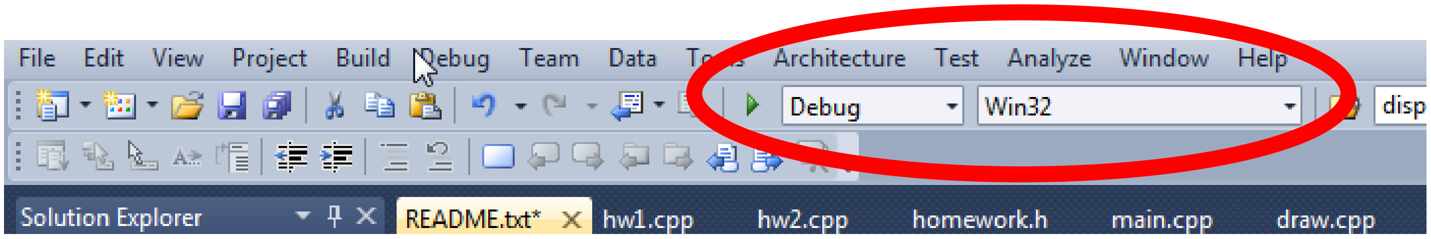
CS 4731 Homework 1 Robert Dabrowski

HOW TO COMPILE AND RUN

1. Double click on the "HW1" icon to load in visual studio 2010
2. Ensure that you have the settings in circled in red below
3. Press "F5" to build and run



Debug
Win32

Usage

- **key m:** Clear screen and draw the mipmap polyline pattern (iteration 1).
- **key p:** Clear screen and draw the mipmap polyline pattern one iteration more (i.e. if you were at iteration 1, redraw the polyline at iteration 2)
- **key l:** Clear screen and draw the mipmap polyline pattern one iteration less (i.e. if you were at iteration 2, redraw the polyline at iteration 1)
- **Key f:** Clear screen and draw the Fern
- **Key s:** Draw Sierpinski gasket
- **Keys q, w, e 'esc'** close program

Program Structure

1. Generic initialization functions are called
2. HW1 specific keyboard and display functions are registered as call backs
3. Glut main loop

Call back functions

They each do the same work: Assemble a "MyPicture" which is made of "MyPolyline"'s and then sends those points to the GPU along with colors and transformation information so that the polylines maintain aspect ratio and are placed properly.

Files

- **utils.h** – Holds useful functions, macros and struct definitions that will hopefully be useful in later homeworks as well as this one
- **homework.h** – Holds function headers useful for only one homework at a time
- **textfile.cpp** – Given functions for reading ANY text file
- **init.cpp** – generalized initialization functions
- **MyPolyline.cpp** – functions relating to the creation and usage of the MyPolyline struct – which holds the number of points in the polyline as well as a pointer to the array of points in the line
- **MyPicture.cpp -** functions relating to the creation and usage of MyPicture structs, which holds the number of polylines in the picture as well as the array of polylines
- **geometries.cpp** – functions to generate single polyline shapes
- **keys.cpp –** functions used with the keyboard call backs
- **color.cpp –** functions related to coloring
- **frame.cpp –** Functions related to the “Frame” abstraction. A frame is defined as the four qualities used to define the world frame: Left, Right, Bottom, Top. This is an abstraction in process to help with world frame and viewport abstraction
- **GRS.cpp –** Functions related to reading GRS formatted files
- **draw.cpp –** Fully functional drawing functions like Fern. Functions capable of being in a callback function followed by flush.
- **hw1.cpp -** functions specific to hw1
- **main.cpp –** the main!
- **vshader1.glsl** – My first vertex shader!
- **fshader1.glsl** – my first fragment shader!