Homework description:

1. Comment by my own:

   I have tested the program several times, and it can run well. I usually create some objects with dimension (1,1,1) and translate within -3~3. But I think it should be ok if you test it with larger dimension or transformation. If you want to run a quick testing of my program, please use “Plan\testing only” first (see section 5.g for more detail). If you want to run an example that I have tested and found the path successfully, please see the two test cases at the end of this report.

2. Collision detection:

   To test the intersection (collision detection) of scene objects, I use the bounding box for both the block and the cylinder. i.e. some points which is out of the cylinder but contained in the bounding box, may considered as inside of the cylinder.

3. Build Road Map

   a. get the bounding box of the whole scene (a bounding box contains all scene object)
   b. In the range of this bounding box, throw 11*11*11 nodes to the scene uniformly for constructing a graph
   c. Test each node, if the node intersects with any scene object, then remove the node.
   d. Say we get n valid nodes left from the step c, connect each valid node to all other valid nodes, we get n*n candidate edge for this graph.
   e. Test each edge, if the edge intersects with any scene object, then remove this edge
   f. All the remained (collision free) nodes and edges construct a road map that we can further use for path planning

4. Path Planning

   a. Once we enter two endpoints (say one is start point, the other is end point)
b. The start endpoint search for its nearest nodes on the road map (say neighbor1)

c. If the path from “start endpoint” to “neighbor1” is collision free, then
   i. Dijkstra Algorithm is adopted to search for the shortest path:
      1) This algorithm can find the shortest path from a single source to all other destination nodes. Here we use the node “neighbor1” as the source.
      2) All visited node is marked as “visited”
      3) All visited node have been recorded its shortest distance and its parent node
   ii. The “end endpoint” search for its nearest nodes (say, neighbor2) which is marked as “visited”.
      1) If the path from “end endpoint” to “neighbor2” is collision free, then
         (1) trace back the parent of “neighbor1” (by the result of Dijkstra Algorithm), till reach “neighbor1”.
         (2) Return nodes on the result path (e.g. start endpoint → neighbor1 → n0 → n13 → n30 → n5 → neighbor2 → end endpoint)
         (3) Return result.
      2) Otherwise, check other nearest nodes
   iii. If no path is found from this neighbor1, back to step (b)

d. If no path found, then return null

5. Program Usage:
   a. First create objects (e.g. blocks, cylinders, groups, …, etc) as assignment4.
   b. Add objects to the scene by clicking “Plan\Edit Scene”.
   c. After finishing editing the scene, you have to click “Plan\Build Map” to build road map before enter endpoints
   d. (Optional) If you want to check the result of road map, click “View\RoadMap on/off”. The display window will first show all the graph nodes and then show the scene edges. Note: here, only a reduced graph will be rendered!
note1: Since there are too many edges built in my road map, the display window will only render parts of them (up to 30000 edges). Otherwise there will be an out of memory exception due to Java heap problem. However, those nodes and edges still exist in the database (just no display).

note2: no edges and paths close to the boundary of the bounding box of the scene will be rendered. Because if I render all those boundary edges, you can only see a box full of nodes and edges as the picture shown in Fig1. Thus I decided not to render those nodes and edges close to the boundary, then you can see a more clear result as shown in fig. 2 (Those edges and paths are still in the database, just nodisplay).

Fig1. all objects are covered by the edges. Fig2. No boundary edges/paths are rendered

e. (Optional) click “View\RoadMap on\off” again to remove the road map from the display window. You may want to do this before enter endpoints, otherwise the planned path may be covered by the road map.

f. Enter two endpoints by clicking “Plan\Enter Endpoints”, the result path will be shown on the display window as Fig3. If no path found, the message will printed on the command window.

Fig3. The endpoint will be rendered as yellow dots, and the path is purple line
g. (Debug only) There are two menu items that I added so that I can **debug/test the program easier**.

i. **“Plan\Testing only”**:  
   If you don’t want to create the scene object one by one to test the result, the function “testing” give you a quick test!
   
   1) Before you use it.  
      Make sure you haven’t created any object before using this function. If you did, please restart the program. Otherwise it may have error occur.
   
   2) click on **“Plan\Testing only”**, it will automatically create two blocks, union them, and add this group object to the scene. It will also build a road map and add two endpoints to the scene. The result will be shown on the display window. Click on “View\RoadMap on\off” to see the road map.
   
   3) After you use it.  
      Make sure you restart the program if you want to use other function after testing

ii. **“Plan\Clear Scene Object”**  
   You can remove all objects from the current scene by clicking “Plan\Clear Scene Object”. Those objects are still in the database and you can add them back to the scene by “Plan\Edit Scene” as usual.

iii. These two functions have not been tested carefully. So no guarantee that the program can still work well after you use these two. Just suggest that you restart the program after using **“Plan\Testing only”**
Test case 1:

1. Create a box A size=(1,1,1)
2. Move box A with the translation (1,0,0) and saved as B
3. Create Group U1 = A U B  …fig1
4. Move box A with the translation (0.5,0,0) and saved as C
5. Create Group U2 = U1 – C …fig2
6. Move group U2 with the translation (0,2,0) and saved as U3 …fig3
7. Edit scene: add U2 and U3 to the scene
8. Click Plan\BuildMap, once complete, click View\RoadMap  … fig4

Note2: (Remember to click on View\RoadMap again before process following operation, otherwise the result path will be covered by all the graph edges!)

9. Enter endpoints (-1, 0.1, 0) and (3, 3, 0)  … fig5
10 Enter endpoints (-1, 0.1, 0) and (0.5, 0.3, 0)  … fig6

(Fig1. U1= A U B  Fig2. U2 = U1- C  Fig3. U3 = moved U2 up
Fig.4 Show road map  Fig.5&6. Path results, yellow points are endpoints
(Since U2 = U1- C, the middle part of U2 is eliminated)
Test case 2:
1. Create a box A size = (1, 0.2, 1)
2. Rotate box A 90 along z axis and saved as B
3. Create Group U1 = A U B …fig1
4. Move group U1 with the translation (1,1,1) and saved as U2 …fig2
5. Create a cylinder C1 (radius = 0.3, height = 1) and translate (1,1,1)
6. Create Group U3 = U2 – C1 …fig3
7. Create U4: Rotate U1 (90 degree) along z-axis, and translate(-1.5,1.5,0)…fig4
8. Edit scene: add U1, U3 and U4 to the scene
9. Click Plan\BuildMap, once complete, click View\RoadMap … fig5

note2: (Remember to click on View\RoadMap again before process following operation, otherwise the result path will be covered by all the graph edges!)

10 Enter endpoints (1.8,1.5, 1.5) and (0.3, 0.6, 1) … fig6
11 Enter endpoints (1.3,1.3, 1) and (-0.4, 0.5, 1) … fig7
Fig. 7. Path result from different view