Ju Young (Amy) Lee

CSCI 480 Spring 2013
Design Document Template

Purpose/Overview
The purpose of the assignment is to create a height field of given 256 x 256 image which the user specifies at the command line and allows for user interactive features. These features include rotating, translating, and scaling as well as rendering in at least three modes: points, wireframes, and triangles. Upon completion, jpeg images are stored to create an animation displaying the different features.

Requirements
This is taken directly from the assignment 1 document.

- Handle at least a 256x256 image for your height field at interactive frame rates (window size of 640x480). Height field manipulations should run smoothly.
- Be able to render the height field as points, lines("wireframe"), or solid triangles (with keys for the user to switch between the three).
- Render as a perspective view, utilizing GL's depth buffer for hidden surface removal.
- Use input from the mouse to spin the heightfield around using glRotate.
- Use input from the mouse to move the heightfield around using glTranslate.
- Use input from the mouse to change the dimensions of the heightfield using glScale.
- Color the vertices using some smooth gradient.
- Be reasonably commented and written in an understandable manner--we will read your code.
- Be submitted along with JPEG frames for the required animation. There's a 300 image maximum.
-

Extra Features

- Handles any size image
- Renders wireframe on top of solid triangles (*MESH*) which utilizes *GL_BLEND* on the wireframe so that it looks visually more appealing.
- Does a rainbow color mapping of the shapes to represent intensity (see functions: *rainbowScale(float z)*).
- Added extra keyboard GUI inputs

Instructions
The required source, header and resources (image files) are all included in the folder assign1, along with the visual studios (2012) project called assign1.

The initial image file chosen in the command argument is "OhioPyle-256.jpg".

Classes
No additional classes were created.
The important source and headers, however, are:

assign1.cpp : the main program that creates and displays the rendering objects

pic.h: creates a struct that contains the width, height of the pixel and the actual data in an array, and the bpp (the rgb channel).

Global Variables

These are new global variables that were created.

The four floating variables (*width, height, pixSize, scalingZ*) help with the actual rendering and coloring of the shapes. *Width*, *height*, and *pixSize* are the pixel sizes. Based on the picture size, the width and height are set. pixSize divides the z value which ranges from 0 to pixel size (ie. 256) by the pix size (ie. 256 in order to normalize it from 0 to 1.0f. This is utitlized for setting the *rainbowScale(). scalingZ* scales the z value to be half of the value for the visual enhancement. By itself, the z values were too big and thus *scalingZ* is set to 2 to half the actual z value.

There is an integer variable (*imageNum)* which works as a counter for saving screenshots.

There is a Boolean variable (*SAVE*) which controls when to start saving screenshots. It gets implemented in the *doIdle()* function.

There are two enums (*CONSTROLSTATE and RENDERSTATE*) that are named (*g_ControlState* and *g_RenderState*) respectively. *g_Controlstate* controls the controlling of ROTATE, TRANSLATE, and SCALE. *g_RenderState* controls the controlling of different rendering shapes which are DOTS (points), WIREFRAME, TRIANGLES, and MESH (combination of triangles and wireframe).
All other global variables had been defined in the starter code.

GUI

Users can interact with the heightmap using their keyboard (not case sensitive) and the mouse.

Mouse inputs are:

Inputs	*Motions*	*Result*
Ctrl+mouse	click and dragging	translates the shape
Shift+mouse	click and dragging	scales the shape in the user's preference
Mouse	click and dragging	rotates the shape about the (0,0,0) which is one corner of the shape

Keyboard inputs are:

Inputs	*Result*
1	rendering of points
2	rendering of wireframe
3	rendering of triangles
4	rendering of the mesh
w	rotates about the x axis "upward"
s	rotates about the x axis "downward"
a	rotates about the z axis "left"
d	rotates about the z axis "right"
p	zooms in (scale*1.1)

m	zooms out(scale*0.9)
i	start saving (max 300 screenshots)
o	stop saving

Functions

void myinit()	Clears color
void rainbowScale (float z)	Passes a floating variable z which has the z value of the current pixel you are drawing This sets the correct color for the vertices for rainbow scaling by dividing the z values into five different sections. If z value is between 0 and .2 f it ranges from red to orange. If z value is between .2 and .4 then it transitions from orange to yellow. So on so that it ranges from red to purple based on the intensity (if z value does get to .8-1.0f then it will be purple)
void renderPoints()	Renders the height map in points. Using two for loops, goes to every single point of the pix array and maps it as a point. Calls the *rainbowScale(z)* at every point.
void renderWireFrame()	Renders the height map as wireframes using GL_LINE_STRIP which are basically cross section of lines going through each column and each row. Thus, two different for loops are called to create lines along column and along row.
void renderTriangles()	Renders the height map as triangles using GL_TRIANGLE_STRIP. This program is executed row-1 and col-1 times
void renderMesh()	Renders the height map as a mesh which is a compilation of *renderTriangles()* and *renderWireFrames()* with an addition of GL_POLYGON_OFFSET and GL_BLEND to fix the z buffer fighting and a blend to make the wireframe transparent for visual enhancement.
void display()	Sets up camera view and controls which rendering to do using *switch(g_RenderState)*
void reshape(int x, int y)	creates paremeters for *glutReshapeFunc(reshape)*. This function reshapes the window size and sets up the *glViewPort(parameters)* with given x and y values which corresponds to the window size. Also the *gluPerspective(parameters).*
void keyboardFunc (unsigned char button, … int x, int y)	creates parameters for *glutKeyboardFunc(keyboardFunc)*. This function is what creates the keyboard GUI using switch function.

void doIdle()	does the screen update and calls the function *void saveScreenshot (char *filename)* to save the 300 jpeg images which is controlled by boolean SAVE (see keyboard input: ‘i’ and ‘o’)
int main (int argc, …	main function that controls and sets things in *glutMainLoop().* It reads in
char argv)**	the image file and creates the pic struct *g_pHeightData*.