

---

HONG KONG INSTITUTE OF VOCATIONAL EDUCATION**Laboratory 1: First Taste of Coding Game**

---

**Module Intended Learning Outcome (#2):**

On completion of the module, students are expected to be able to:

- develop 2D and 3D graphics programs for general gaming purposes;

**Lesson Intended Learning Outcome:**

On completion of this lab, students are expected to be able to:

- Create a simple program for Windows using MonoGame and Visual Studio.
- Compile, debug and execute a game program in Visual Studio.

**First MonoGame Program – A Blank Project**

1. Install the Visual Studio 2017 Community and MonoGame 3.7 in your computer if the software has not been installed. The files can be downloaded from Microsoft Visual Studio homepage (<http://www.visualstudio.com/>) and Monogame homepage (<http://www.monogame.net/>).
2. Create a new Project by choosing **Window Project** icon under Visual C# > MonoGame tab in the New Project dialog. Remember to give your project a name and set an appropriate path for your project.
3. By default, the Visual Studio will generate a *Game1.cs* file, a *Program.cs* file, a *Game.ico* file and some other files. You can add files to / remove files from the project in the Solution Explorer by right-clicking the project name.
4. Compile and run the project. A basic window with nothing but the title bar will appear.
5. Familiar with the following debugging tools to help you fix your errors in the future:

(1) Error List

Located at the bottom of the project page. (If not found, open it through the View menu). If your code fails to compile, the Error List will show error messages alerting you to reasons why. The list will show a warning message if the compiler finds an issue that is not causing a build error.

(2) Breakpoints

Breakpoints allow you to pause on a specific line of code while your program is running. The program will run until the line where your breakpoint is set. You can then view the variable values by mouse over the variables. A breakpoint can be set by left-clicking on the gray margin besides your coding. A red dot will appear to show the location of your breakpoint.

(3) Stepping

When the breakpoint is reached, you can step through the code line by line to watch how your code is executed in run time. **Step Over (F10)** will not enter a new method from a call statement and is useful if you are sure the method works properly. **Step Into (F11)** follow call statements into the methods that are being called, even the method is given by XNA Framework.

(4) Watch Lists


A watch list tracks variables that are declared within your program. When you are debugging the program, a watch list can help you to simultaneously track multiple variables that may exist in different sections of your code. To add a variable to your watch list, you can either type in the variable name directly or right-click a variable and select **Add Watch** during debugging process.

**TASK 1**

Study *Game1.cs* generated by Visual Studio. Name the **SIX functions** in the program (including the constructor). Try finding them in *PlatformerGame.cs* in the Platformer project. Try guessing the running sequence of these functions. You may use the debug tools and ***Console.WriteLine()*** function to help you.

**Convert an XNA Project to MonoGame Project – Platformer Project**

Though Xbox Live Indie Game portal / AppHub had been closed due to its end of supporting period. SimonDarksideJ (<https://github.com/SimonDarksideJ/XNAGameStudio>) had backed up the examples and tutorials of XNA Games originally provided in the site. We can easily convert them into MonoGame games. This time, we are going to use a starter kit called **Platformer** as an example.

1. Choose “Platformer” from the description listed out in Simon’s site. It will lead you to the Platformer dedicated page.  
(<https://github.com/SimonDarksideJ/XNAGameStudio/wiki/Platformer>). Download the zip file (at the bottom of the page) and extract it to any destination folder.
2. Create a new MonoGame project from Visual Studio. Remove and delete the given Program.cs and Game1.cs. We will replace the files using those inside the Platformer project.
3. Copy the code files from the Platformer XNA project to the MonoGame project. Open the Window Explorer and subfolder “Platformer/Platformer” under the unzipped file. Select all \*.cs files and drag them to the MonoGame project. The files will be copied to the MonoGame project automatically.
4. Next, copy the media contents to your MonoGame project. In the original Platformer project, the files for the images, sounds, font and etc. are stored in different subfolders under “Platformer/Content” folder. Select all these folders (except the “obj” and “bin” ones) and copy them to “Content” folder under your MonoGame project using Window Explorer (not directly drag them into the Visual Studio interface).
5. Also copy the “Content.contentproj” file to the same folder as above.
6. Now, you can double-click and open the Content Pipeline Tool .
7. Select “File” > “Import” from the menu bar to import the “Content.contentproj” file you just copied.
8. Save the Content project by selecting “File” > “Save” from the menu.
9. Build the Content project by selecting “Build” > “Build” from the menu.
10. You can now quit the Content Pipeline tool. Although the Visual Studio may contain many files and objects you are not familiar with, you can simply build the program and try to execute it using the start debugging button (▶).

**TASK 2**

Locate the *PlatformerGame.cs* in the Platformer project. This is the main program of the whole platformer game. Find the constant variable *numberOfLevel*. Currently, the game has three individual levels which will be reused after a round. Change the number to 4 and see what happens.

Probably, you will need to add some other files. The files are included into the project through different sub-folders in the “Content” folder. Try to find the “Level” subfolder. Referring to other text file in the folder, create your own level file and store it as “3.txt” in this folder. Remember to set the properties of this file as other text files in the folder (e.g. Build Action should be **Copy**). And you need to include the files inside the project.

The Content project should still have some problems. It cannot load the background pictures. In your Window Explorer, locate the “Background” subfolder. Copy and rename some of the pictures. Each level has 3 layers of background. The first number of the file name denotes the layer number and the second number is the level number.

After manipulating the files in your Window Explorer, remember to include the files in the Content project. Save and Rebuild your project.

## Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder