Basic C# Game Guide

Rene Lisasi

Kennesaw State University

TCOM2010-W02

Table of Contents

[How to build your first game in Visual Studio using C# 2](#_Toc107266791)

[Introduction: 2](#_Toc107266792)

[Equipment and Supplies 2](#_Toc107266793)

[Prerequisite: Installing Visual Studio 2](#_Toc107266794)

[Setting up our environment 2](#_Toc107266795)

[Coding Essentials 3](#_Toc107266796)

[Data Types 3](#_Toc107266797)

[Statements 3](#_Toc107266798)

[Return types 3](#_Toc107266799)

[Accessibility Modifiers 4](#_Toc107266800)

[Call 4](#_Toc107266801)

[Conditional Statements 4](#_Toc107266802)

[Objects(Classes) 4](#_Toc107266803)

[Properties 4](#_Toc107266804)

[Methods 4](#_Toc107266805)

[Arguments 4](#_Toc107266806)

[Planning 5](#_Toc107266807)

[Logical Steps/ Ideas 5](#_Toc107266808)

[Flowchart 5](#_Toc107266809)

[Decisions Flowchart 5](#_Toc107266810)

[Coding Instructions 6](#_Toc107266811)

[GUI 7](#_Toc107266812)

[Events 8](#_Toc107266813)

[Testing 8](#_Toc107266814)

# How to build your first game in Visual Studio using C#

## Introduction:

This document will teach you how to code a simple game. The audience is expected to have no prior knowledge to computer programming and will cover the absolute bare minimum you need to code anything without going into detail. However, you must be computer literate. If you follow these instructions, you will gain a basic understanding of how basic desktop applications are designed. Hopefully this will inspire you to code some of your own programs using this knowledge.

## Equipment and Supplies

For this project you will need:

* A working computer
* Windows 10 or above (This guide is intended for windows users, however, if you are using another platform simply follow the coding guide)
* A web browser to download visual studio

## Prerequisite: Installing Visual Studio

Visual studio is your IDE and compiler. An IDE is an application that organizes and allows you to directly interact with code using a GUI. A compiler translates the code into assembly language. Having these integrated allows you the programmer to focus on writing the code. With that said let’s begin.

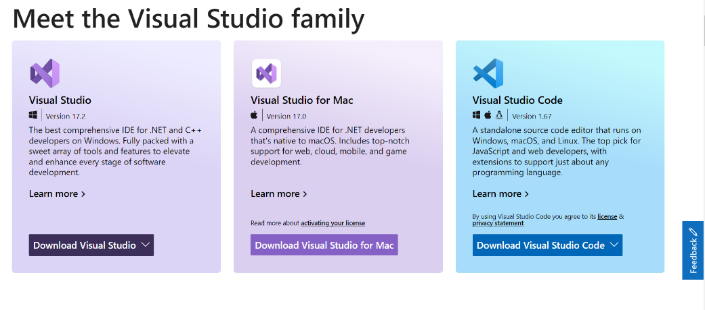
1. Visit Microsoft’s website by clicking this link <https://visualstudio.microsoft.com/>
2. Scroll down to find the Community edition.  
   

Figure 1.1 Visual Studio editions

1. Download the installer
2. Save to your Downloads folder  
   Graphical user interface

   Description automatically generated

Figure 2 Downloads Folder

1. Run the installer  
   Graphical user interface, website

   Description automatically generated

Figure 3 Workload options

1. Do not be intimidated by all these options simply click install
2. Once the download is complete you may begin

# Setting up our environment

Now that you have Visual Studio open, we’re going to set up our environment. Be sure to have deep house music electronic music playing in the background as this helps you get into the groove.

1. Choose .NET as your language
2. Wait for it to install
3. Once it is done, click Create a new project  
   A screenshot of a computer

   Description automatically generated with medium confidence

Figure 4 Visual Studio welcome screen

1. Scroll to find Windows Forms App, select it and click next.  
   Text

   Description automatically generated

Figure 5 Visual Studio project screen

1. Name the project HighLowGame and leave the default location and click next  
   *In this exercise we will use the Java naming convention which consist of capitalized first letters for each word and no spaces. If you must use a space the character for it is \_ or underscore.*Graphical user interface, application, website

   Description automatically generated

Figure 6 Directory

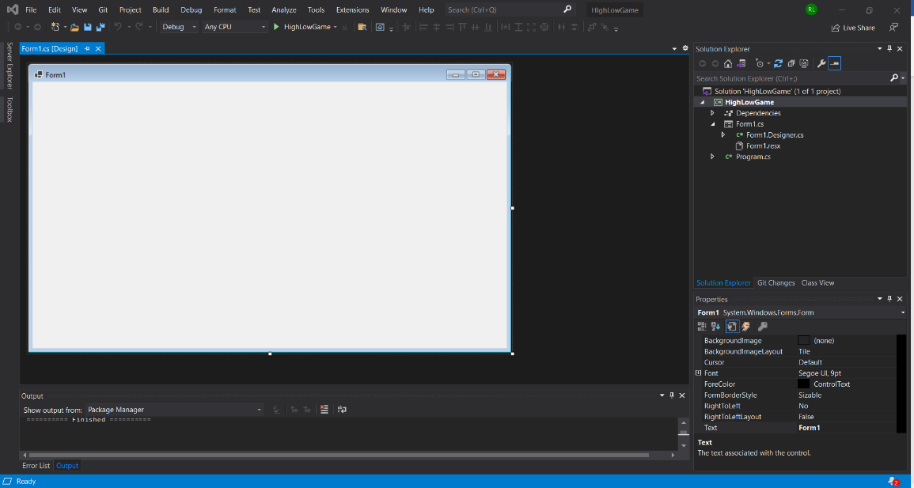
1. Use the default setting for target framework and click create. Welcome to your coding environment!  
   

Figure 7 gui designer

# Coding Essentials

Now that your environment is set up let’s talk a little about coding itself. There are some things you must know about before we dive in.

There are many programming languages out there just like there are many recipes to make the same dish. In this case our dish is a high or low guessing game but we’re following the C# recipe. There are similar recipes to make a beef stew each with slight variations but what matters is that the beef stew tastes good, this is the equivalence of having a program that runs without errors. So, it does not matter which language you use to build a program but that you solve problems and have an error free program. With that said let’s dive into some common programming details (these apply to most recipes):

## Data Types

In programming we have 8 common data types, most of which are primitive (basic):   
*Where and how you use them depends on the situation sometimes, some data types will be more suited to a situation than others.*

* Strings
* Integers
* Booleans
* Floats
* Longs
* Bytes
* Shorts
* Doubles
* Chars

## Statements

A statement is line(s) of code

## Return types

Return types indicate what data type the compiler should expect from a statement.

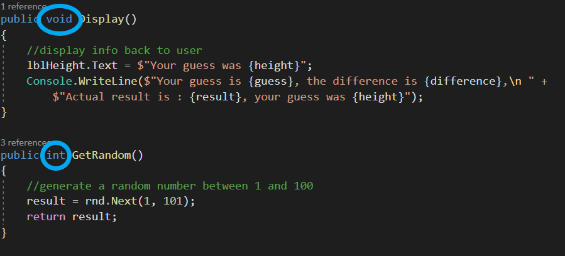


Figure 8 return types

## Accessibility Modifiers

Accessibility modifiers indicate whether or not an entity can be modified outside of its scope (scope is its local environment, most likely a class or object which we will get into next)

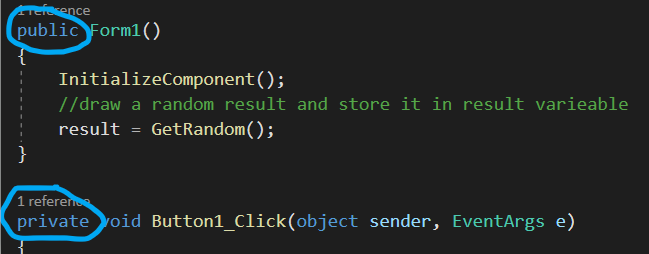


Figure 9 accessibility modifiers

## Call

A call is a reference. Calls usually have open parentheses and can take arguments.

## Conditional Statements

Conditional statements are lines of code that are compiled when a condition is met. They usually use the if, else syntax.

## Objects(Classes)

Objects are entities that hold other entities and can give those entities functions called methods.

### Properties

These are the building blocks of an object; they can be data types or other objects.

Text

Description automatically generated

Figure 10 Properties

### Methods

These are function definitions. They define what properties do, have accessibility modifiers, and have return types. Once defined they can be called.

Graphical user interface

Description automatically generated

Figure 11 Methods

### Arguments

Arguments are what values to pass to a method call. They go in the parentheses, are separated by commas, and if a method is defined with arguments, then having an inappropriate data type or missing argument will result in an exception (error).

#### Constructors

Constructors are methods that define an object. They can be divides into two kinds:

##### Empty Constructors

Empty constructors define an object in its most basic form.

##### Loaded Constructors

Loaded constructors define objects with pre-existing changes.

#### Event Handlers

Event handlers are methods that handle interactions. They receive instructions and call the appropriate methods.

Now that you have a basic idea of what our building blocks are, let’s put ourselves in the player’s shoes and plan how the application will work.

# Planning

This part of building the program requires us to plan how the user will use the app. You can use any planning software you like or use a pen and paper; the choice is yours. Here are the steps:

## Logical Steps/ Ideas

* The app should randomly generate a number
* The user should guess the number by typing it in
* The user should click the guess button
* The app should interpret the user’s input and compare it to the randomly generated number
* If the user is too high, it should indicate that it’s too high.
* If the user is a little high, it should indicate that it’s little high.
* If the user is close, it should indicate that it’s close.
* If the user guesses correctly, it should indicate that they won and offer the user a new game.
* If the user is a little low, it should indicate that it’s a little low.
* If the user is too low, it should indicate that it’s too low.

## Flowchart

Next you want to organize your ideas into a flowchart the computer will follow in decision making.

### Decisions Flowchart

Figure flowchart between decisions

*Where there are conditions, conditional statements must be used.*

Now that you have completed your planning, we may begin coding!

# Coding Instructions

Now that you know the fundamental characteristics that make up a computer program , and you have planned how the user will interact with the application, you are ready to put it all together. Here are the steps to code a guessing game:

## Code

1. Double click the form, it will bring you to the Form1.cs class
2. Code in your properties below the public partial class Form1: Form curly bracket (line 15)  
   Text

   Description automatically generated

Figure 13 Code properties

1. Inside public Form1(){, under initialize component, code in result= GetRandom();
2. Create a new method called GetRandom, it should be public and have an integer return type.  
   Text

   Description automatically generated

Figure 14 getRandom() method

1. Create a new method called GetGuess, it should be public and have an integer return type.  
   Text

   Description automatically generated

Figure 15getGuess() method

1. Create a new method called SetDifference, it should be public and have a void return type.  
   Text

   Description automatically generated

Figure 16 setDifference method

1. Create a new method called SetProgBar, it should be private and have a void return type. Inside, type: progressBar.Value=prog;
2. Create a new method called SetNull, it should be public and have a void return type. Inside, type:  
   //reset

guess = 0;

result = 0;

difference = 0;

height = null;

attempts = 0;

prog = 50;

userInput.Text = 0.ToString();

lbl\_Attempts.Text = 0.ToString();

progressBar.Value = 0;

1. Create a new method called SetStrings, it should be public and have a void return type.  
   a. Add multiple if statements for each outcome ( too low, too high, close, low, high, and right)  
   b. Add a DialogResult and equate it to MessageBox.Show.  
   c. Add an if else statement to your method.  
   Text

   Description automatically generated

Figure 17 setStrings method

1. Create a new method called Display, it should be public and have a void return type. Inside, type:   
   //display info back to user  
   lbl\_Height.Text = $"Your guess was {height}";

Console.WriteLine($"Your guess is {guess}, the difference is {difference},\n " +$"Actual result is : {result}, your guess was {height}");

1. Create a new method called PlayGame, it should be public and have a void return type. It will call all the above-mentioned methods.  
   //call all methods

GetGuess();

SetDifference();

SetStrings();

SetProgBar();

Display();

If you cannot figure out how to code this, the full code is here for reference. Complete the data entry yourself if you want to learn how it works. Otherwise simply copy and paste the code into your Form1.cs document.



Figure 18: Form1.cs Code

This is it for now when it comes to coding. Our code is expecting graphical controls, so we need to design some GUI to get rid of the red underlines.

## GUI

1. Enter the Form1.cs designer by clicking (Form1.cs[Design]) tab  
   A screenshot of a computer

   Description automatically generated with medium confidence

Figure Begin gui

1. From your toolbox double-click “NumericUpDown”.   
   Graphical user interface, application

   Description automatically generated

Figure numeric up down

1. Click the number box to highlight it and from your properties tab click font and change the size to 100pt.  
   Diagram

   Description automatically generated

Figure bigger numeric up down

1. Under design from the properties tab, change the name for this “NumericUpDown” control to “userInput”  
   A screenshot of a computer

   Description automatically generated with medium confidence

Figure user input

1. Under properties tab, under the data sub tab, set the maximum value to 100  
   A screenshot of a computer

   Description automatically generated with medium confidence

Figure maximum value

1. Add a progress bar from your toolbox, like in step 2.
2. From the properties tab, set the progress bar value to 50 and keep the step 10.  
   A screenshot of a computer

   Description automatically generated with medium confidence

Figure progress bar properties

1. Add two buttons from your toolbox and place them below the “NumericUpDown”.
2. From the properties tab, change the *text* to “Guess” for one button, and to “Restart” for the other.
3. Under design from the properties tab for each button change their names to btnRestart and “btnGuess” accordingly.  
   Graphical user interface, application

   Description automatically generated

Figure button from toolbox

1. Add two labels from toolbox and name them lbl\_Height and lbl\_Title.
2. Change the *text* for lbl\_Height to “Guess the temperature”.  
   Diagram

   Description automatically generated

Figure labels example

1. Add two more labels and name them “lbl\_Attempts” and “lbl\_Result” and set their values to 0.
2. Add two more labels, don’t name them, just change their text to attempts and previous result.  
   Graphical user interface, diagram

   Description automatically generated

Figure more labels

1. Double click the guess button. It will take you to the Form1.cs editor.  
   Text

   Description automatically generated

Figure Editor after double click

1. Go back to the design editor for Form1 (Form1.cs[Design]) and double click the restart button.  
   *Double clicking a button automatically creates an event handler method and takes you to the editor.*

Graphical user interface, application

Description automatically generatedA hand holding a white device

Description automatically generated with low confidenceYour form should look like the image on the left, but with more practice can look like mine on the right:

Figure How the form should look

Figure how the form could look with images

## 

## Events

The user interface is now complete. You will now put actions behind each of your controls and allow the user to play the game.

1. Inside your btn\_Guess\_Click method call the PlayGame method be sure to end your statement with a semicolon. This tells the compiler that the statement is done.
2. Increase the number of attempts by one.
3. Equate the lbl\_Attempts to attempts property (properties are also known as variables).  
   Text

   Description automatically generated

Figure Guess event handler

1. Inside your btn\_Restart\_Click method equate the lbl\_Result to the result variable.
2. Call the setNull method
3. Call the GetRandom method  
   Text

   Description automatically generated

Figure Restart event handler

1. Inside the public Form1 method, under initialize component method call, make sure to equate your result variable to the GetRandom method call.  
     
   Here’s the final version of what your editor should have.  
   

That’s it for events. Now to test the game!

# Testing

Testing includes black box and white box testing. White box testing focuses on testing units of your code, while black box testing focuses on ensuring that the user can play the game from start to finish without errors. Normally when building an app, you would white box test as you go, but I was kind enough to do that for you, so you can do all the fun stuff.

1. Click the green play button at the top or press F5 on your keyboard to run.
2. Make sure the computer is generating a random number (not the default 0).  
   Graphical user interface

   Description automatically generated with medium confidence

Figure testing the form on low

1. Make sure you are able to guess numbers.
2. Make sure the progress bar is moving when you guess a number.
3. Make sure that when you interact with an event, that you get feedback from the labels.
4. If all of these tests pass the game will pass the white box test.
5. If the app crashes at any point, make sure you did not make any typos and have semicolons after each statement.
6. Make sure that you used the correct syntax (using google).
7. If there are any exceptions, Visual Studio will point them out, search the exception name to find the solution or wrap the exception causing statement in a try catch block (advanced).

This should be your screen when testing, and if you are right, this message should appear:

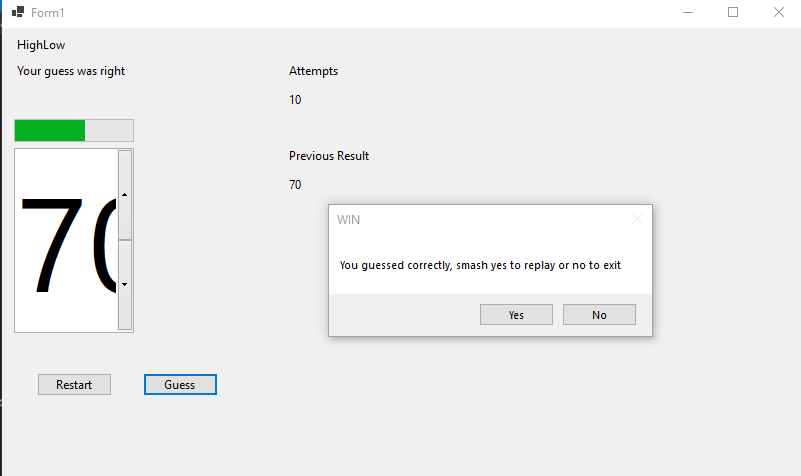


Figure finished product

Congratulations!

You have now built your first ever application and I am sure you will be ready to build more. Keep in mind that the problem solving was completed for you in this example, but it should serve as a guide for you to build more. Good luck and revisit frequently and try to produce your own game and build it using what you learned here.