## Homework

For all your programs use the code from F**igure 2.10** (2nd ed) or Figure 2.11(3rd ed) as your base, and make necessary changes for each program.

**1**.      Download the **full directory** for F**igure 2.10** (2nd ed)/Figure 2.11(3rd ed) located on canvas in cs440/files/ SampleCodeFig10-11/fig2.10-11-2018.zip

(<https://cwu.instructure.com/courses/50755/files/folder/SampleCodeFig10-11>)

**2.** Ensure that you use **Visual Studio 2012** (VC2012 is installed on all lab computers). This will make your program compatibility with environment that TA will use to grade your program.

**3.** Unzip fig2.10-11-2018.

**4.** Click on solution file: **fig2-10.sln** to open fig2.10-11-2018 in VC2012.

**5.** Rebuild solution and run the program.

Note: The code fig2.10 in Hill’s book 2nd ed has a line of code #include<gl/glut.h> that was augmented in the directory fig2.10-11-2018 by #include “glut.h” with glut.h located in the this directory.

**6.** Make a **full copy** of the directory fig2.10-11-2018 for each your program. It is very important to ensure that your C++ solution and project will be compatible with OpenGL with all libraries and DLLs located in the right places.

**7.      Rename** your copy of the directory to Program1, Program2 and so on.

**8.** Click on solution file: **fig2-10.sln** to open your program fig2.10-11-2018 in VC2012.

**9.      Edit fig2-10.CPP in VS2012 after you opened it in fig2-10.sln.**

Do not work with fig2-10.CPP and CPP of any other your programs on VC2012 outside of the solution fig2-10.sln. This was a common error in the past.

## *Each your program must start from the statement that you wrote this program yourself using the code from the textbook. It must be a comment line of the code. Plagiarism will not be tolerated and CWU rules for such plagiarism cases will be applied.*

|  |  |  |
| --- | --- | --- |
| # | Due | Assignment summary |
| Homework 1 | Oct 4 | Implement   fig2-14, fig 2-16, 2-17a, 2-18, 2-20, 2-24, 2-25, 2-27, 2-28 |
| Homework 2 | Oct 18 | Implement   example 2.4.1, figure 2.38, example 2.4.3, figure 2.40  example 2.4.5, figure 2.41 (pp.63-67) |
| Homework 3 | Nov 1 | Implement   figures 3.7, 3.8, 3.13, 3.17, 3.25-3.29, 3.32, 3.57  3.72, 3.85, 3.86, pp. 134-135 |
| Homework 4 | Nov 15 | Implement   Figure 4.22, 4.48, 4.49 |

## More detail on homework

**Homework 1**

Program 1: Using **figure 2.14**, create the Sierpinski gasket on screen

Program 2: Type in the listing in **figure 2.16** and run.

Program 3: Create a program that generates the thin lines image in **figure 2.17**.

Program 4: Create a program that generates both the polyline and the polygon in **figure 2.18**.

Program 5: Using **figure 2.20**, Create a program that will plot a function using the screen as a line graph. You can use any function you wish to illustrate this.

Program 6: Using **figure 2.24**, create a program that draws a house with fixed dimensions.

Program 7: Using **figure 2.25**, create a program that draws a house with parameterized dimensions.

Program 8: Using **figure 2.27** and **figure 2.28**, create a program that draws a series of polylines. It should use a class to store the polyline data. You may generate any image you wish to illustrate this.

**Homework 2**

Program 1: Using **example 2.4.1**, create a program that allows the user to plot dots on the screen using the mouse.

Program 2: Using **figure 2.38**, create a program that allows the user to draw rectangles on the screen using the mouse.

Program 3: Using **example 2.4.3** and **figure 2.14**, create a program that allows the user to draw a Sierpinski gasket with the mouse by setting the three points to generate the image.

Program 4: Using **figure 2.40**, create a program that draws polylines on screen using the mouse.

Program 5: Using **example 2.4.5**, create a program that allows freehand paintbrush-style drawing on screen using the mouse.

Program 6: Using **figure 2.41**, create a program tht uses the mouse and keyboard to allow the user to draw dots on the screen by pressing p. The E key should exit the program. If you wish, you can implment the point storage in the listing to achieve an elaborate effect of some kind.

**Homework 3**

Program 1: Using **figure 3.7** and **figure 3.8**, create a program that plots the sine function, using parameterized window and viewport settings.

Program 2: Using **figure 3.13**, create an zoom animation. You may want to use the double-buffering technique described to create a smooth effect.

Program 3: Using **figure 3.17**, create an image that then uses the reshape function so that if the user resizes the window the image alters to keep the same aspect ratio it had.

Program 4: Using **figure 3.25** through **figure 3.29**, implement the Canvas class. Create a simple image to illustrate the use of the Canvas class.

Program 5: Using **figure 3.32** and the Canvas class, createa progrm that draws arrows on the screen with relative positioning.

Program 6: Using **figure 3.57** and the Canvas class, create a program that draws the arc of a circle on the screen.

Program 7: Using **figure 3.72**, create a program that draws an ellipse on the screen.

**Homework 4**

Program 1: Using **figure 4.22** create an animated effect of twining, You can use the Canvas class and the image of the house shape changing to a T, or anything else you wish.

Program 2: Using **figures 4.48, 4.49** implement Cyrus-Beck clipping algorithm for **figure 4.46.**

Program 2 will be graded as other programs and **bonus** will be given for the complete solution:

## 