Assignment #2:

Visualization Using OpenGL

Due September 12, before midnight

Goals:

The goals of this project are two-fold: (1) get you familiar with OpenGL programming and using OpenGL to do visualization; (2) get familiar with the PLY file format. You will be provided a skeleton code to start with.

You should start with the skeleton codes provided to you. Before you move on to the following tasks***, please first select a version of the skeleton codes with the interface that you are comfortable to work with! You can choose to work with either C++ (there are two versions of skeleton code in cpp, you can choose either) or Javascript.***

Tasks:

1. Writing question (10 points)

1.1 What are the ISSUES of the rainbow color scheme? Please find one to two examples from the provided data sets to illustrate these issues. You can use the rainbow_color() function provided in the end of this description to help you answer this question.

1.2 In what situations that rainbow colors can be used?

2. Modify the PLY file loader to load the given data (10 points)

You will be provided **a number of models** in PLY format with scalar or vector values defined on them. *For the data with vector values, simply compute their magnitude as the scalar field for this assignment.*

To load the data value, you need to modify the ply loader as follows:

For C++ skeleton code: In the "`Skeleton.h`" file, modify the "Vertex" class as follows:

```
class Vertex {
public:
  double x,y,z;
  float s;
  float vx, vy, vz;
  int index;

  …
}; // the highlighted code should be added to the current skeleton code by you
```

In the "`Geometry.cpp`" file, modify the "`Vertex_io`" structure as follows

```
typedef struct Vertex_io {
  float x,y,z;
  float s;
  float vx, vy, vz;
  void *other_props;       /* other properties */
} Vertex_io;
```

Modify the vertex property list as follows

```
PlyProperty vert_props[] = { /* list of property information for a vertex */
  {"x", Float32, Float32, offsetof(Vertex_io,x), 0, 0, 0, 0},
  {"y", Float32, Float32, offsetof(Vertex_io,y), 0, 0, 0, 0},
  {"z", Float32, Float32, offsetof(Vertex_io,z), 0, 0, 0, 0},
  {"s", Float32, Float32, offsetof(Vertex_io,s), 0, 0, 0, 0},
  {"vx", Float32, Float32, offsetof(Vertex_io,vx), 0, 0, 0, 0},
  {"vy", Float32, Float32, offsetof(Vertex_io,vy), 0, 0, 0, 0},
  {"vz", Float32, Float32, offsetof(Vertex_io,vz), 0, 0, 0, 0},
};
```

Modify the vertex property set up list of the routine "`Polyhedron::Polyhedron(FILE *file)`" as follows

```
    /* set up for getting vertex elements */

     setup_property_ply (in_ply, &vert_props[0]);
     setup_property_ply (in_ply, &vert_props[1]);
     setup_property_ply (in_ply, &vert_props[2]);

     setup_property_ply (in_ply, &vert_props[3]);
     setup_property_ply (in_ply, &vert_props[4]);
     setup_property_ply (in_ply, &vert_props[5]);
     setup_property_ply (in_ply, &vert_props[6]);
      …

      /* copy info from the "vert" variable */
      vlist[j] = new Vertex (vert.x, vert.y, vert.z);
      vlist[j]->other_props = vert.other_props;
      vlist[j]->s = vert.s;
      vlist[j]->vx = vert.vx;
      vlist[j]->vy = vert.vy;
      vlist[j]->vz = vert.vz;
```

For Javascript version: In plyLoader.js file change the buffer variable inside parseASCII() function as follows,

```
var buffer = {
        indices: [],
        vertices: [],
        scalar: [],
        velocityVector: [],
 };
```

Modify the postProcess() function and add following attributes to the buffer,

```
.
.
geometry.addAttribute( 'position', new Float32BufferAttribute( buffer.vertices, 3 ) );
geometry.addAttribute('scalar', new Float32BufferAttribute(buffer.scalar, 1) );
geometry.addAttribute('velocityVector', new Float32BufferAttribute(buffer.velocityVector, 3) );
.
.
```

```
Add the following statements to the HandleElement() function,
```

```
.
.
buffer.vertices.push( element.x, element.y, element.z );
buffer.scalar.push(element.s);
buffer.velocityVector.push(element.vx, element.vy, element.vz);
if ( 'nx' in element && 'ny' in element && 'nz' in element ) {
        buffer.normals.push( element.nx, element.ny, element.nz );
}
.
.
```

Once you modify these functions in plyLoader.js you can access these attributes in assignment2.js script.

Please develop an interface to load different data sets.

3. Play with colors

3.1 Implement the “blue-white-red (BWR)” and the “heat map” continuous color schemes **(30 points)**

Apply these two color schemes to the provided data and report what you see (any extrema and saddle points in the data? Is the field smooth? Any patterns? Etc.)

3.2 Implement a discrete color scheme **(10 points)**

3.3 Implement the following user interactions **(40 points)**

(a) Provide an interface for the user to modify the value in the data range that is mapped to white for the BWR. Report what you discover for the given data sets. Given a particular data set, which threshold value for white is more suitable? Why?

(b) In addition to the linear mapping, provide one non-linear mapping from the data value to colors. Describe how you design your new color mapping (or transfer) function and how the generated result with the new function different from the linear mapping. Which mapping function do you think is more suitable for the data? Why?

Note that to enable color shading under lighting condition, you need to add the following line in the `Display()` routine before you render your colored surfaces if it is not already added.

```
……
glEnable(GL_COLOR_MATERIAL);
……
```

PLEASE submit your source code and report via blackboard.

Grades:

Tasks	*Total points*
1	10
2	10
3.1	30
3.2	10
3.3	40

There are other color mapping you can find in the internet, here is one good site http://dept.astro.lsa.umich.edu/~msshin/science/code/matplotlib_cm/ .

Example – color coding in cpp

```
// An example implementation of rainbow color coding
void rainbow_color (float s,
                    float s_max,
                    float s_min,
                    float rgb[3])
{
```

```
    float t = (s-s_min)/(s_max-s_min);
    // make sure t is between 0 and 1, if not, rgb should be black
    if (t<0 || t>1){
        rgb[0] = rgb[1] = rgb[2] = 0.;
        return;
    }

    float hsv[3] = {1.};
    // map the scalar value linearly to the hue channel of the HSV
    hsv[0] = (1. - t) * 240;
    hsv[1] = hsv[2] = 1.; // set the saturation and value as 1

    // Call the HSV to RGB conversion function
    HsvRgb(hsv, rgb);
}
```

Example- color coding in Js

```
function rainbow_color(s_min, s_max, s)
  {
    var hsv = [];
    var t = (s - s_min) / (s_max - s_min);
    // make sure t is between 0 and 1, if not, rgb should be black
    if(t < 0 || t > 1)
    {
      var rgb = [];
      rgb[0] = rgb[1] = rgb[2] = 0.0;
      return rgb;
    }
    // map the scalar value linearly to the hue channel of the HSV
    hsv[0] = (1 - t)*240;
    // set the saturation and value as 1
    hsv[1] = 1.0;
    hsv[2] = 1.0;
    // Call the HSV to RGB conversion function
    var rgb = hsvRgb(hsv);
    return rgb;
  }
```

Be creative and have fun!