

Lab 10: Structs

ENGN 150

Purpose: This lab introduces structs as a data structure to organize information.

Problem:

In this lab you are to implement a computer drafting program. The program reads in a set of objects and implements the ability to translate, scale, and rotate individual objects. Objects are defined by a set of lines, each line defined by its endpoints. Software is provided which will create a visual of your objects by allowing you to draw the individual lines.

Support:

Graphics code is provided to you supporting the drawing of lines. Student code needs to use the struct Point3D found in Point3D.h when define a point in a Cartesian coordinate system. The graphics is initialized by a call to InitializeGraphics which is already called for you in the provided main function. Lines are provided to the graphics display by individual calls to:

```
void DrawLine(Point3D pt1, Point3D pt2);
```

providing the two endpoints of the line as pt1 and pt2. Prior to drawing any lines for a new scene, the new scene should be cleared by a call to:

```
void RefreshGraphics();
```

Once all lines to be drawn are provided to the graphics by calling DrawLine for each line in a scene, the graphics can be updated to the new scene by a call to:

```
void UpdateGraphics();
```

All three of these functions are defined in OglWorker.h.

The idea is to define all lines in a scene by updating individual objects in the scene, and then the scene should be redrawn by providing all lines to the graphics, and then updating the graphics.

Approach:

Step 1: Define data structures.

Your program should work on a Scene which can have up to 10 objects. Each object is defined to have up to 20 lines to provide it a 3D shape. Each line has 2 endpoints, each a 3D point. A struct is provided for you to define a 3D point (found in Point3D.h).

You need to define a Scene, Object, and Line in Draw3D.h.

Step 2: Declare functions for the following operations in Draw3D.h and implement them in Draw3D.cpp.

InitScene – This function should initialize a scene by reading information about the scene from a file. A string should be provided to indicate the file to read. The format of the file is provided below.

DrawScene – This function draws the scene to the window. It should loop through all objects and for each object loop through all lines calling DrawLine (provided for you) for each line. The first line of the function should call RefreshGraphics() (provided for you) and the last line of the function should call UpdateGraphics() (provided for you).

Translate – This function should translate a single object by a given offset. The offset can be provided by a Point3D, where x, y, and z provide the x offset, y offset, and z offset. Then

for each point (pt) in the object (remember lines are defined by endpoints), the point pt should be modified by:

```
pt.x += offset.x
pt.y += offset.y
pt.z += offset.z
```

Scale – This function is provided an object, a base point, and a scale factor. It then scales the object's size by scaling the distance of all points in the object relative to the base point. This can be accomplished by translating the object by the negative of the base point (this moves the point by which the object will be scaled to the origin), then multiplying all points in the object by the factor, and then translating the object back to its original position (translate by the base point). Alternatively, each point pt in the object can be updated by

```
pt.x = (pt.x - base.x)*factor + base.x
pt.y = (pt.y - base.y)*factor + base.y
pt.z = (pt.z - base.z)*factor + base.z
```

RotateX – This function rotates an object about the x axis by an angle provided in degrees. Each point in the object must be rotated to its new position by

```
new_pt.x = old_pt.x
new_pt.y = old_pt.y * cos(angle) - old_pt.z * sin(angle);
new_pt.z = old_pt.y * sin(angle) + old_pt.z * cos(angle);
```

where old_pt is the original point and new_pt is the rotated point.

RotateY – This function rotates an object about the y axis by an angle provided in degrees. Each point in the object must be rotated to its new position by

```
new_pt.x = old_pt.z * sin(angle) + old_pt.x * cos(angle);
new_pt.y = old_pt.y
new_pt.z = old_pt.z * cos(angle) - old_pt.x * sin(angle);
```

where old_pt is the original point and new_pt is the rotated point.

RotateZ – This function rotates an object about the z axis by an angle provided in degrees. Each point in the object must be rotated to its new position by

```
new_pt.x = old_pt.x * cos(angle) - old_pt.y * sin(angle);
new_pt.y = old_pt.x * sin(angle) + old_pt.y * cos(angle);
new_pt.z = old_pt.z
```

where old_pt is the original point and new_pt is the rotated point.

Input File Format:

The initial scene is defined in a file. A sample file is provided in SceneFile.txt. The first line in the file defines how many objects are in the scene. Then each object is defined individually by providing the number of lines in the object on the first line in the file for that object, and then defining each line on successive file lines, each file line having x, y, and z values for the 1st endpoint of the line and again for the 2nd endpoint. The contents of SceneFile.txt for a single square pyramid is:

```
2
8
0.0  0.0  0.0  30.0  0.0  0.0
30.0  0.0  0.0  30.0  30.0  0.0
30.0  30.0  0.0  0.0  30.0  0.0
0.0  30.0  0.0  0.0  0.0  0.0
```

```

0.0  0.0  0.0 15.0 15.0 20.0
30.0  0.0  0.0 15.0 15.0 20.0
30.0 30.0  0.0 15.0 15.0 20.0
0.0 30.0  0.0 15.0 15.0 20.0
4
0.0  0.0  0.0 -10.0  0.0  0.0
-10.0  0.0  0.0 -10.0 -10.0  0.0
-10.0 -10.0  0.0  0.0 -10.0  0.0
0.0 -10.0  0.0  0.0  0.0  0.0

```

Tasks:

- Design the algorithms for each function.
- Implement the code in the provided project (provided project has graphics support).
- Test your code using the provided SceneFile.txt and the operation sequence provided in the main function.
- Add another object to the input file and test your ability to manipulate the objects individually. You should define a test sequence that you think tests all functionality. For instance, can you manipulate 1 object and not the other.

Deliverables:

- Design: 5 pts.
 - Define each function in your design.
 - Provide a behavioral description of the function – “What does it do”
 - Provide the algorithm for the function – “How does it do it”
- Visual Studio project 10 pts.
 - Make sure to properly document your code, especially your .h file as discussed in recitation.
- New scene file 2 pts.
- Demonstrate your program to the TA. 3 pts.

Extra Credit:

You can get 1 point of extra credit for each of the 5 functions (Translate, Scale, RotateX, RotateY, RotateZ) if you modify their behavior so that a single call to one does not just modify the object, but rather smoothly moves the object to its new location/position. To do so, it is suggested to modify each function so that rather than making one large move, you make 50000 small moves (change based on the speed of your computer, 50000 looks good on my computer), redrawing the scene each time you do so. This will change the behavior of each function, the algorithm, and possibly the interface (the function signatures), and thus main. Therefore, you should provide a new design and implementation.

While you can directly attempt to design and implement this, it is suggested to do the standard lab first and make sure you have the translate, scale and rotate operations working first.