```
#include <stdio.h>
#include <stdlib.h>
#include<iostream>
#include <math.h>
#include <windows.h>
#include <vector>
#include <glut.h>
#include<algorithm>
#include <unistd.h>
#define Length 4

using namespace std;
double A[Length][Length]; //Angle of Rotation
double S[Length][Length]; //1 if Selected 0 if not
double C[Length][Length][3];//Color in RGB
double V[Length][Length]; //Value
double F[Length][Length]; //Finished
vector<int> values;
vector<int> R,G,B;
double Size=10;

int SELECTED_X=0,SELECTED_Y=0;

void darwSquare(double SIZE, double R, double G, double B){
    glColor3f(R,G,B);
    glBegin(GL_QUADS);
    {
        glVertex3f(0,0,0);
        glVertex3f(SIZE,0,0);
        glVertex3f(SIZE,SIZE,0);
        glVertex3f(0,SIZE,0);
    }
    glEnd();
}
```

Here the global variable A is the angle of rotation. S is mainly used as whether the cube is selected or not. C identifies the colour of the block. V holds the value of the blocks and F determines whether the puzzle is finished or not.

In the “drawSquare” Function the square in which the whole work is running is drawn.

```
35  void drawHollowRectangle(int X,int Y){
36      int Border=1;
37      glTranslatef(20*X,20*Y,0);
38      glBegin(GL_LINES);
39
40      glColor3f(1, 1, 0);//
41
42      glVertex3f(-Border, -Border, 0);
43      glVertex3f(Size+Border, -Border, 0);
44
45      glColor3f(1, 1, 0);//
46      glVertex3f(Size+Border, -Border, 0);
47      glVertex3f(Size+Border, Size+Border, 0);
48
49      glVertex3f(Size+Border, Size+Border, 0);
50      glVertex3f(-Border, Size+Border, 0);
51
52      glVertex3f(-Border, Size+Border, 0);
53      glVertex3f(-Border, -Border, 0);
54
55      glEnd();
56  }
```

“drawHollowRectangle” function is used as the function where the selector block is drawn. Mainly the block which selects the blocks.

```
57  void drawGrid(){
58      glBegin(GL_LINES);
59
60      glColor3f(1, 0, 0);//red
61      glVertex3f(-1000, 0, 0);
62      glVertex3f(1000, 0, 0);
63
64      glColor3f(1, 0, 0);//red
65      glVertex3f(0, -1000, 0);
66      glVertex3f(0, 1000, 0);
67
68      glEnd();
69
70  }
71  void drawTiles(int length){
72      for(int i=0;i<length;i++){
73          for(int j=0;j<length;j++)
74          {
75              glPushMatrix();
76              glTranslatef(20*i,20*j,0);
77              glTranslatef(Size/2,0,0);
78              glRotatef(A[i][j],0,1,0);
79              glTranslatef(-Size/2,0,0);
80              darwSquare(Size,C[i][j][0],C[i][j][1],C[i][j][2]);
81              glPopMatrix();
82          }
83      }
84  }
```

“drawGrid” function mainly draws the grid alongside all the cubes visible in the square.

“drawTiles” is used to draw all the tiles visible in the grid.

```
85  bool isMatch(int x1,int y1,int x2,int y2){
86      if(V[x1][y1]==V[x2][y2])
87          return true;
88      return false;
89  }
90  void display(){
91      glClearColor(0, 0, 0, 0);
92      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
93
94      /**************************
95      / set-up camera (view) here
96      **************************/
97
98      //load the correct matrix -- MODEL-VIEW matrix
99      //specify which matrix is the current matrix
100     glMatrixMode(GL_MODELVIEW);
101
102     //initialize the matrix
103     //replace the current matrix with the identity matrix [Diagonals have 1, others have 0]
104     glLoadIdentity();
105
106     //now give three info
107     //1. where is the camera (viewer)?
108     //2. where is the camera looking?
109     //3. Which direction is the camera's UP direction?
110     gluLookAt(40,20,150,    40,20,0,    0,1,0);
111     //drawGrid();
112     drawTiles(Length);
113     glPushMatrix();
114     drawHollowRectangle(SELECTED_X,SELECTED_Y);
115     glPopMatrix();
116     //ADD this line in the end --- if you use double buffer (i.e. GL_DOUBLE)
117     glutSwapBuffers();
118 }
```

“isMatch” is the function where it shows whether the tiles are matching are not. It basically checking condition for the tiles to match.

“display” this is the function every function is being called to show in the cmd.

```
void animate(){
    //codes for any changes in Models, Camera
    //this will call the display AGAIN
    vector<int> v;
    for(int i=0;i<Length;i++){
        for(int j=0;j<Length;j++){
            if(S[i][j]==1){
                    if(A[i][j]==180){
                        v.push_back(i);
                        v.push_back(j);
                    }
                if(A[i][j]<180)
                    A[i][j]+=.5;
                C[i][j][0]=R[V[i][j]];
                C[i][j][1]=G[V[i][j]];
                C[i][j][2]=B[V[i][j]];
            }
            if(A[i][j]==180 && v.size()==4){

                    if(isMatch(v[0],v[1],v[2],v[3])){ //Matched

                            A[v[0]][v[1]]=0;
                            F[v[0]][v[1]]=1;
                            S[v[0]][v[1]]=0;
                            C[v[0]][v[1]][0]=0;
                            C[v[0]][v[1]][1]=0;
                            C[v[0]][v[1]][2]=0;



                        A[v[2]][v[3]]=0;
                        F[v[2]][v[3]]=1;
                        S[v[2]][v[3]]=0;
                        C[v[2]][v[3]][0]=0;
                        C[v[2]][v[3]][1]=0 ;
                        C[v[2]][v[3]][2]=0;

                        break;
                        cout << "Game over!!!!" << endl;

                }
                else{   //Didn't match
                    A[v[0]][v[1]]=0;
                    S[v[0]][v[1]]=0;
                    C[v[0]][v[1]][0]=0;
                    C[v[0]][v[1]][1]=0;
                    C[v[0]][v[1]][2]=1;


                    A[v[2]][v[3]]=0;
                    S[v[2]][v[3]]=0;
                    C[v[2]][v[3]][0]=0;
                    C[v[2]][v[3]][1]=0;
                    C[v[2]][v[3]][2]=1;
                    break;
                }
            }
        }
    }
    glutPostRedisplay();
}
```

“animate” is the function where mainly the rotation is done and also the matching condition is being checked in every loop.

```
180  void init(){
181      //clear the screen
182      glClearColor(0, 0, 0, 0);
183      //load the PROJECTION matrix
184      glMatrixMode(GL_PROJECTION);
185      //initialize the matrix
186      glLoadIdentity();
187      R.push_back(1);
188      R.push_back(1);
189      R.push_back(1);
190      R.push_back(0);
191      G.push_back(1);
192      G.push_back(0);
193      G.push_back(1);
194      G.push_back(1);
195      B.push_back(1);
196      B.push_back(0);
197      B.push_back(0);
198      B.push_back(0);
199      for(int i=0;i<Length;i++){
200          values.push_back(i);
201          values.push_back(i);
202          values.push_back(i);
203          values.push_back(i);
204      }
205      random_shuffle(values.begin(),values.end());
206      int val=0;
207      for(int i=0;i<Length;i++){
208          for(int j=0;j<Length;j++){
209
210              A[i][j]=0;
211              S[i][j]=0;
212              V[i][j]=values[val];
213              val++;
214              F[i][j]=0;
215              C[i][j][0]=0;
216              C[i][j][1]=0;
217              C[i][j][2]=1;
218          }
219      }
220      gluPerspective(70, 1, 0.1, 10000.0);
221  }
222
223  void keyboardListener(unsigned char key, int x, int y){
224      if(key=='w')
225      {
226          if(SELECTED_Y<Length-1)
227          {
228              SELECTED_Y+=1;
229          }
230      }
231      else if(key=='s')
232      {
233          if(SELECTED_Y>0)
234          {
235              SELECTED_Y-=1;
236          }
237      }
238      else if (key=='a')
239      {
240          if (SELECTED_X>0)
241          {
242              SELECTED_X-=1;
243          }
244      }
245      else if (key== 'd')
246      {
247          if (SELECTED_X<Length-1)
248          {
249              SELECTED_X+=1;
250          }
251      }
252      else if (key == ' ')
253      {
254          if(F[SELECTED_X][SELECTED_Y]==0)
255              S[SELECTED_X][SELECTED_Y]=1;
256      }
257  }
```

"init" function is used as the projection plane and all the colors are inserted in ever tiles in a random manner. And the whole grid is shown in the perspective method.

"keyboardListener" is the function where the keyboard functionalities are done.

```
293  int main(int argc, char **argv){
294      //initialize the GLUT library
295
296      glutInit(&argc, argv);
297
298      glutInitWindowSize(1000, 1000);
299      glutInitWindowPosition(0, 0);
300
301      /*
302      glutInitDisplayMode - inits display mode
303      GLUT_DOUBLE - allows for display on the double buffer window
304      GLUT_RGBA - shows color (Red, green, blue) and an alpha
305      GLUT_DEPTH - allows for depth buffer
306      */
307      glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGB);
308
309      glutCreateWindow("Match Pair");
310
311      //codes for initialization
312      init();
313
314      //enable Depth Testing
315      glEnable(GL_DEPTH_TEST);
316
317      //display callback function
318      glutDisplayFunc(display);
319
320      glutSpecialFunc(specialKeyListener);
321      glutKeyboardFunc(keyboardListener);
322
323      //what you want to do in the idle time (when no drawing is occurring)
324      glutIdleFunc(animate);
325
326      //The main loop of OpenGL
327      glutMainLoop();
328
329      return 0;
330  }
```

“main” function is the function where the window size is determined and display function is called to put everything in one place.