**Path finding methods:**

**A Star Path Finding**

**RRT Path Finding**

A close-up of a machine

Description automatically generated with low confidence

Slope

Slope

Bridge

LevelTwo

LevelOne

PlaneOne

Both methods work on the grid I generated. There are 3 grids in total. One for PlaneOne, one for LevelOne and one for LevelTwo. Each grid holds a 2d array of Nodes. The Node class has these attributes: x and y coordinates, the center position, a boolean that indicates if there is obstacle in that node, a parent, Also, f value and g value that is going to be used in A Star Path Finding.

**A Star Path Finding:**

The algorithm is implemented exactly like what was showed in class, but since there are 3 separate grids. I add some methods to connect the path when needed.

For example, if the NPC wants to go to LevelOne from PlaneOne, we first find the first part of the path by using A star to find the path from the starting position to the bridge entrance on the PlaneOne side. Then add the positions on the bridge to the path. Then we use A star to find the path from the bridge exit (which is on LevelOne) to the destination.

It is a little different if NPC wants to go to LevelTwo from PlaneOne, we want to use the slope that is closer to the NPC. The rest is the same as above

**RRT Path Finding:**  
While the destination is not in the tree, I generate a random point on the plane, if the Node that this point belongs to has an obstacle on it, this point would be invalid. If no point in the existing tree can connect a straight line between the random point without crossing any obstacle, the point would be invalid. Otherwise, add the point to the nearest point in the tree that can be connected to it with a straight line without crossing any obstacle. In each iteration, check if the destination point can be added to the tree.

The connection between different planes is implemented same as the methods mentioned in A Star Path Finding.

**How did I accommodate the transporters?**

Each NPC object has a bool attribute wantToTransport, if an NPC wants to travel from LevelOne to LevelTwo or the other way around, set wantToTransport to true. If an NPC wants to travel from the slope to LevelOne, set wantToTransport to true. If an NPC wants to travel from the bridge to LevelTwo, set wantToTransport to true. Only when the NPC’s wantToTransport is true, the transporter would work for this NPC.

**P.S. I didn’t implement the “Transporter can only transport up to 3 NPCs at a time” part. I just really can’t find a good way to implement this. I figured I should just say this right here to save your time, because I believe this is the only feature that I didn’t implement (At least I think so).**

**How did I deal with blocking?**

In the GeneratePath() method in NPC, it only finds a new destination when the destination attribute is set to Vector3.zero. The destination attribute is only set to Vector3.zero in these two situations: 1) The NPC arrives at the destination. 2)The NPC has been blocked for a while and cannot make any progress at all.