

# CS 553 Scientific Visualization

---

Project #4:

Volume Slicing

Project #5:

Wireframe Isosurfaces

Yu Zhang

Zhangy6@onid.oregonstate.edu

Master Student in Computer Science

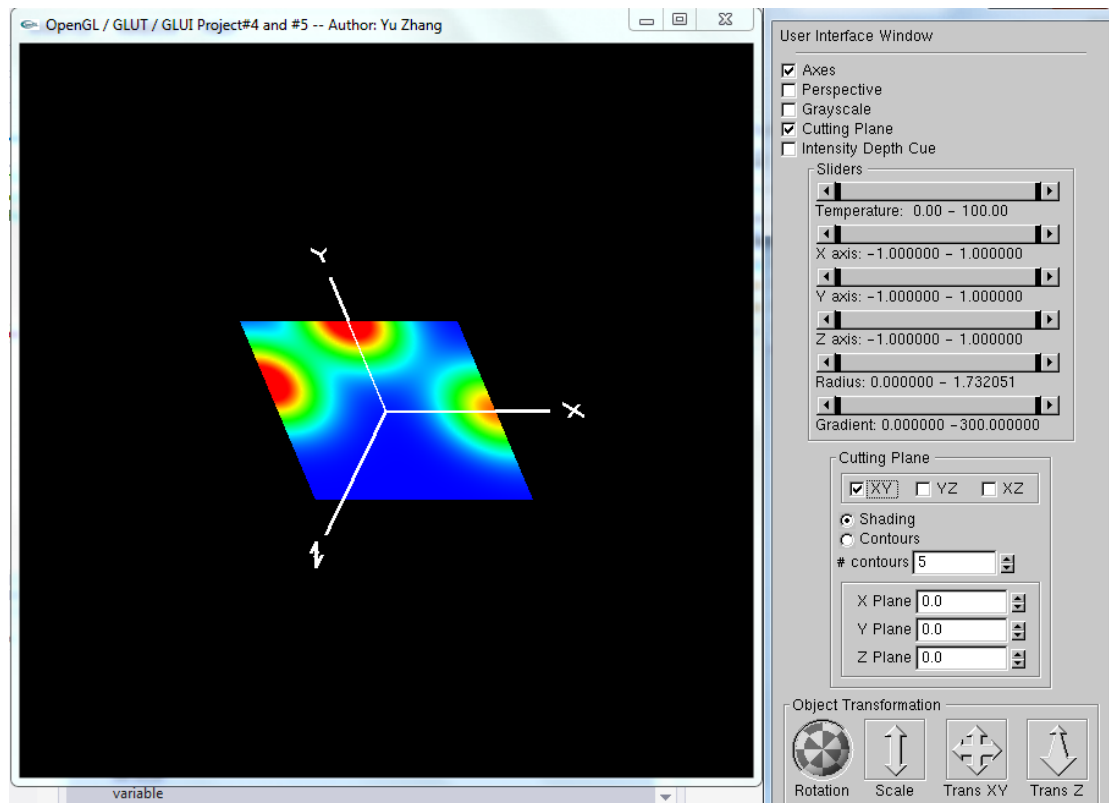
School of Electrical Engineering and Computer Science

Oregon State University

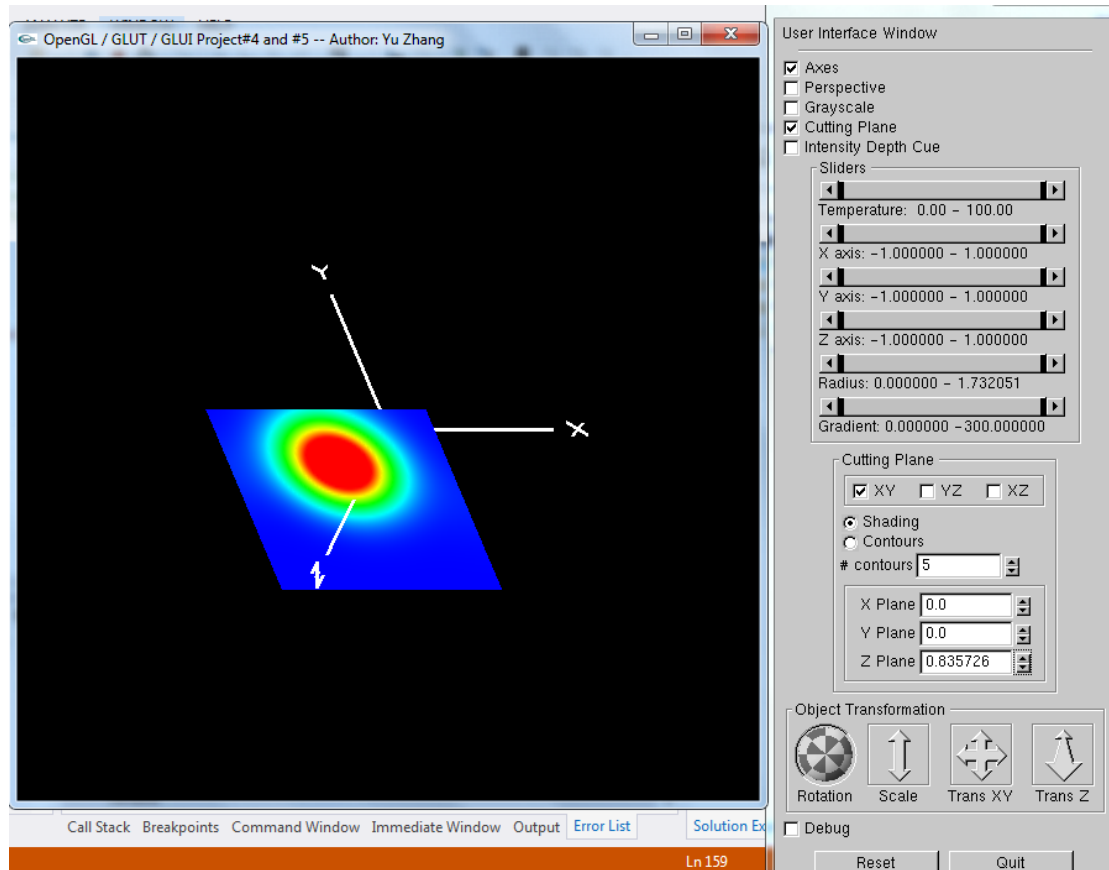
04/14/2015

# 1 Images of my projects

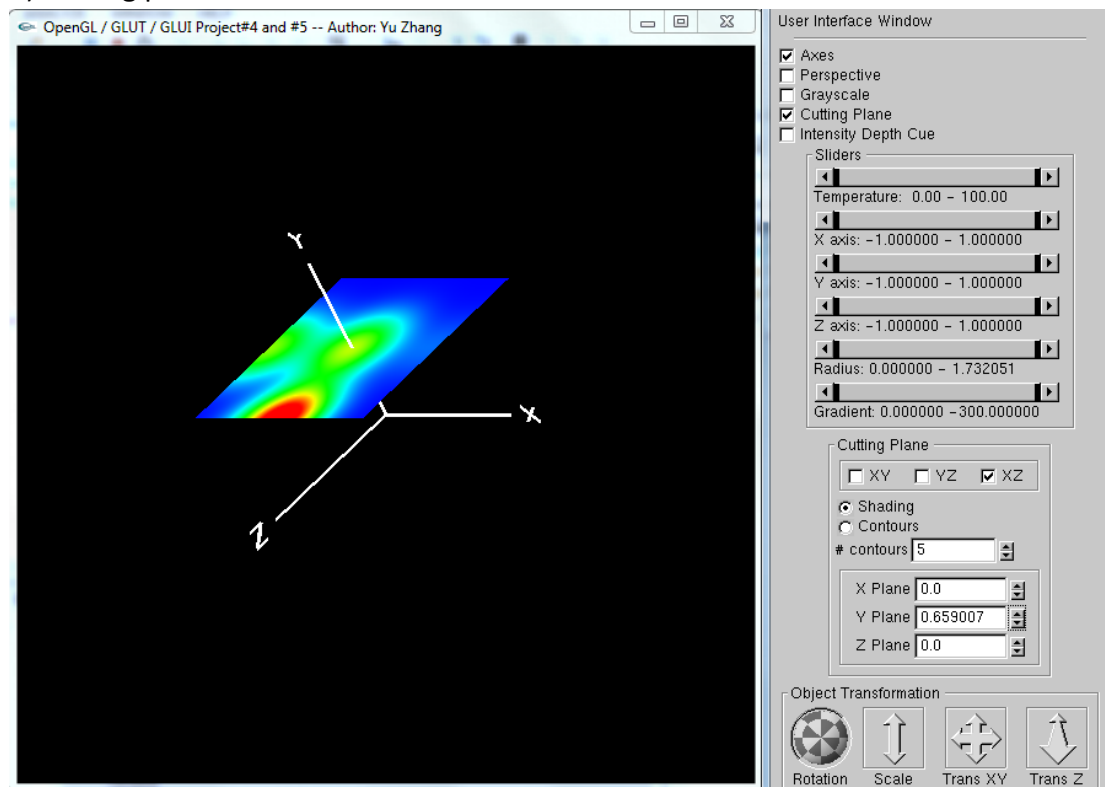
## a) Cutting plane --- PlaneXY:



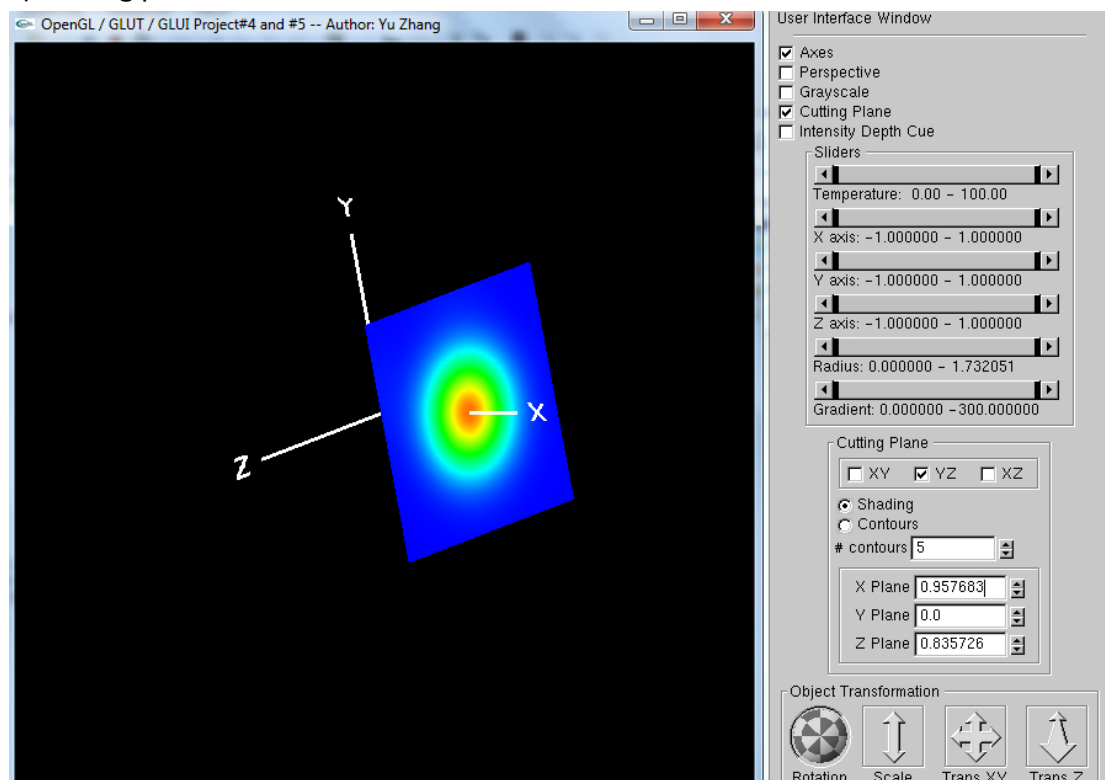
and movement of PlaneXY:



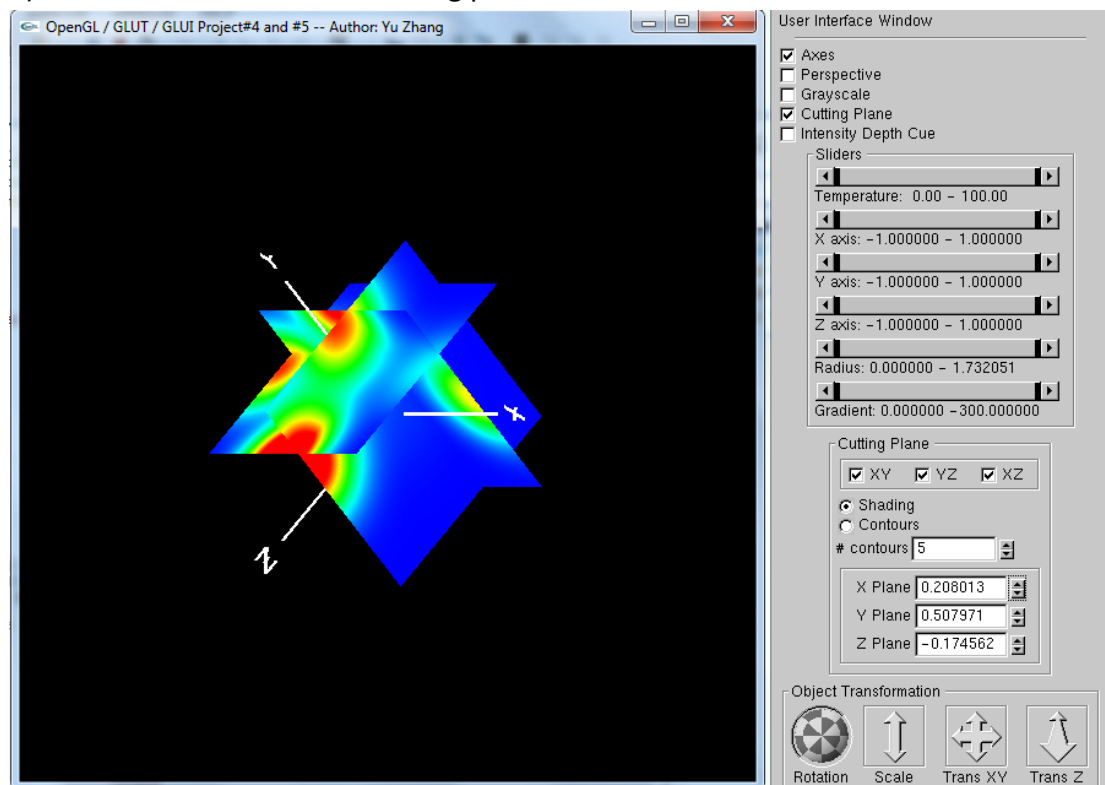
## b) Cutting plane --- PlaneXZ:



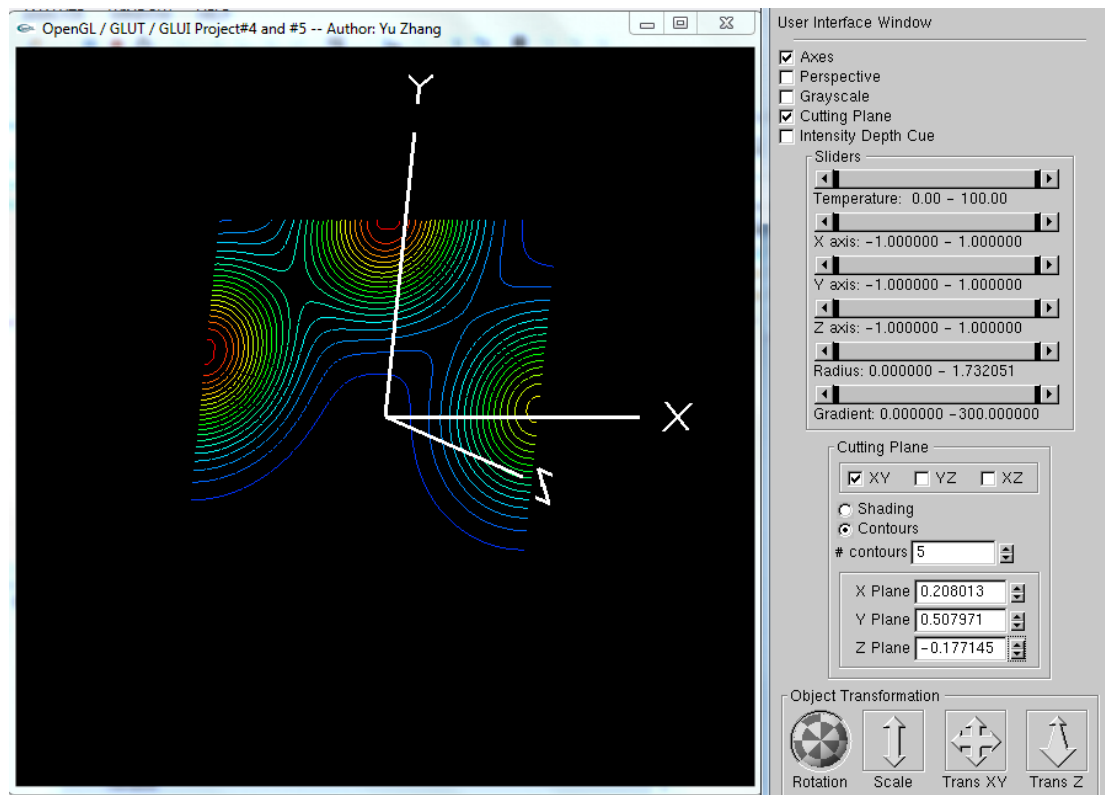
## c) Cutting plane --- PlaneYZ:



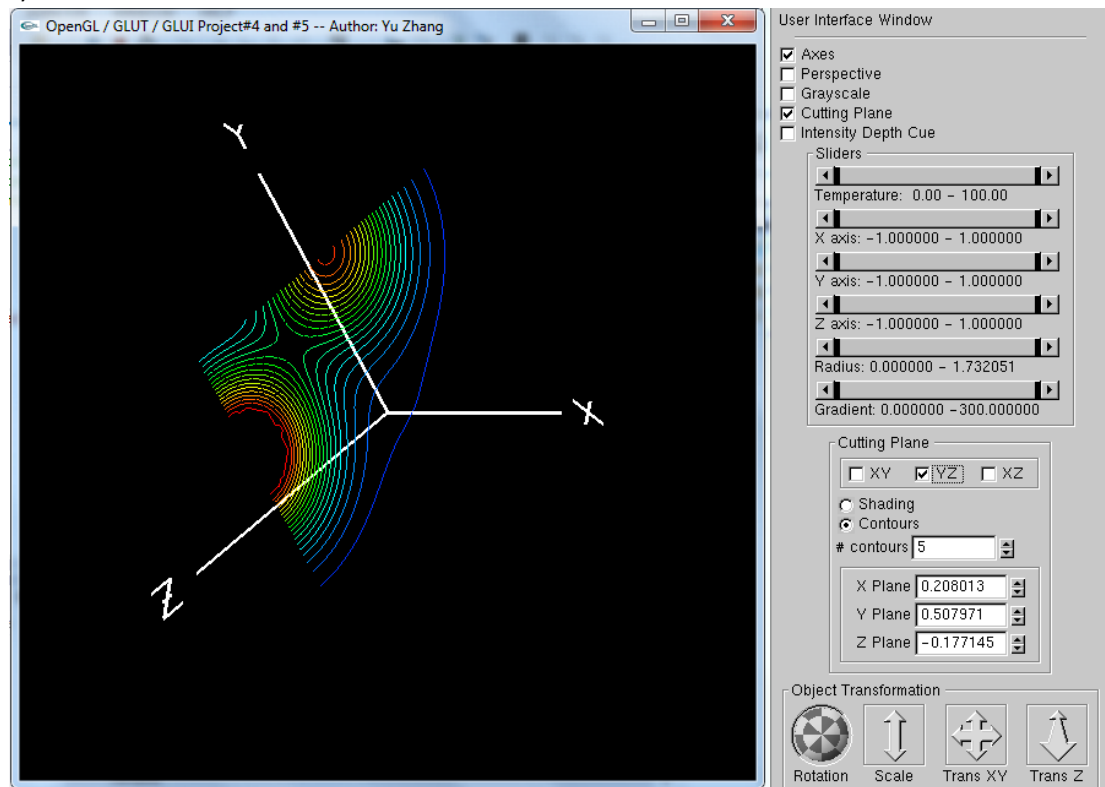
d) The combination of each cutting plane:



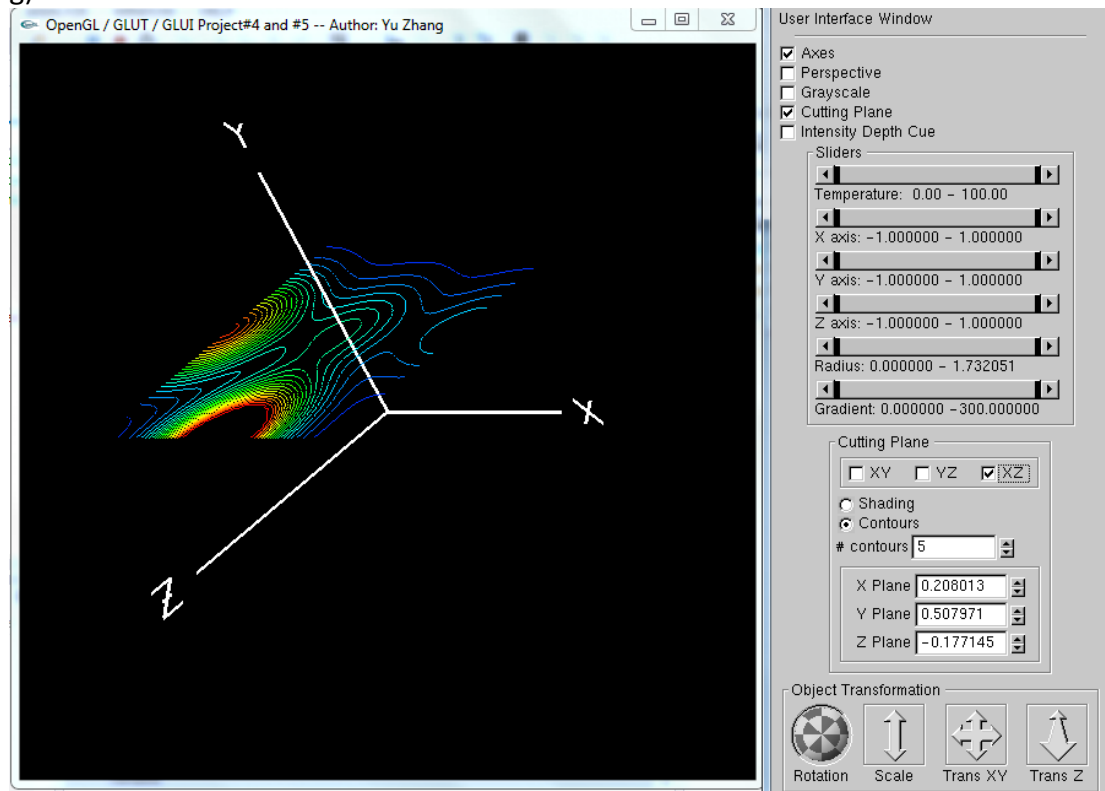
e) Contour lines --- PlaneXY



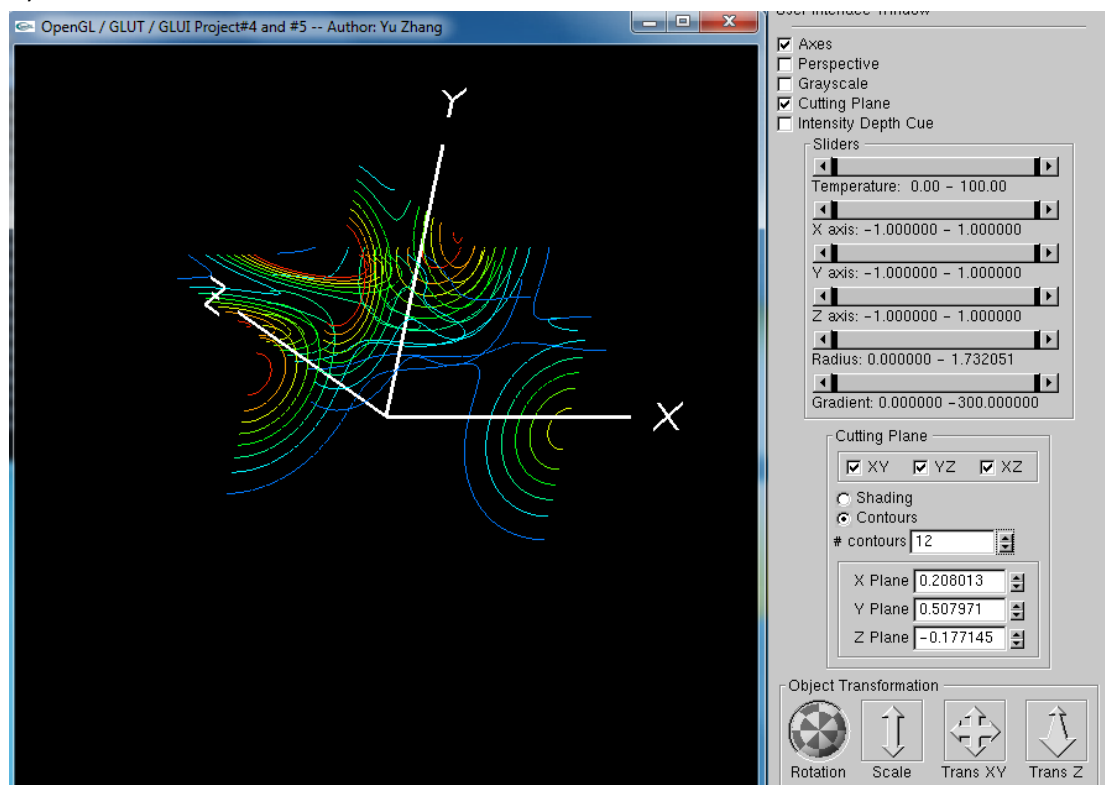
# f) Contour lines --- PlaneYZ



# g) Contour lines --- PlaneXZ

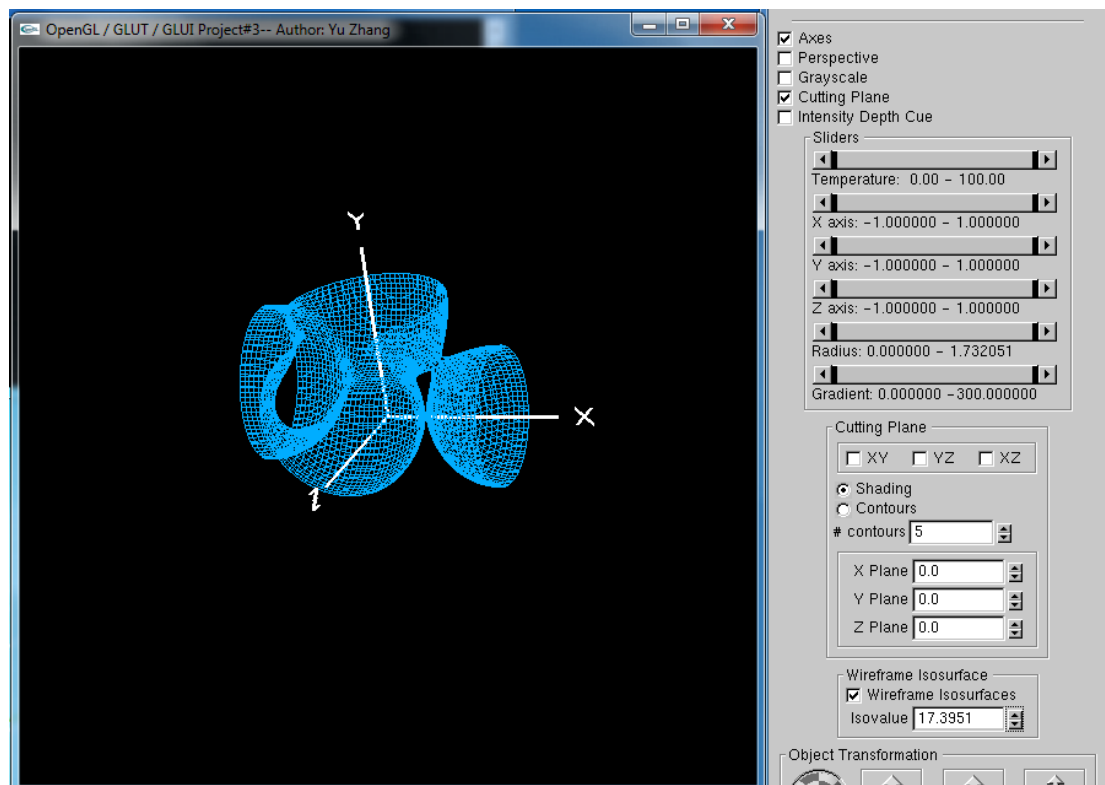


## h) The combination of contour lines

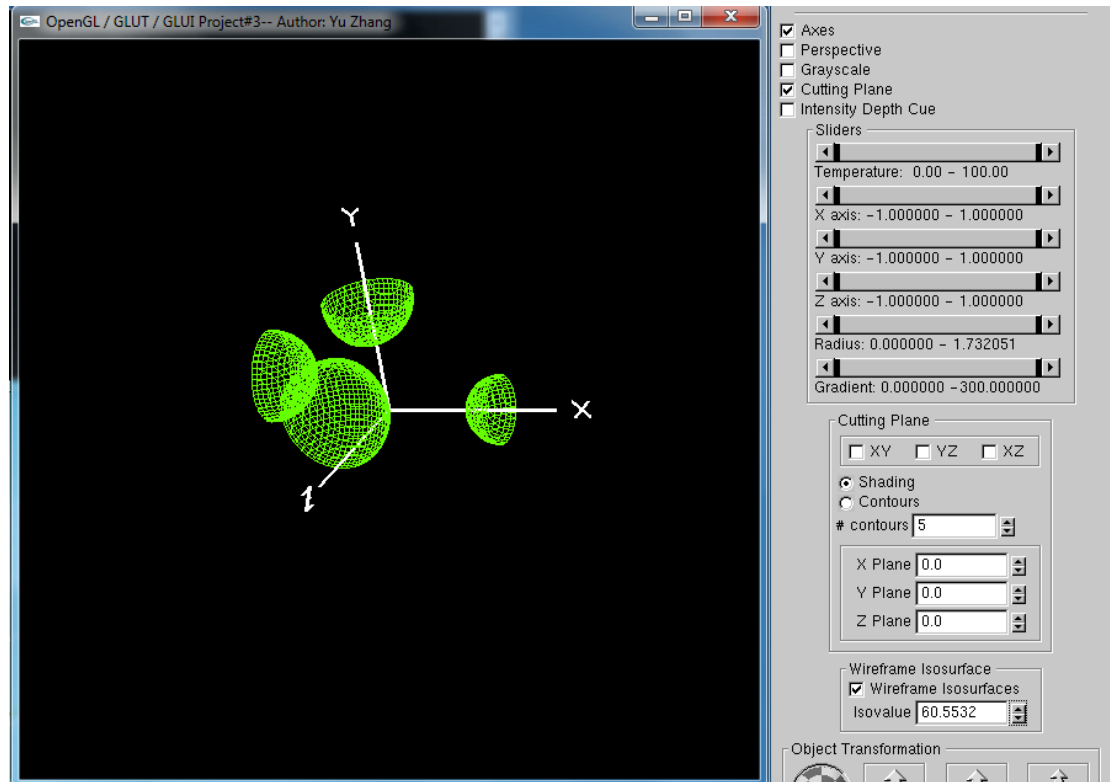


## i) Isosurfaces

When the isovalue is set as 17.3951



and the isovalue is set as higher to 60.5532

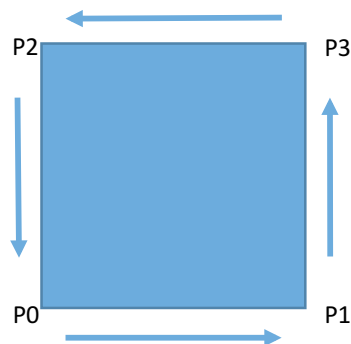


## 2 Main points of my project

This report contains project 4 and project 5, so this report will introduce 3 main things: cutting plane, contour lines, and isosurfaces.

### a) Cutting plane

For implementing this part, I initialized a new set of nodes by 2-dimension arrays, such as `PlaneXY[][]`, this array is 2D because cutting plane displays a slice which is 2D and the plane only changes when the axis which perpendicular to this plane changes. So I compute each node of one cutting plane by the idea that used in project3, and set these *struct nodes* as the parameters to function `ColorSquare(struct nodeTwoD*, struct nodeTwoD*, struct nodeTwoD*, struct nodeTwoD*)`. This function is to compute and draw each squares with four points we gave before.



We have stored each node's information in the `struct nodeTwoD*`, and then we need to use 4 point to do the color interpolation, the sequence of 4 points is

P0 → P1 → P3 → P2 → P0 and the coordinate order is (0,0), (1,0), (1,1), (0,1), (0,0).

So that for implementing this part, I use the following way to pass arguments to the ColorSquare().

```
ColorSquare(&PlaneXY[i][j], &PlaneXY[i + 1][j], &PlaneXY[i + 1][j + 1],
&PlaneXY[i][j + 1]);
```

In function ColorSquare(), I used four glColor3fv() and four glVertex3f() to draw the square.

After drawing each square, we need to add spinner to the project and enable the spinner to change the cutting plane position. Luckily, our sweet professor offers us the codes of glui codes, and what we need to do is just to modify a few codes and enable the arguments PlaneX, PlaneY, PlaneZ to receive the user commands from the interface.

## b) Contour lines

Contour lines is also computed in a square, the algorithm is done like this:

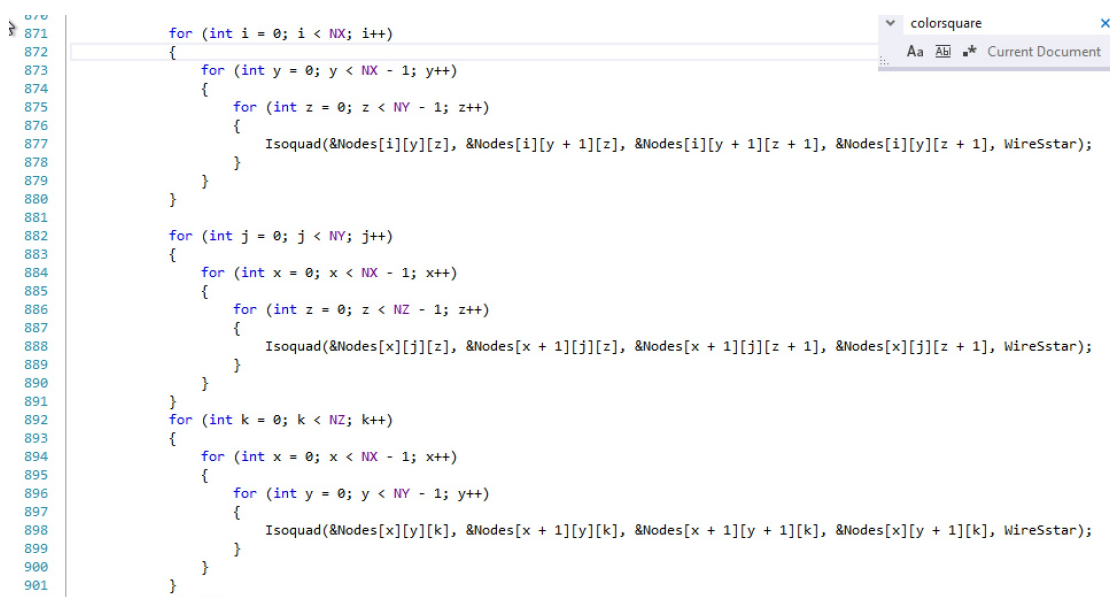
1st  $t^*$  should be computed:

$$t^* = \frac{S^* - S_0}{S_1 - S_0}$$

In this equation,  $S^*$  is the input by a for loop from TEMPMIN to TEMPMAX and each time pluses NsSteps. When  $t^*$  is between 0 and 1, the position of the node is calculated. I used a two dimensional array plot[n][3] to store node's information. Specifically, n means the number of intersections. When n is equal to 2, we draw the line between the two nodes. There's another restriction that we need to ignore the edge that the two vertices have the same temperature. The order of computing the square is also P0 → P1 → P3 → P2 → P0.

## C) Isosurfaces

This part belongs to project 5, and if the contour lines is implemented, this part should be easier. The main point of this part is to use 3 separated for-loops to calculate and draw contour lines in each dimension. Such as:



```
871 for (int i = 0; i < NX; i++)
872 {
873     for (int y = 0; y < NY - 1; y++)
874     {
875         for (int z = 0; z < NZ - 1; z++)
876         {
877             Isoquad(&Nodes[i][y][z], &Nodes[i][y + 1][z], &Nodes[i][y + 1][z + 1], &Nodes[i][y][z + 1], WireSstar);
878         }
879     }
880 }
881
882 for (int j = 0; j < NY; j++)
883 {
884     for (int x = 0; x < NX - 1; x++)
885     {
886         for (int z = 0; z < NZ - 1; z++)
887         {
888             Isoquad(&Nodes[x][j][z], &Nodes[x + 1][j][z], &Nodes[x + 1][j][z + 1], &Nodes[x][j][z + 1], WireSstar);
889         }
890     }
891 }
892 for (int k = 0; k < NZ; k++)
893 {
894     for (int x = 0; x < NX - 1; x++)
895     {
896         for (int y = 0; y < NY - 1; y++)
897         {
898             Isoquad(&Nodes[x][y][k], &Nodes[x + 1][y][k], &Nodes[x + 1][y + 1][k], &Nodes[x][y + 1][k], WireSstar);
899         }
900     }
901 }
```

This structure is clear to show how to calculate each dimension's contour lines.