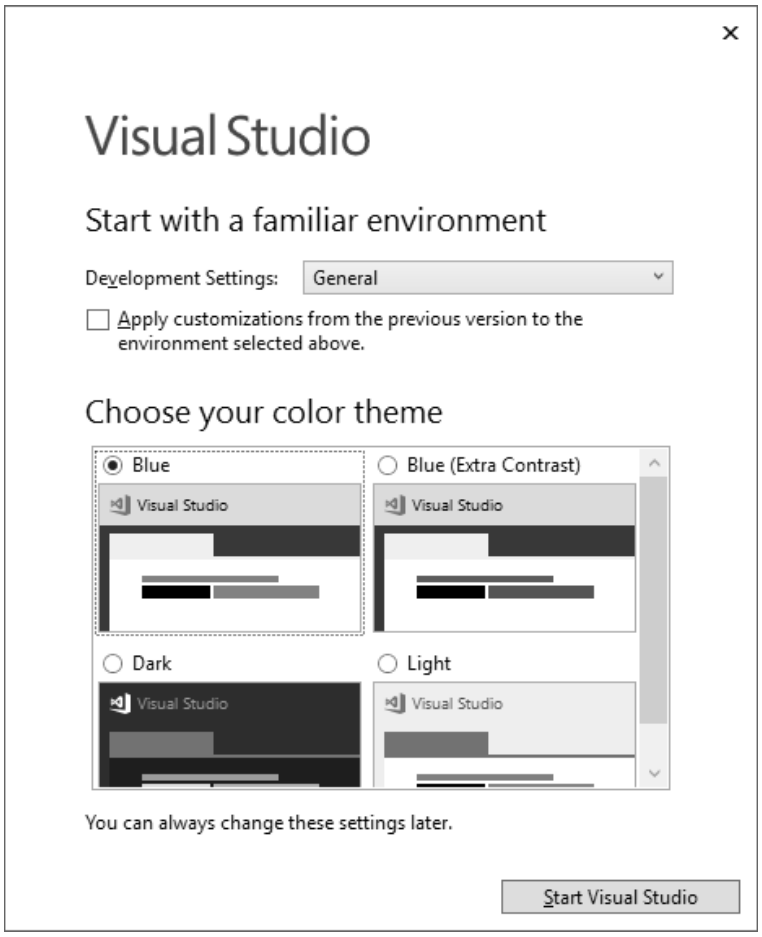


Type in your university email address:

<moodle username>@students.waikato.ac.nz

You will then be taken to a university login page so type in your university password then click ok and it will register Visual Studio to your university account. You can do this if you also install Visual Studio on your own computer.

You may then see the following screen:


Visual Studio

Start with a familiar environment

Development Settings: General

☐ Apply customizations from the previous version to the environment selected above.

Choose your color theme

● Blue ○ Blue (Extra Contrast) ○ Dark ○ Light

You can always change these settings later.

Change the **Development Settings** option to **Visual C#** and then choose a colour theme of your choice and then click on the **Start Visual Studio** button.

Using Visual Studio

It is vital that you watch the Introduction to Visual Studio recording in the Panopto section on Moodle to show you how the lab works and on how to use Visual Studio. Even if you have used Visual Studio before it is highly recommended you watch the video as it explains how it works in our lab.

Exercise 1

In this exercise you will learn how to draw shapes in a picturebox.

a) Create a new C# .NET solution and save it in your **Week 2** folder. Add a picturebox control to the form and give it the name **`pictureBoxDisplay`**. Create a button at the bottom of the form and give it the name **`buttonDrawLine`** and make the text in the button read **`Draw Line`**. Change the title of the form to be **`Drawing Application`**. Also set the **`StartPosition`** property of the form to **`CenterScreen`**. This means that whenever the applications runs the form is always in the centre of the screen. You should do this for all applications you create unless you have a specific reason not to.

Hint: To change the title of the form, change the **`Text`** property of the form in the designer window.

Create a click event procedure for the button (by double clicking on the button in the design window) and inside the curly braces type in the following code:

```
Graphics paper = pictureBoxDisplay.CreateGraphics();
Pen pen1 = new Pen(Color.Blue, 5);

paper.DrawLine(pen1, 10, 10, 100, 100);
```

Now run the application and when you click on the button you should see a line appear in the picturebox. Try changing the thickness of the pen with different colours. Can you draw multiple lines in the picturebox? Try doing this. Can you create pens of different colours and draw lines with them? Explore, have fun!

b) Create an **`Exit`** button next to the **`Draw Line`** button which closes the application. Give the button an appropriate name.

The statement to exit from the application is:

```
Application.Exit();
```

c) Create a **`Draw Square`** button and draw a square with each side of the square drawn in a different colour.

Note: If the form is minimised or another window is placed on top of the form then the drawing will disappear. Do not worry about trying to fix this, as it is not important. When getting the exercise verified just keep the drawing window open, or click the Draw Line button again.

Exercise 2

a) Create a new C# .NET solution and name the project **`MouseMoveGraphics`** and save it in your **Week 2** folder. Make the form reasonably large. Add a picturebox control to the form and give it the name **`pictureBoxDisplay`**. Make the picturebox fill up all of the form.

b) You will now need to create a custom event for the picturebox. Follow the instructions below and if you have problems then ask the demonstrator for help.

Click on the picturebox and in the **`Properties`** window click on the *lightning bolt*. This will switch the window to show the events you can associate with a control instead of the properties. Scroll down until you find the **`MouseMove`** event and double click on it. It will then generate the MouseMove event handler for the picturebox. This is how you can associate different events to controls from the default ones which are generated when you double click on a control.

The **`MouseMove`** event is triggered whenever you move the mouse pointer while it is over the picturebox. What you will be doing is drawing lines as you move the mouse.

c) In the **`MouseMove`** event, create a Graphics object connected to the picturebox. Also create a pen called **pen1** and give it a colour and a size of 2.

What you need to do is draw a line from position 0,0 to where the mouse pointer is. The **`e`** parameter variable stores the x and y position of the mouse inside its properties. Type in the following:

```
paper.DrawLine(pen1, 0, 0, e.X, e.Y);
```

This draws a line from 0,0 to the x and y position of the mouse. Run the application and see what happens when you move the mouse pointer around the picturebox. What happens if you move the mouse pointer outside the picturebox?

d) Change the code so that it draws from the centre of the picturebox to the mouse pointer location. Then run the application.

Exercise 3

Create a new C# .NET solution and save it in your **Week 2** folder.

For this exercise, you only need to create the user interface. You do not need to make the application work. You will do that in Week 3's practical.

Nilesh has started a new business, "Hard As Concrete", which lays concrete driveways for new homes. Nilesh remembered you were a genius programmer and wants you to build an application which allows him to type in the length (e.g. 5.3) and width (e.g. 2.5) of the driveway in metres. It will then calculate and display (in separate textboxes) the volume of concrete required (the depth of every driveway is always 0.5m), the number of kilograms of cement required, the number of bags of concrete required and the total cost of the cement. One kg of cement will make 1.5m^3 of concrete and the cost of a bag of 2kg cement is $15.50.

Your application must have **3** buttons: **`Calculate Concrete Cost`**, **`Clear`** and **`Exit`**.

The **`Clear`** button will clear the input text boxes and all output textboxes and set the focus to the first input text box. The **`Exit`** button will end the application.

a) Produce a paper prototype of the user interface. Think about what controls you will need and the layout of the controls on the form. Think carefully which controls should be read-only. Get your demonstrator to check the paper prototype before continuing.

b) Now build the user interface and add your name and id number as comments at the top of your code. You don't need to write any code yet, you will finish this application in Week 3.

Paper Prototype:

Week 2 Practical: REVIEW PAGE Name: ____________________

Questions: Complete the following questions. (Note: this must be done before you seek a verification).

1. What is the purpose of paper prototyping the user interface of an application?

2. In exercise 2, what happens if you move the mouse pointer outside the picturebox?

3. If you wanted to display the cost of an order that is calculated by an application, which control would you use?

4. What is an object? List the objects you used in exercise 1.

Verification: To assess your competency of this material your demonstrator will verify that you have:

1. Drawn the paper prototype for exercise 2.
2. Answered the set questions.
3. Created the required applications.
4. Used appropriate names for your controls.

Week 3: Practical

Using Objects, Variables and Constants

This practical will introduce you to using variables and objects in C# to store values while an application is running. You will also learn about the scope of variables and how to create and use constants. Read the entire practical before you begin creating your application.

Reading

Before starting this practical, you should have read Chapter 3 of Gaddis.

You are not required to produce a paper prototype for this practical.

Exercise 1

a) Create a folder called **Week 3** in your **101** folder. Copy the **`MouseMoveGraphics`** project from Week 2 into your Week 3 folder and then open the project.

b) Add a **`Set Colour`** button to your form. Now add the **`ColorDialog`** control to your project from the **`Dialogs`** section in the toolbox and change the name to **`colorDialog1`**. This control allows the user to visually select a colour which is stored in the **`Color`** property of the dialog control. In the click event method of the **`Set Colour`** button type in the following to display the dialog window to the user:

```
colorDialog1.ShowDialog();
```

c) Modify the **`MouseMove`** event method so that the lines are drawn in the colour chosen in the dialog control. Run the application and try changing the colour several times.

Hint: When creating the pen, remove the colour value and replace it with colorDialog1.Color.

Exercise 2

a) Create a folder called **Week 3** in your **101** folder. Create a new C# .NET solution and save it in your **Week 3** folder.

Add a **`Draw Triangle`** and **`Draw Square`** button to the form.

In the click event method of the **`Draw Triangle`** button ***declare three point objects, corner1, corner2 and corner3*** with different coordinates. Using the **`DrawLine`** method draw lines between the corners to form a triangle on the form.

b) Create an **`Erase`** button that will erase any graphics in the picturebox. In the click event procedure of the button type the following:

```
pictureBoxDisplay.Refresh();
```

Note: This command will only clear the graphics that have been drawn on the form. It does not clear the contents of controls like text boxes and labels.

c) In the click event method of the **`Draw Square`** button, create an x variable set to 50, a y variable set to 80 and a size variable set to 60. Create a graphics object connected to the picturebox and a pen object and then use the **`DrawRectangle`** method to draw a rectangle using the variables you declared previously. Use the size variable for the width and height of the rectangle.

Exercise 3

a) Watch the recording on using the Visual Studio debugger tool that is posted in the Week 3 section on Moodle.

b) Copy your project folder for **Week 2 Exercise 3** into your **Week 3** folder and then open it in Visual Studio. You will now complete the code for the application.

Add the following constants:

```
//The depth of every driveway.
const double DRIVEWAY_DEPTH = 0.5;
//Amount of concrete created from 1 kg of cement
const double CONCRETE_PER_KG = 1.5;
//Weight of a bag of cement
const double BAG_WEIGHT = 2.0;
//Cost of a bag of cement
const decimal BAG_COST = 15.5m;
```

c) Write the code for the **`Clear`** button which will clear the input text boxes and all output values and set the focus to the first text box. Write the code for the **`Exit`** button.

d) Write the code for the **`Calculate Concrete Cost`** button using the pseudo-code given below (also use a Try..Catch structure in the code to deal with errors):

```
Declare Variables
TRY
   GET the length of the driveway
   GET the width of the driveway
   CALCULATE the volume of concrete required
     ( = length * width * depth)
   CALCULATE the number of kilograms of cement required
     ( = concrete volume / concrete from 1 kg of cement)
   CALCULATE the number of bags of cement (rounded up)
     ( = kilograms required / weight of a bag of cement
   CALCULATE the total cost of the cement
     ( = number of bags of cement * cost per bag)
   DISPLAY the volume of concrete required (to 3dp)
   DISPLAY the number of kilograms of cement required (to 3dp)
   Display the number of bags of cement required
   DISPLAY the total cost of the bags formatted as currency
CATCH
   Display error message
   Clear all textboxes
   Set focus to first input textbox
ENDTRY
```

Hint: *int x = (int)Math.Ceiling(y) gives the value of y rounded up to the nearest whole number.*

Week 3 Practical: REVIEW PAGE Name: ____________________

Questions: Complete the following questions. (Note: this must be done before you seek a verification).

1. What is a variable and what is it used for?

2. How do you create a variable that can be accessed by any method in the form's class? Where should this variable be declared?

3. What is a **constant** and why would you use one?

4. What does the Parse method do? Give an example of how to use it.

Verification: To assess your competency of this material your demonstrator will verify that you have:

1. Created the required applications.
2. Answered the set questions.
3. Can use the debugger to set a breakpoint, step through the code and display values of variables using the mouse.