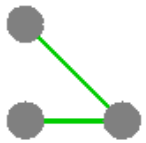
Graph System

Data Structures and Object-oriented Programming

Sai Keung Wong (黃世強)
National Yang Ming Chiao Tung University, Taiwan

Graph

- A graph G = (V, E) consists of a set of nodes (V) and a set of edges (E).
- An edge is attached at two nodes.



A gray dot is a node.
A green line segment is an edge.

Assumptions

- There are at most 1000 nodes and 1000 edges.
- The degree of a node is at most 1000. That is that there are at most 1000 edges attached at a node.

- We do not handle parallel edges.

Major task

- Design a data structure to manipulate a graph. You should use the given data structure GRAPH_NODE and GRAPH_EDGE. The position of a node is on the x-z plane. Keep the y-coordinate of a node as zero.

- Add a node (GRAPH_NODE)
- Add an edge (GRAPH_EDGE)
- Delete a node and all the edges attached at it

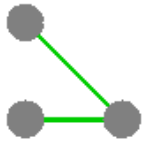
Modify the GRAPH_NODE and GRAPH_EDGE in graph_basics.h

- Add extra data members and functions if you want to do so.

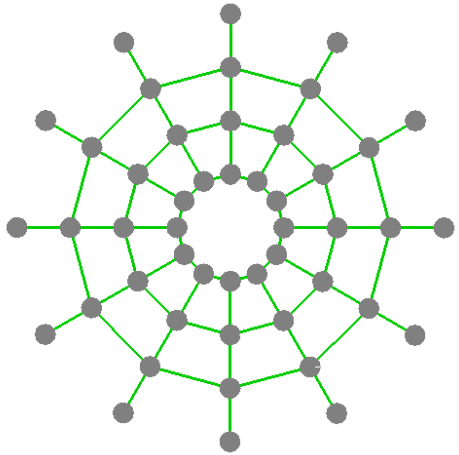
Tasks

int addNode(float x, float y, float z, float r = 1.0);	Add a node and return the id of the node
int addEdge(int nodeID_0, int nodeID_1);	Add an edge and return the id of an edge, where nodeID_0 and nodeID_1 are the node IDs of the two nodes of the edge.
void createDefaultGraph( )	Create the default graph
void createNet_Circular(int n, int num_layers);	create nodes arranged in circles
void createNet_Square(int n, int num_layers);	create nodes arranged in squares. Some nodes at the center are removed.
void createNet_RadialCircular(int n);	create nodes arranged in a circle and one node at center.
void createRandomGraph_DoubleCircles(int n);	create nodes arranged in two circles. The edges are randomly created for nodes in the inner and outer circle.

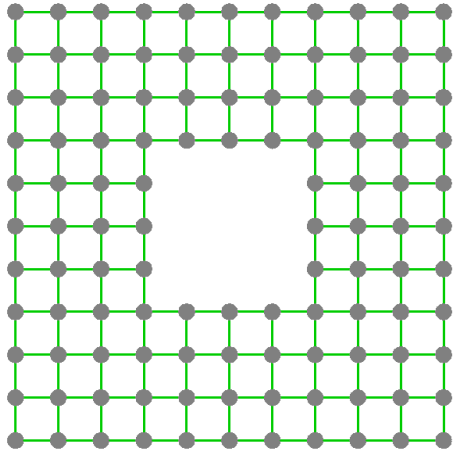
Graphs



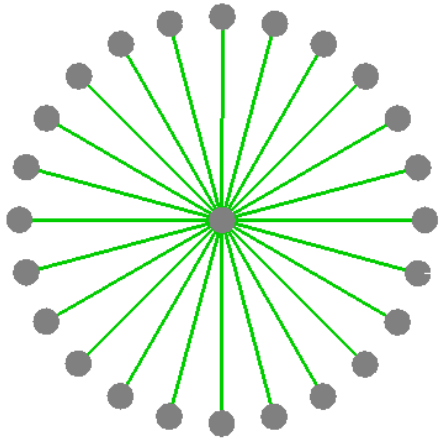
createDefaultGraph



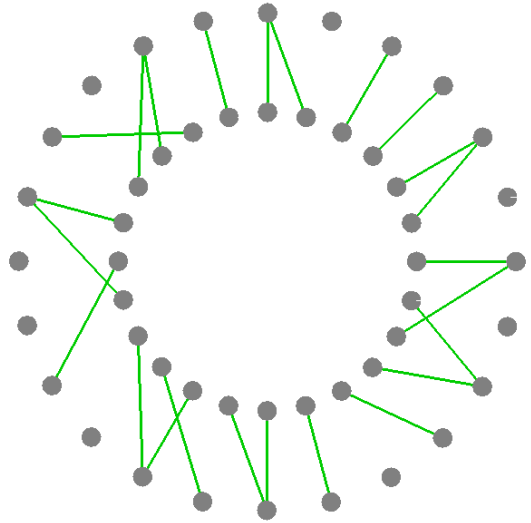
createNet_Circular



createNet_Square

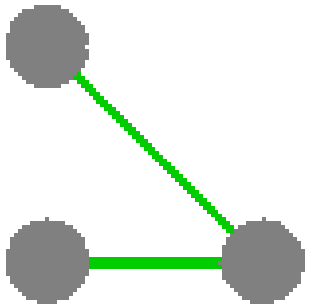


createNet_RadialCircular



createRandomGraph_DoubleCircles

createDefaultGraph. Press ‘1’



void createNet_Circular(int n, int num_layers);
Press ‘2’

createNet_Circular(12, 3);

There are 12 nodes arranged in each circle.

There are three inner layers and one outer layer.

There are edges connecting adjacent nodes between the inner layers. And also edges connect adjacent nodes in the same layer.

In the outer layer, there are no edges connecting the nodes in the outer layer.

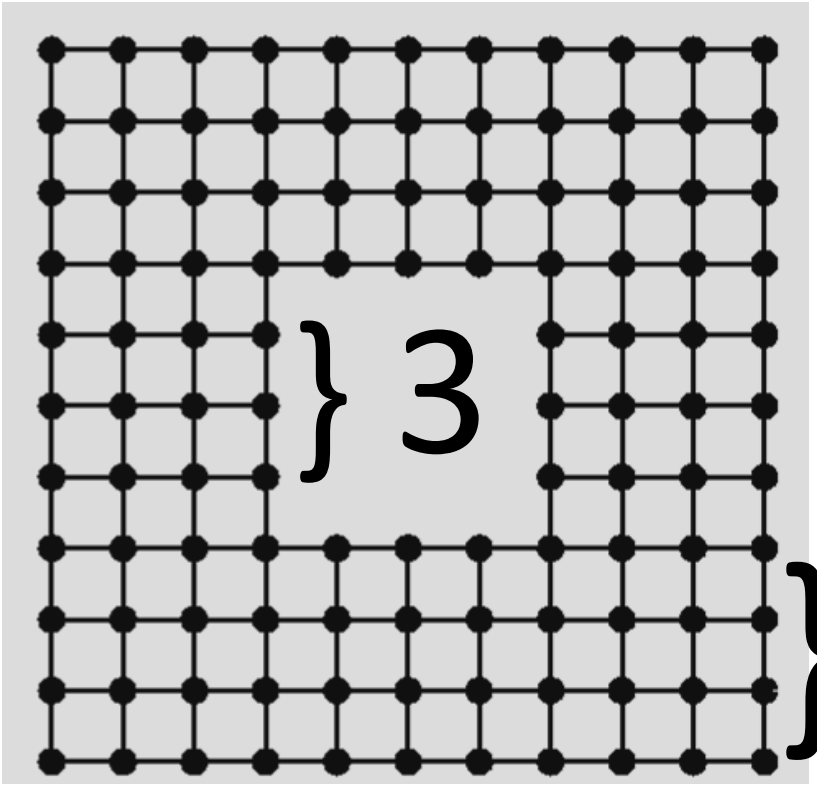
void createNet_Square(int n, int num_layers);
Press ‘3’

createNet_Square(3, 11)

n is the width of the inner square.

num_layers is the number of layers of the net.

Based on your own design, you can set your own parameters.

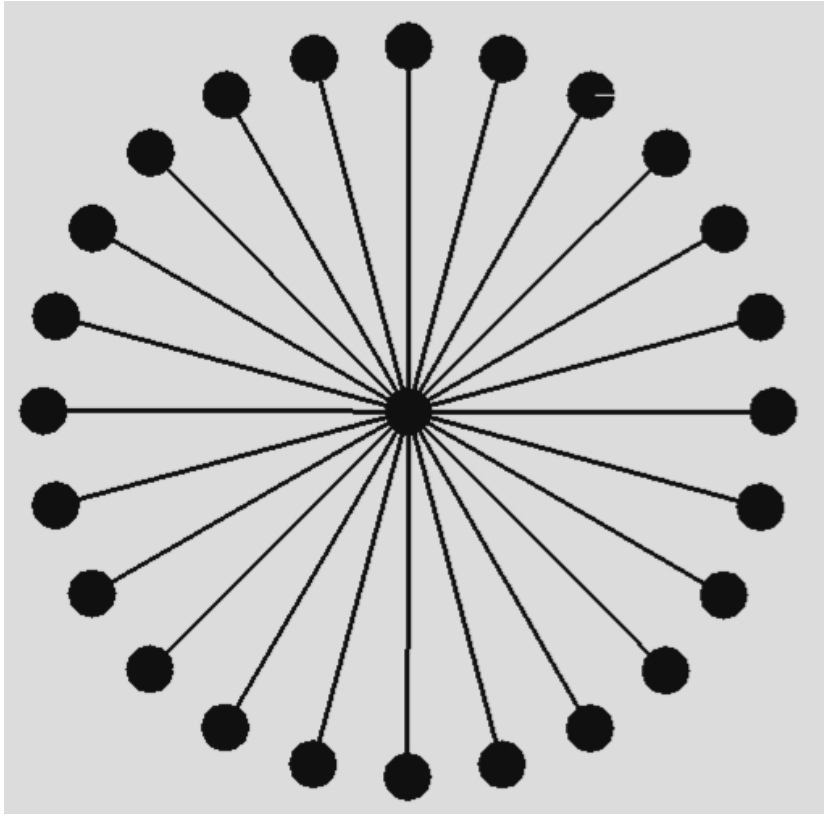


Number of layers = 11.

void createNet_RadialCircular(int n);
Press ‘4’

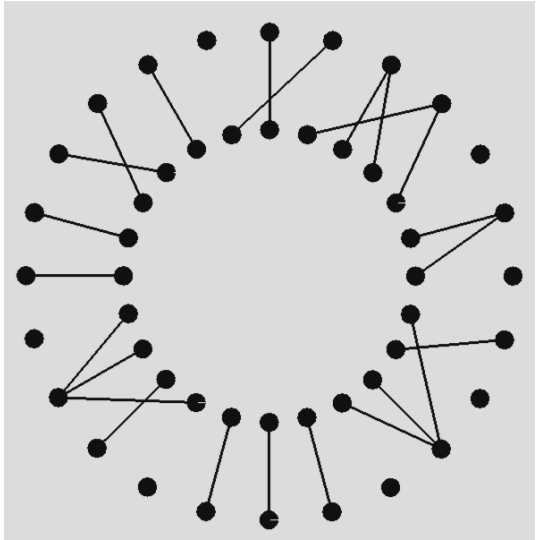
createNet_RadialCircular(24)

- See the figure and do the same☺



void createRandomGraph_DoubleCircles(int n); Press ‘5’

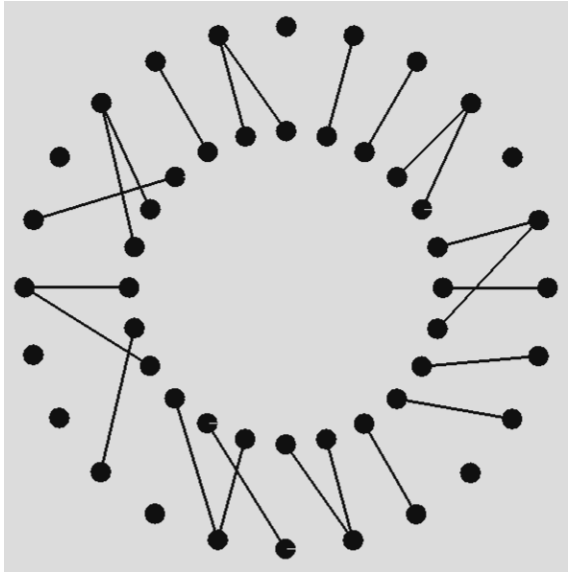
createRandomGraph_DoubleCircles(24)



There are two circles.

There are n nodes in each circle.

For each node of the inner circle, it connects to a close node in the outer circle.



A node of the outer circle is **close** to a node in the inner circle if the edge formed by these two nodes do not intersect the **inner circle**.

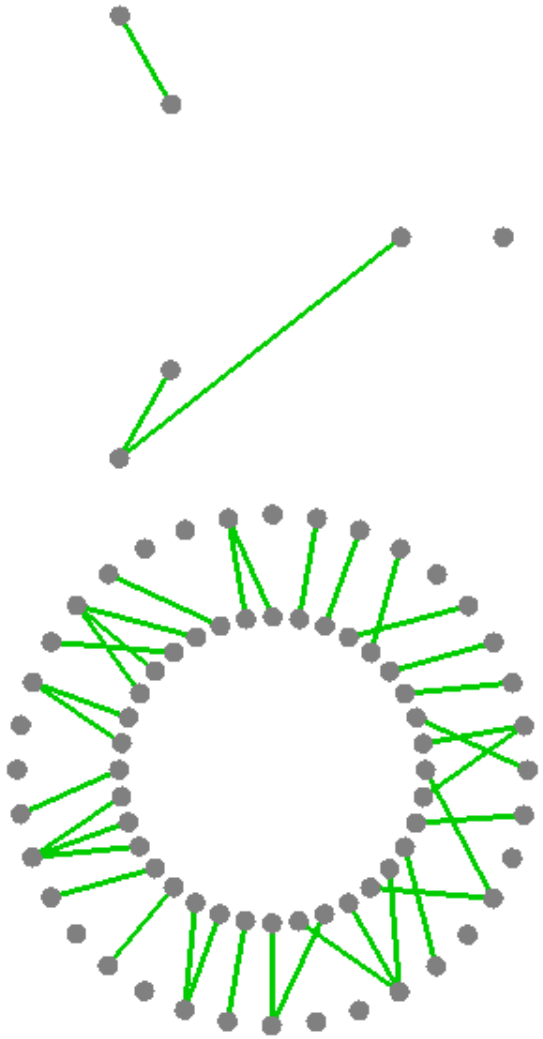
Different graphs can be created by pressing ‘5’.

Key usage for createRandomGraph_DoubleCircles

createRandomGraph_DoubleCircles(

mNumPoints_DoubleCircles

)



Press ‘<’ to decrease the number of nodes.

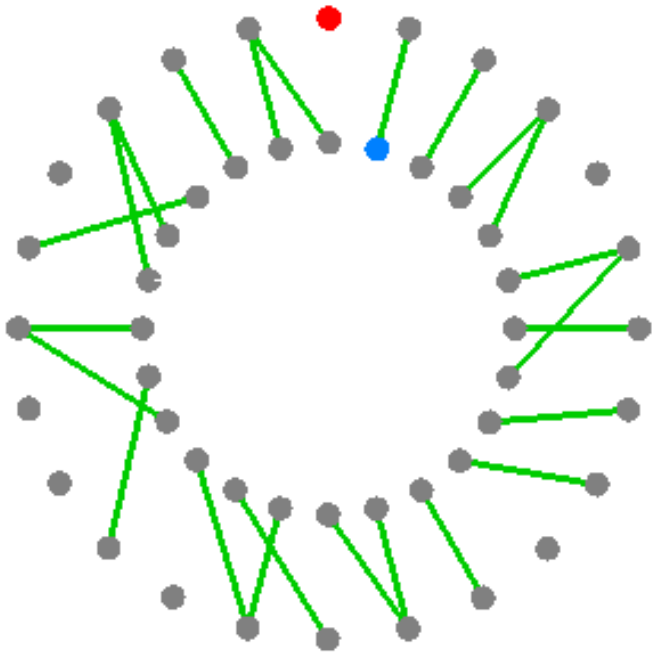
The smallest number of nodes of a circle is 3.

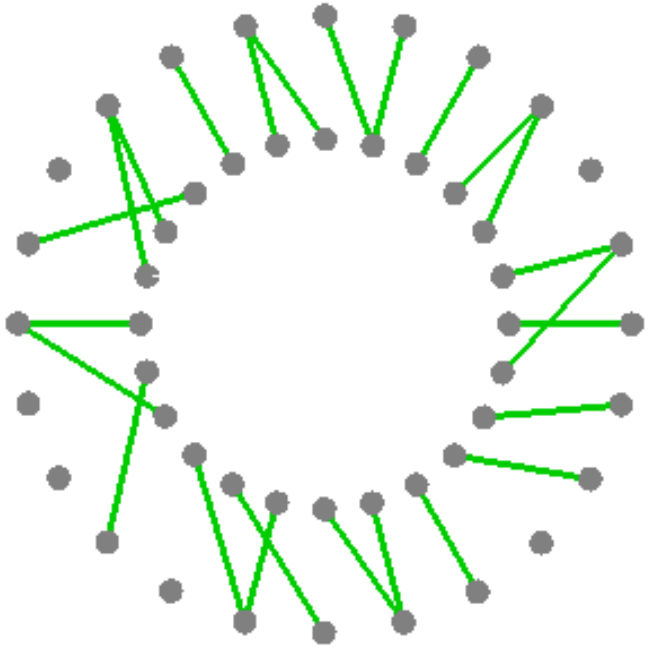
Press ‘>’ to increase the number of nodes.

The largest number of nodes of a circle is 36.

Tasks

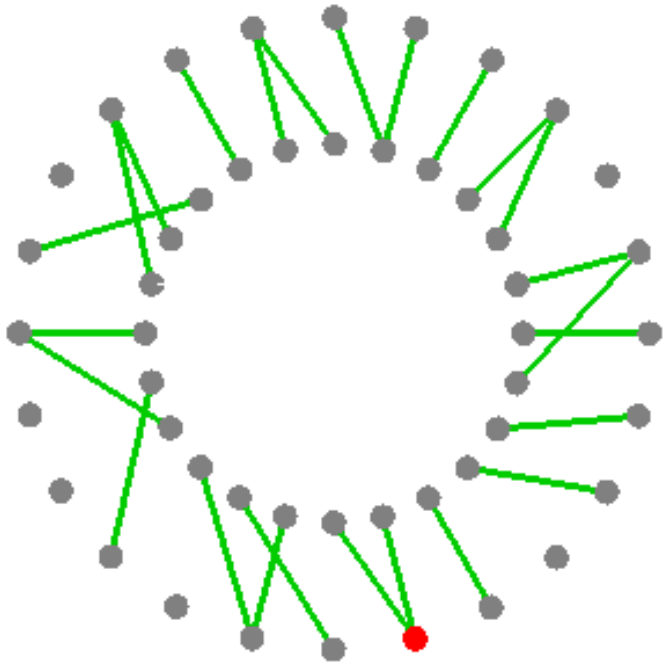
- Use the mouse to select two nodes and create an edge.
- Click the left mouse button to select a node.
- If the same node is selected, the node is unselected.

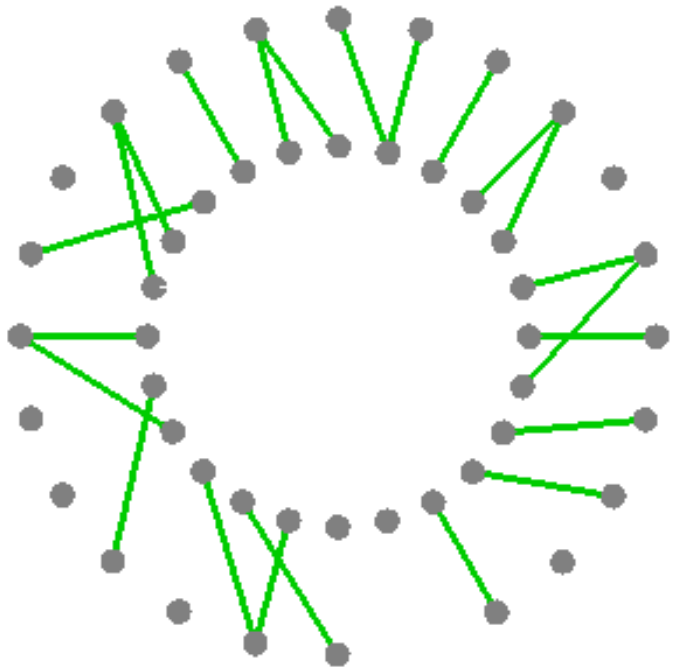




Tasks

- Use the mouse to select a node.
- Press DELETE to delete the node and all the edges attached at the node





Tasks

int getNumOfNodes( ) const; get the number of nodes N

void getNodeInfo(int nodeIndex, double &r, vector3 &p) const;

Return the node position (p) and radius of the node (r) with nodeindex.

The nodeIndex starts from 0 to (N-1).

int getNumOfEdges( ) const; get the number of edges

vector3 getNodePositionOfEdge(int edgeIndex, int nodeIndex) const;

return the node position of the node with nodeIndex of an edge with edgeIndex.

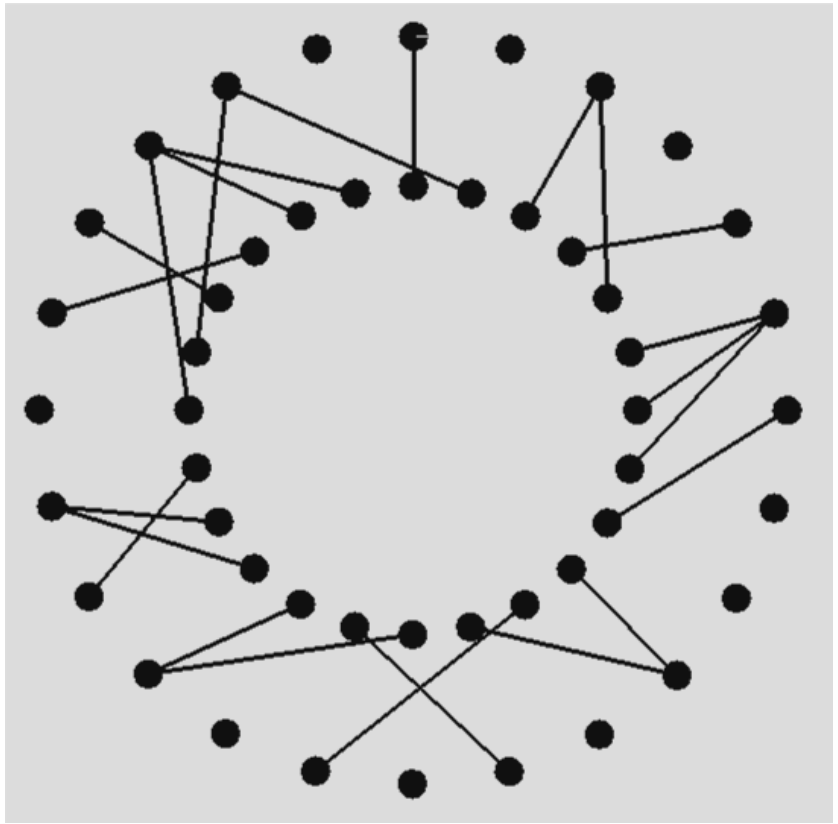
nodeIndex 0 for the first node of the edge

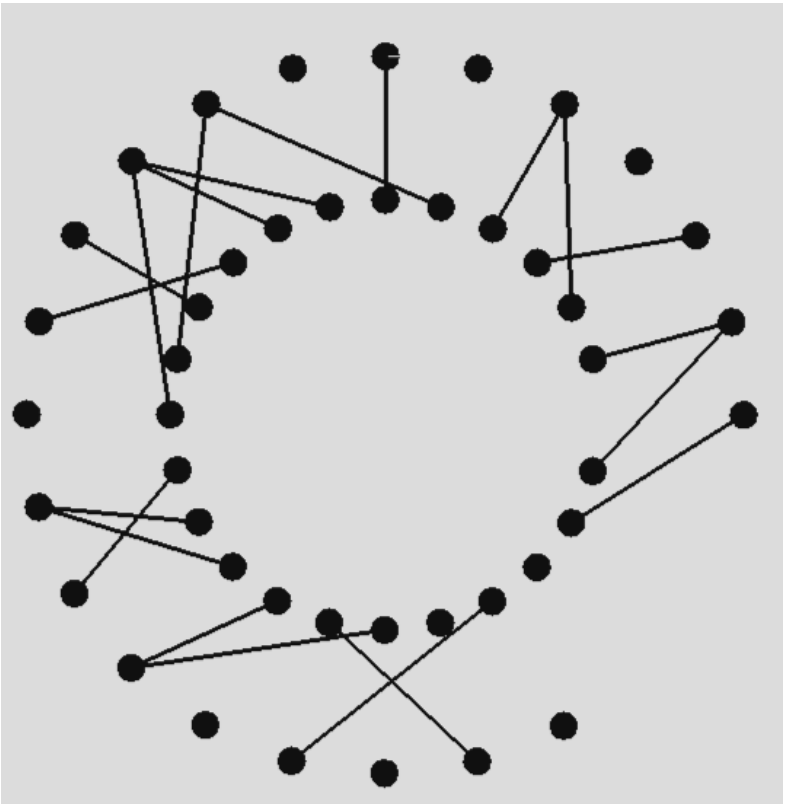
nodeIndex 1 for the second node of the edge

GRAPH_NODE *findNearestNode(double x, double z, double &cur_distance2) const

- Find the nearest node to the given position (x, y, z). Thus there's no y-coordinate in the parameter list.
- Note that we work in x-z plane only. y should be set to 0.
- The squared distance of the node and the given position should be stored in cur_distance2.

Press ‘d’ to toggle automatic deletion process for nodes. Delay with 250 msec.





Task: Modify the message if necessary

```
void GRAPH_SYSTEM::askForInput( )
{
    cout << "GRAPH_SYSTEM" << endl;
    cout << "Key usage:" << endl;
    cout << "1: create a default graph" << endl;
    cout << "2: create a graph with 10x10 nodes. Connect the consecutive nodes
horizontally" << endl;
    cout << "3: create a graph with 10x10 nodes. Connect the consecutive nodes
vertically" << endl;
    cout << "4: create a graph with 10x10 nodes. Create 10 randomly generated edges"
<< endl;
    cout << "5: create a graph with 10x10 nodes. Create 10 randomly generated edges
attached at a random node" << endl;
    cout << "Delete: delete a node and all the edges attached at it" << endl;
    cout << "Spacebar: unselect the selected node" << endl;
    cout << " " << endl;
    cout << "Use the mouse to select nodes and add edges" << endl;
    cout << "Click the left button to select/unselect or create an edge" << endl;
    cout << " " << endl;
    cout << "A selected node is highlighted as red." << endl;
```

Enjoy programming!