1. Arrange your space
   1. Plane
      1. Barriers
   2. Cube token for each agent
   3. Agent
      1. Though it isn't required that you make these children of the plane, it is easer to select them all as Navigation static
2. Select Plane and Barriers (Everything an agent can't go through)
   1. In the dropdown next to the word Static, select navigation static
   2. You'll be asked if you want the children of the object to be static as well and you do
3. Go to Menu / Window / Navigation
   1. Dock it behind the Inspector
4. Select the button Bake
   1. Note this defines the agent size, max slope, etc.
   2. Press Bake, you'll see the paths along which an agent may pass
5. Select all your agents
   1. Add Component / Navigation / Agent
   2. Note parameters
      1. Set Speed to 10
6. Write the script:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.AI;

public class AgentControl : MonoBehaviour {

    public Transform home;

    NavMeshAgent agent;

    void Start () {

        agent = GetComponent<NavMeshAgent>();

        agent.SetDestination(home.position);

    }

}

* 1. Attach it to the Agents

1. Select each agent
   1. Drag the corresponding home into the home slot
2. Play the game.
3. Create an empty gameObject
   1. Name it Agent Groups
   2. Duplicate your agents
   3. Drag the duplicates into the empty game object
   4. Duplicate AgentGroups 10 times (this will give you 44 agents)
   5. Play the game.

**Dynamic Obstacles**

1. Create the Obstacle
   1. Create a cube
      1. Name it NM Obs 1
      2. Position it so that it blocks a popular path
      3. AddComponent / Navigation / Nav Mesh Obstacle
   2. Play the game
      1. Note they hesitate to go through, but try, and often do.
      2. Select the NM Obs 1 and select
   3. Select tha Nav Mesh window
      1. Click the Carve checkbox
      2. Sometimes you'll want objects to avoid a moving target (like the FPC). I that case, it is worth it to have them recalculate constantly). Other times it is more efficient to let Unity change the shape of the nav mesh dynamically. That is what carve does. When an obstacle has that setting, then if it is near the surface, it dynamically carves a hole into the nav mesh.
   4. Click the Carve checkbox
      1. Play the game.
      2. Go back to the nave mesh window
         1. Note the hole it has carved into the mesh
      3. Raise it up
         1. Note the hole goes away
   5. Add the control script
      1. Create a script called Obs\_1\_Control
      2. Add the code:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Obs\_1\_Control : MonoBehaviour {

    bool open = false;

    void Update () {

        if(Input.GetKeyDown("space")){

            if(open){

                transform.Translate(0, -4, 0);

            } else {

                transform.Translate(0, 4, 0);

            }

            open = !open;

        }

    }

}

* 1. Play the game, opening and closing the obstical to see how the agents respond

**Letting Agents Wander**

1. Tag the "homes"
   1. In the Inspector select the Tag dropDown
      1. Select Add Tag
      2. Click the Plus button
      3. Add the string "NavTarget" -- note capitalization is critical.
   2. Select all the "home" game objects
      1. Select the Tag dropdown and tag them all NavTarget
   3. Create a new C# script
      1. Name it AgentControl\_2
      2. Add the code

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.AI;

public class AgentControl\_2 : MonoBehaviour {

    NavMeshAgent agent;

    GameObject[] targets;

    Vector3 target;

    void Start () {

        agent = GetComponent<NavMeshAgent>();

        targets = GameObject.FindGameObjectsWithTag("NavTarget");

        GetTarget();

    }

    void Update () {

        if(Vector3.Distance(target, transform.position) < 3f){

            GetTarget();

        }

    }

    void GetTarget(){

        target = targets[Random.Range(0, targets.Length)].transform.position;

        agent.SetDestination(target);

    }

}

* 1. Replace the existing agent Control scripts with the new one
  2. Play the game.