CS 770/870 Assignment 2: Frequently-asked Questions

1. What are those 4×4 matrices?

   translate(tx,ty,tz):

   ```
   1 0 0 tx
   0 1 0 ty
   0 0 1 tz
   0 0 0 1
   ```

   scale(sx,sy,sz):

   ```
   sx  0  0 0
    0 sy  0 0
    0  0 sz 0
    0  0  0 1
   ```

   Setting `c`=cos(angle), and `s`=sin(angle), we have:

   rotateX(angle):

   ```
   1  0  0  0
   0  c -s  0
   0  s  c  0
   0  0  0  1
   ```

   rotateY(angle):

   ```
    c  0  s  0
    0  1  0  0
   -s  0  c  0
    0  0  0  1
   ```

   rotateZ(angle):

```
c -s  0  0
s  c  0  0
0  0  1  0
0  0  0  1
```

2. How do I multiply two matrices?

   If C = A * B, then

   $C_{i,j} = \Sigma_k A_{i,k} B_{k,j}$

3. In `untangle`, what should I do, when?

   In `mouse_button_callback`, you must find out if the mouse press was on some vertex. Call `vertex_at`. If so, remember the vertex's index.

   In `mouse_position_callback`, IF a vertex is being dragged, change that vertex's x and y.

   In `display`, instead of drawing one QUAD, draw several QUADs, one per vertex.

   In `display`, instead of drawing on LINE, draw several, one per graph edge.

   In `display`, check first if there are any crossings. If so, use a greenish background color (at the top of the `display` function).

   You will also have to write some code to check if two edges cross.

4. How do I determine if two line segments cross?

   In class, I discussed how to check if the finite line segment through A, B, hits the infinite line through C, D.

   1. P = A + t (B - A)
   2. N dot (P - Q) = 0

   where Q is C, and N is perpendicular to D-C

   You get t = (N dot (Q - A)) / (N dot (B - A))

   Then plug into 1. to get P.

   If you have a working `GeomLib`, you can write C++ code that looks like math:

```
Point4 A, B, C, D;
Vector4 V = D - C;
Vector4 N(-V.Y(), V.X(), 0);
Point4 Q = C;
// check if denominator is zero before the next line:
double t = (N * (Q-A)) / (N*(B-A));
// check that 0 < t < 1.  If so,
Point4 P = A + t * (B - A);
```

5. How do I store the adjacency matrix? Should I use a `Matrix4`?

   NO. The graph may have more than 4 vertices, so the adjacency matrix may need more than 4x4 entries. Use a 2D array of booleans. Here is how to do that in C++. You have to allocate each row of the matrix, then you have fill in the matrix.

```
// GLOBAL VARIABLES:
int n_vertices;
bool **is_edge;

...
// read_graph:

ifstream ifs(filename, std::ifstream::in);
ifs >> n_vertices;
// allocate pointers for the rows
is_edge = new *bool[n_vertices];
for (int i = 0; i < n_vertices; i++) {
    // allocate a row
    is_edge[i] = new bool[n_vertices];
    // now, read one row of the matrix
    for (int j = 0; j < n_vertices; j++) {
        int flag;
        ifs >> flag;
        is_edge[i][j] = (flag == 1);
    }
}
```

   Here is another way, using `vectors`:

```
vector<vector<bool> > is_edge;

...
// read_graph:

for (int i = 0; i < n_vertices; i++) {
    is_edge.push_back(vector<bool>());
    for (int j = 0; j < n_vertices; j++) {
        int flag;
        ifs >> flag;
        is_edge[i].push_back(flag == 1);
    }
}
```

6. How to I store the positions of the vertices?

   There are `n_vertices` of them. So, create an array of `Point4`, or two arrays of `float` (x and y):

```
Point4 *vertices;

...

// read_graph:
vertices = new Point4[n_vertices];
```

   or another way:

```
float *vertex_x;
float *vertex_y;

...

// read_graph:
vertex_x = new float[n_vertices];
vertex_y = new float[n_vertices];
```