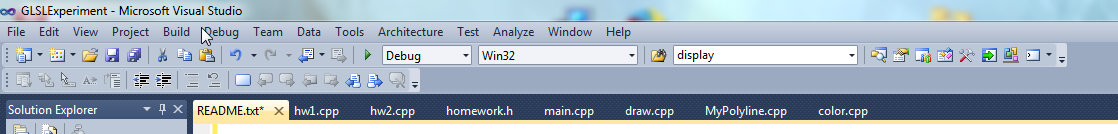
**CS 4731 Homework 3 Robert Dabrowski**

**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



**Mandatory Key Controls**

* "n": Slide camera 1 unit in the +ve n direction
* "N": Slide camera 1 unit in the -ve n direction
* "v": Slide camera 1 unit in the +ve v direction
* "V": Slide camera 1 unit in the -ve v direction
* "u": Slide camera 1 unit in the +ve u direction
* "U": Slide camera 1 unit in the -ve u direction
* "j": Change camera yaw by 2 degrees in the clockwise direction
* "J": Change camera yaw by 2 degrees in the anticlockwise direction
* "k": Change camera roll by 2 degrees in the clockwise direction
* "K": Change camera roll by 2 degrees in the anticlockwise direction
* "l": Change camera pitch by 2 degrees in the clockwise direction
* "L": Change camera pitch by 2 degrees in the anticlockwise direction
* **(User hits 'e'):** Draw a box around each mesh showing the extents of each mesh wireframe. This keystroke should be a toggle button. If the extents box is not drawn, hitting e draws the extents box. If the extents box is already drawn, hitting e erases the extents box. Even though the mesh rotates about the Y axis, the extents box should not. It should be stationary around the mesh
* **(User hits 's'):** The arms of the hierarchy follow a gentle sinusoid. i.e. in addition to going round clockwise or counterclockwise, the hierarchy arms (and objects) also go slightly up and down like on a merry go round. The sinusoidal movements of all levels of the hierarchy should have the same amplitude (i.e. move up and down gently by the same amount). The s key should be a toggle key. If the sinusoidal movement is currently off, hitting the s key initiates the sinusoidal movement. If the sinusoidal movement is currently on, hitting the s key turns it off so that objects just move in a simple circle. Note that whether the sinusoidal movement is on or off, consecutive levels of the hierarchy should alternate between rotating clockwise and counter-clockwise.
* **(User hits 'r'):** Reset the camera position to its initial position. No pitch, yaw, roll or slide applied. Make sure to define a suitable "initial position".

**Additional Key Controls**

* **Default:** message noting “not implemented yet” printed to console
* **“q” :** quit and close window

**Program Structure**

1. Generic initialization functions are called
2. PLY Files are read in and static transform stack is prepared
3. HW3 specific keyboard and display functions are registered as call backs
4. Glut main loop
5. Display function draws all PLY meshes using matrix stack of transformations
6. Idle function runs in background for continuous motion (once started)

**New Files**

* Hw3.cpp – primary implementation of hw3
* matrixStack.cpp – matrix stack abstraction

**Old Files (some have minor additions)**

* Hw2.cpp – primary implementation of homework2 - deprecated
* PLYreader.cpp – utilities for reading PLY files
* PLYPicture.cpp - abstraction of PLY pictures
* **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!