UNIVERSITÀ DEGLI STUDI
DI MILANO

Navigation Meshes

A.I. for Video Games

Dario Maggiorini (dario@di.unimi.it)

Navigation Meshes

- Are a widely used methodology
  - … and it is kind of cheating for you, because Unity will take care of everything (please, do not propose a project where NM are the main focus)

- We already asked our level designer for help
- The game designer must describe regions and how they are connected anyway
  - This is a lengthy process !
- But … the game itself is made of polygons!

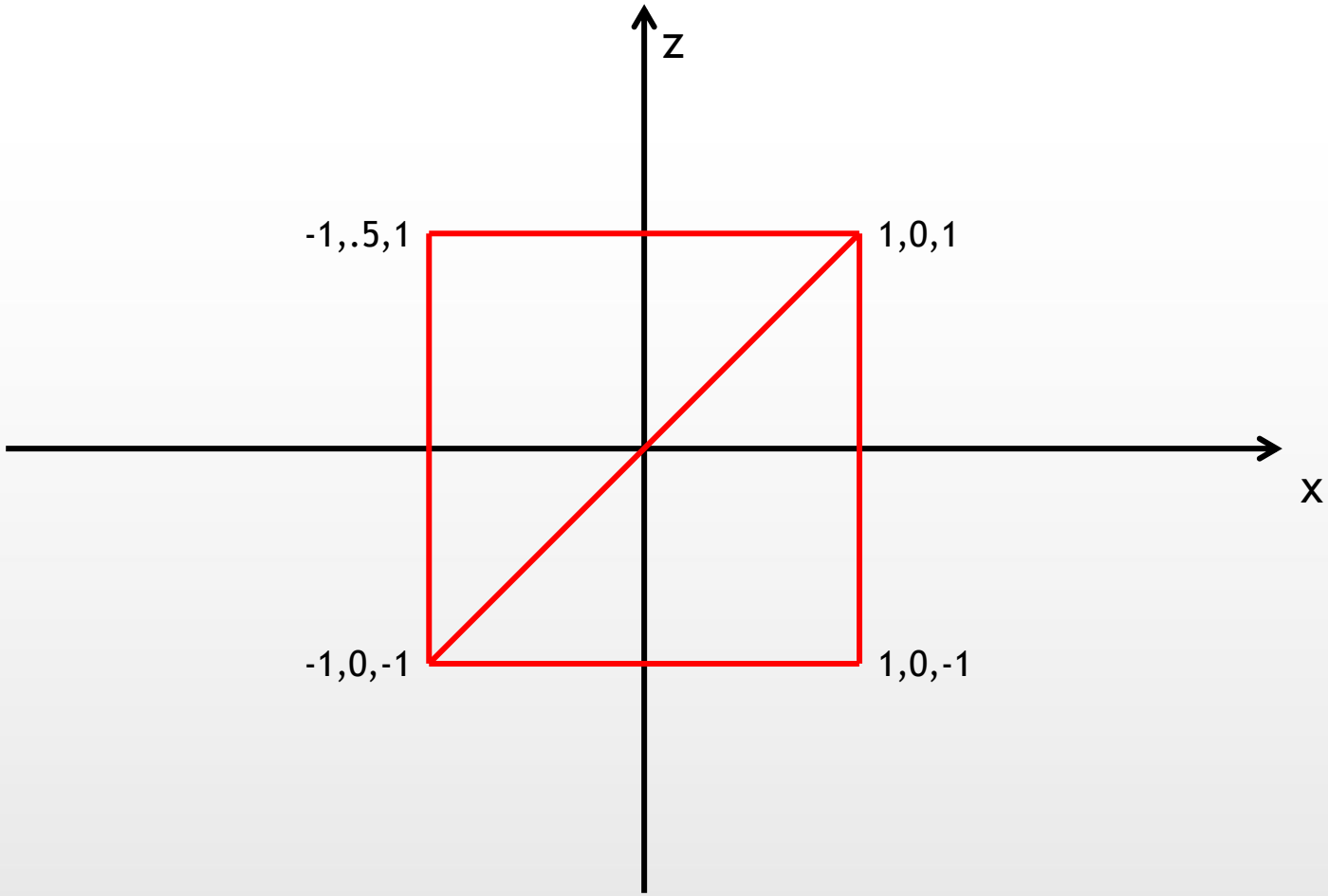
- … let's leverage on the graphical structure as the foundation of our pathfinding representation

What is a Mesh?

- A mesh is a data structure used to describe shapes inside a game engine

- In unity, Mesh is a class made (mainly) by:
  - A list of vertices in space
    - An array of *Vector3* objects
  - A sequence of triangles
    - An array of integers where every element is an index to use in the array of vertices

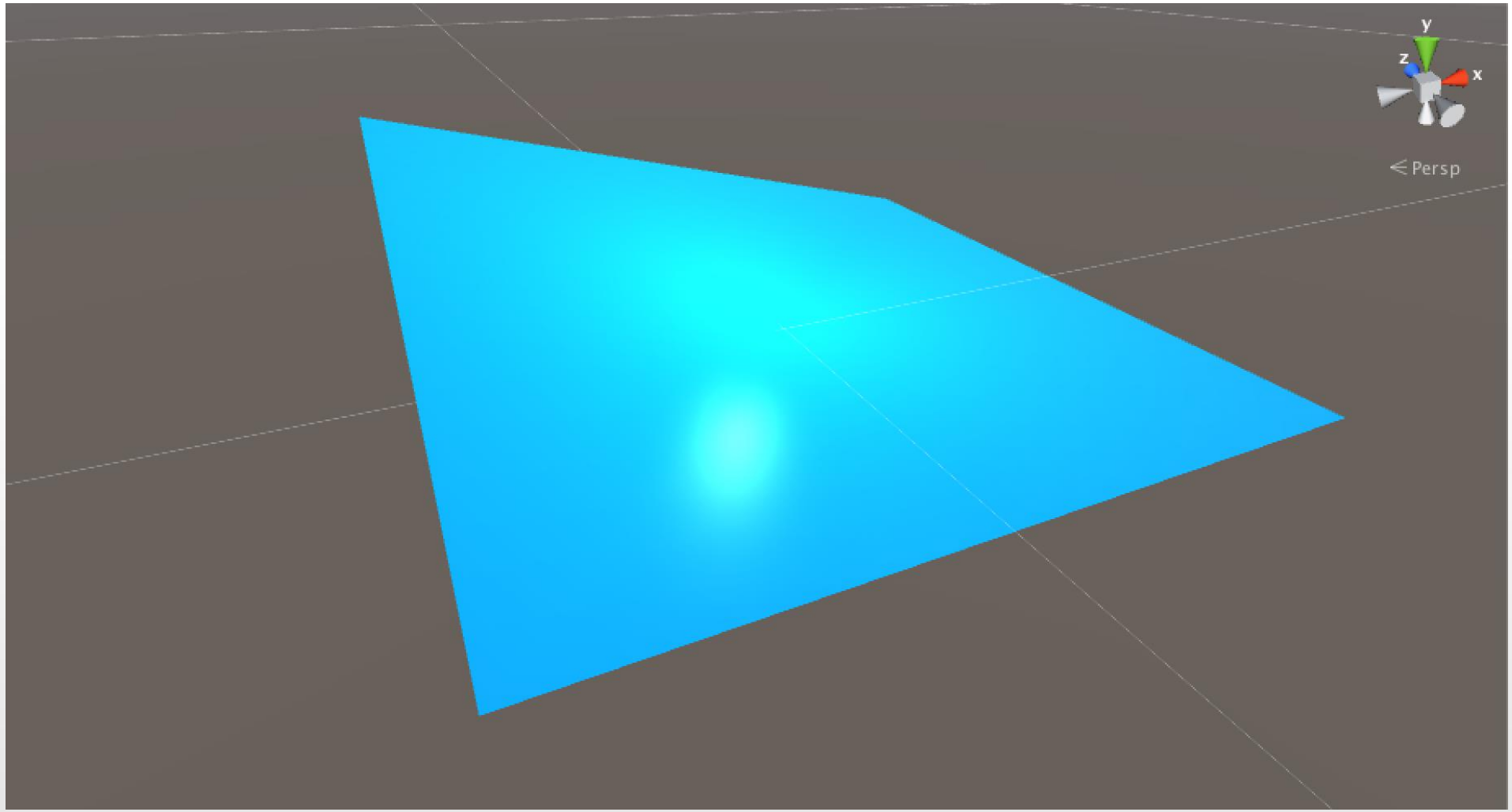
Mesh



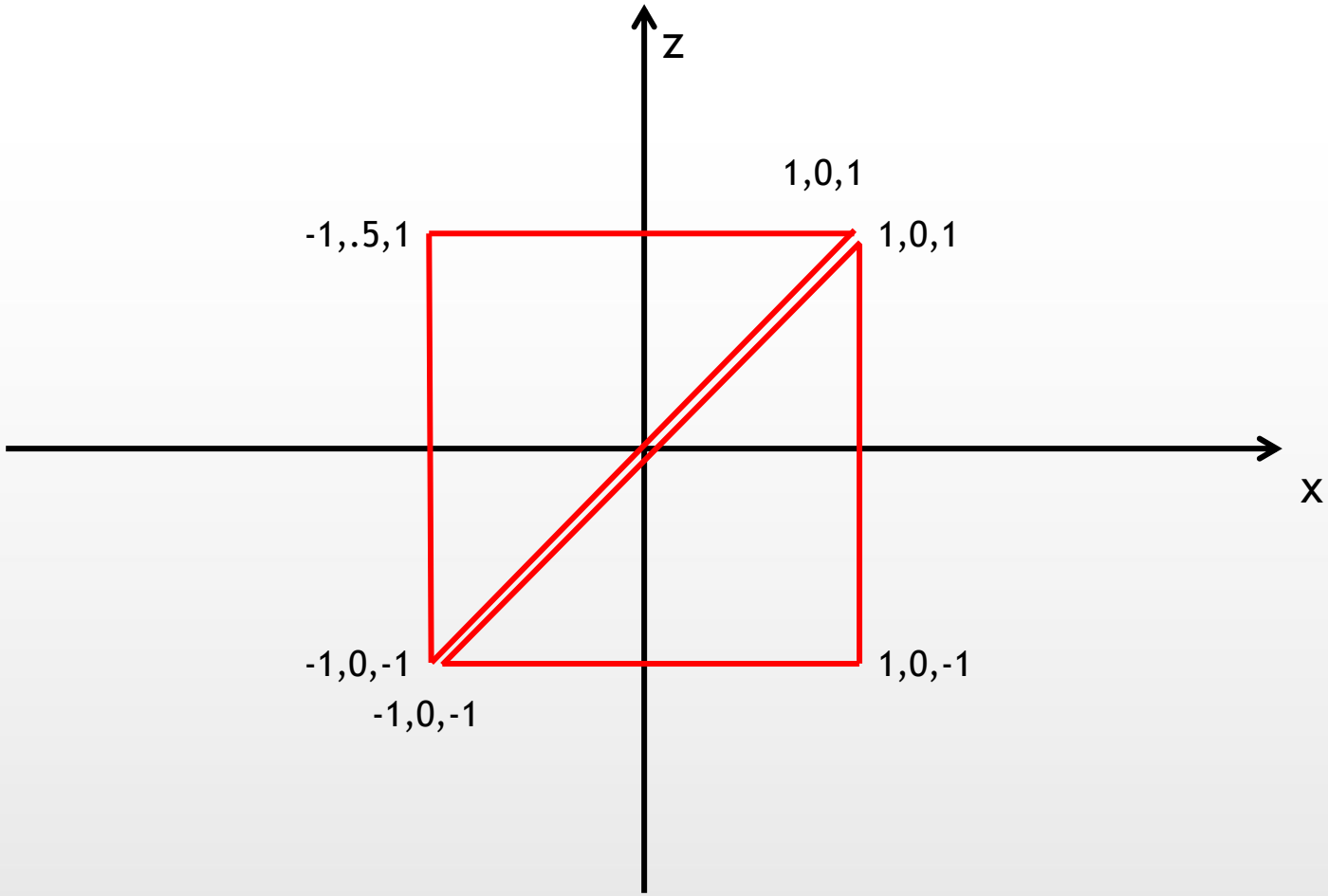
Example

```
void Start () {

    Vector3[] v = new Vector3[] {
        new Vector3 (1f, 0f, 1f),
        new Vector3 (1f, 0f, -1f),
        new Vector3 (-1f, 0f, -1f),
        new Vector3 (-1f, .5f, 1f) };

    int[] t = new int[] {
        0, 1, 2,
        0, 2, 3 };

    Mesh m = new Mesh();
    m.vertices = v; // MUST set this before assigning triangles
    m.triangles = t;
    // make material and lights work
    m.RecalculateNormals ();

    GetComponent<MeshFilter> ().mesh = m;
}
```

A Quad Made With Two Triangles



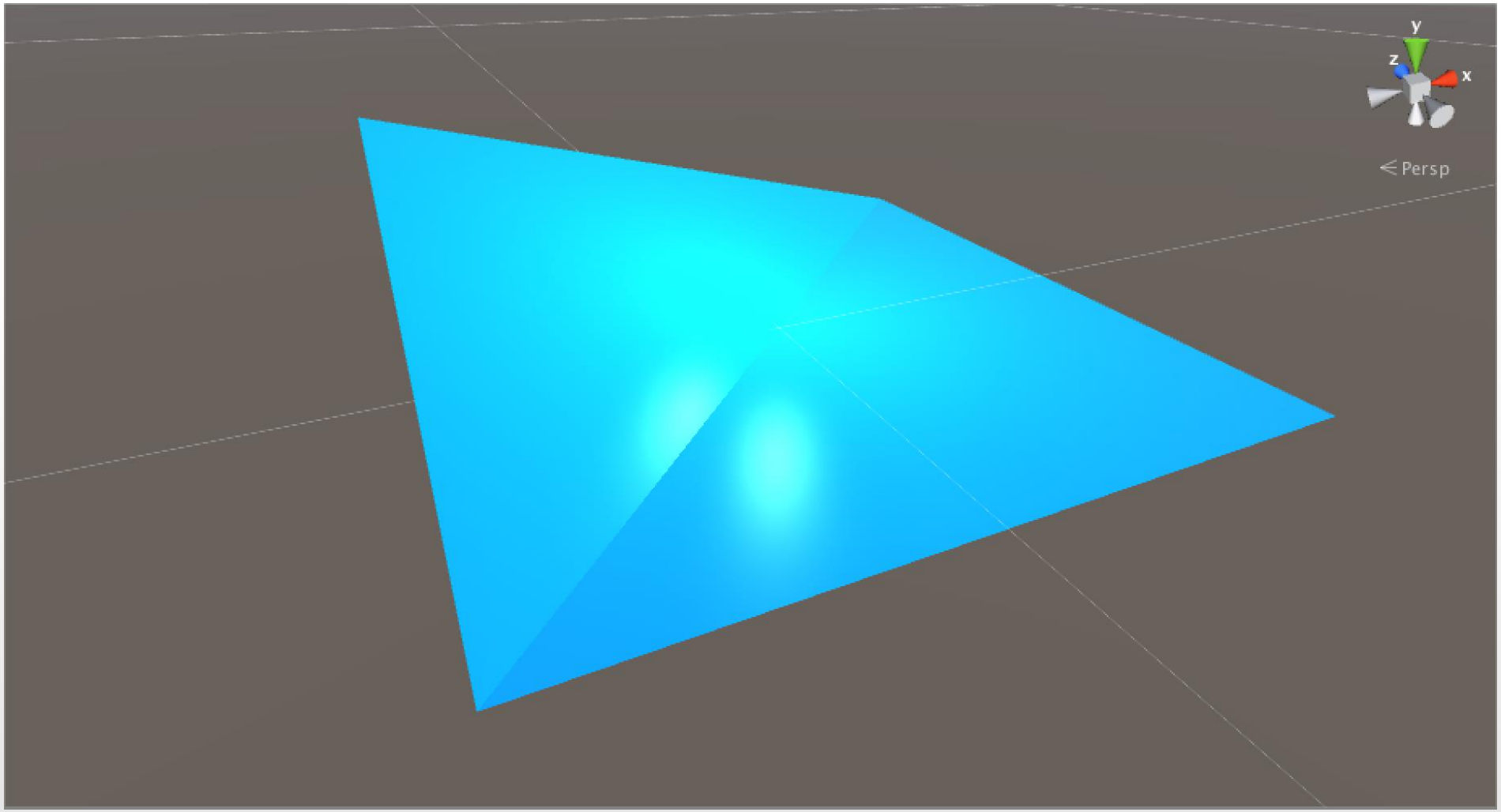
Mesh



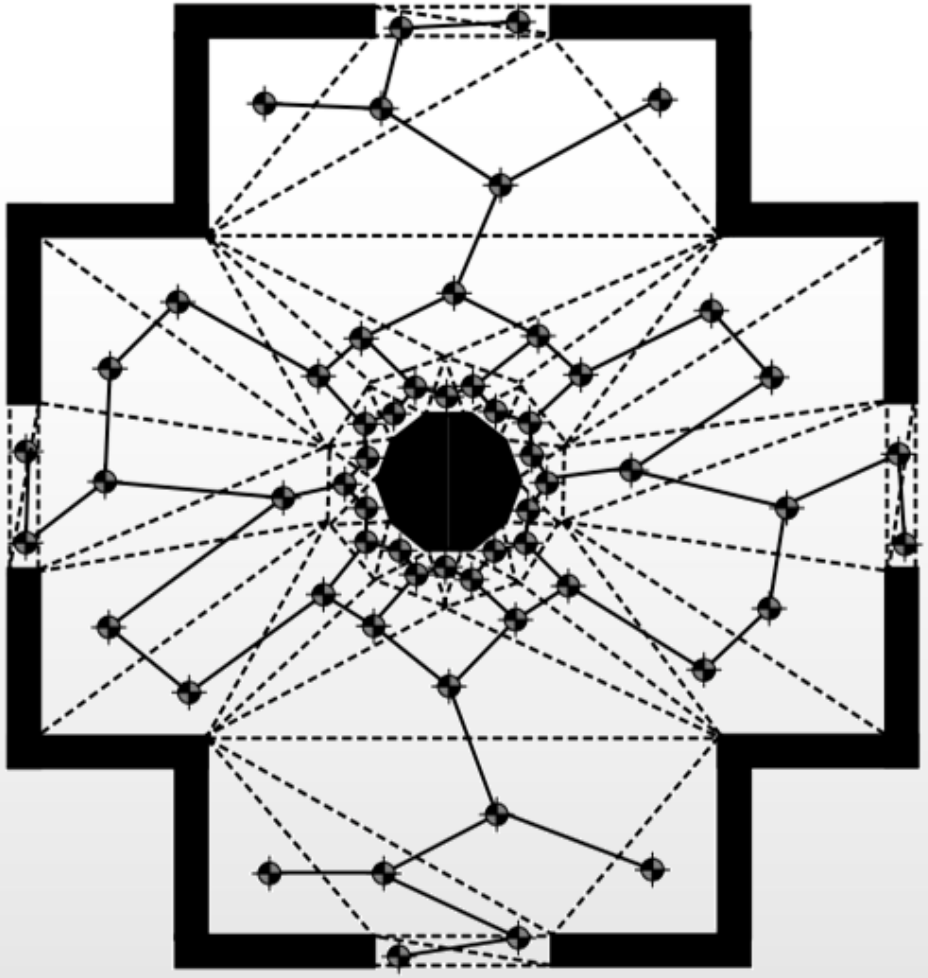
Other Example

```
void Start () {

    Vector3[] v = new Vector3[] {
        new Vector3 (1f, 0f, 1f),
        new Vector3 (1f, 0f, -1f),
        new Vector3 (-1f, 0f, -1f),
        new Vector3 (-1f, .5f, 1f),
        new Vector3 (1f, 0f, 1f),
        new Vector3 (-1f, 0f, -1f) };

    int[] t = new int[] {
        0, 1, 2,
        3, 4, 5 };

    Mesh m = new Mesh();
    m.vertices = v; // MUST set this before assigning triangles
    m.triangles = t;
    // make material and lights work
    m.RecalculateNormals ();

    GetComponent<MeshFilter> ().mesh = m;
}
```

Two Triangles



Navigation Meshes



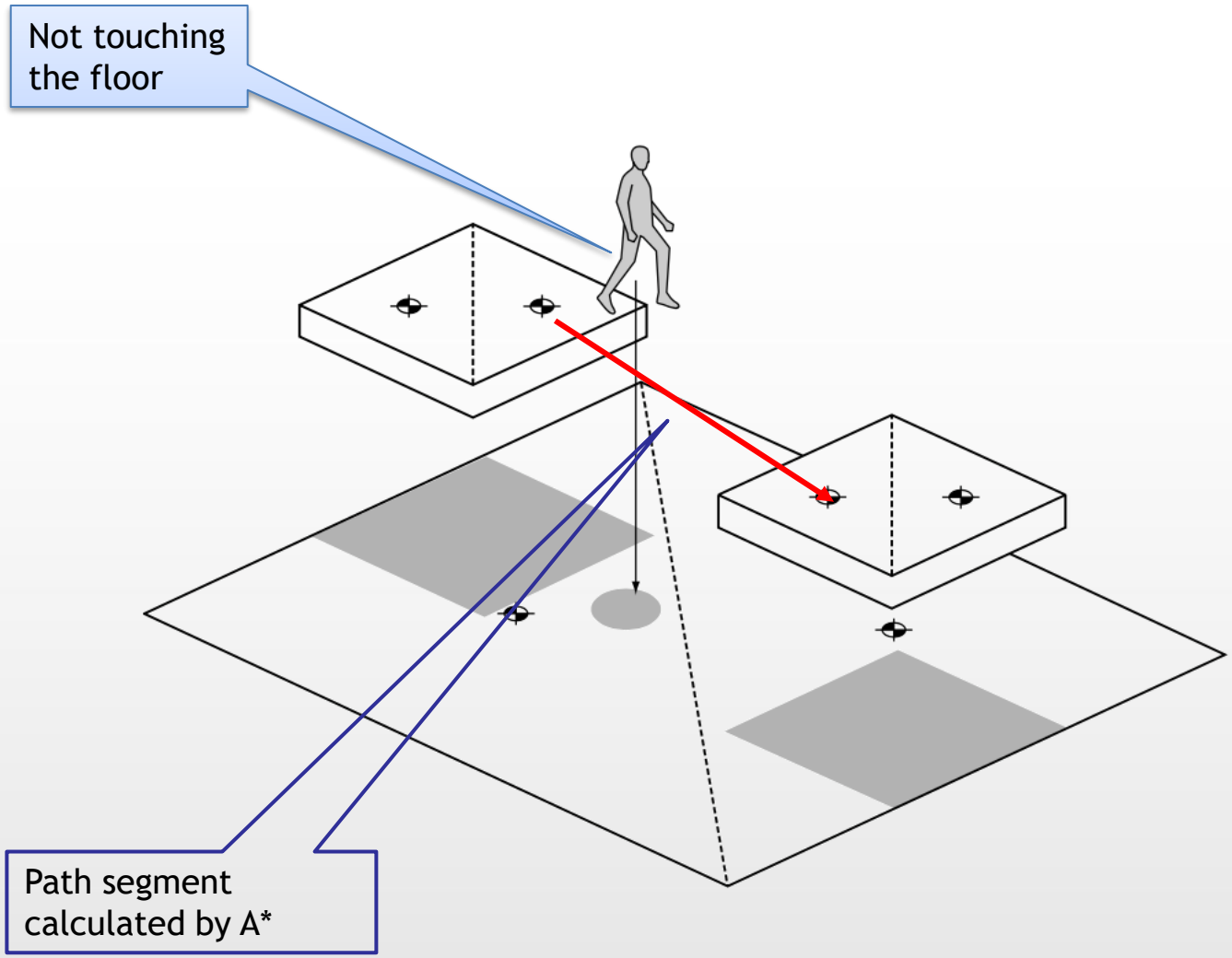
- We can use floor polygons to build a graph
  - Therefore, each nodes has at most N connections where N is the number of sides of each polygon (triangles → N = 3)
    - Optimizations can be performed starting from this assumption

Quantization and Localization

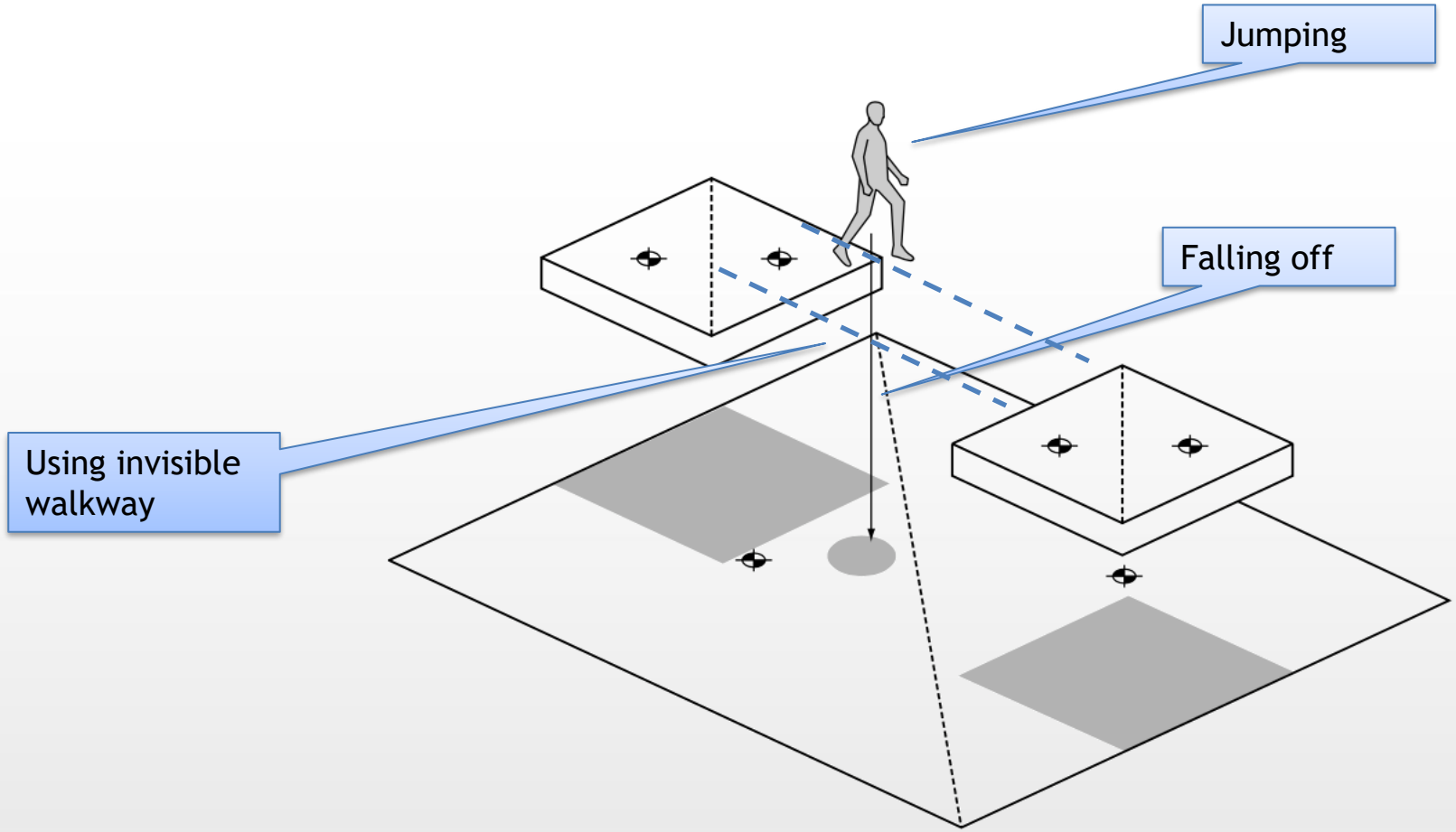
- Quantization
  - Each position is associated to the polygon containing it
    - We could end up searching a huge number of polygons
    - It is possible to make some assumptions to improve performances
  - Uses the coherence assumption
    - Like the “principle of locality”
    - If an NPC is moving, most probably it will move in a connected polygon. So, start checking there first
  - May be problematic with falls and jumps

- Localization
  - Any point in the polygon will do, the geometric center of a triangle is not a bad choice
    - Noob TIP: to obtain the center of a triangle we can just average the position of its vertices

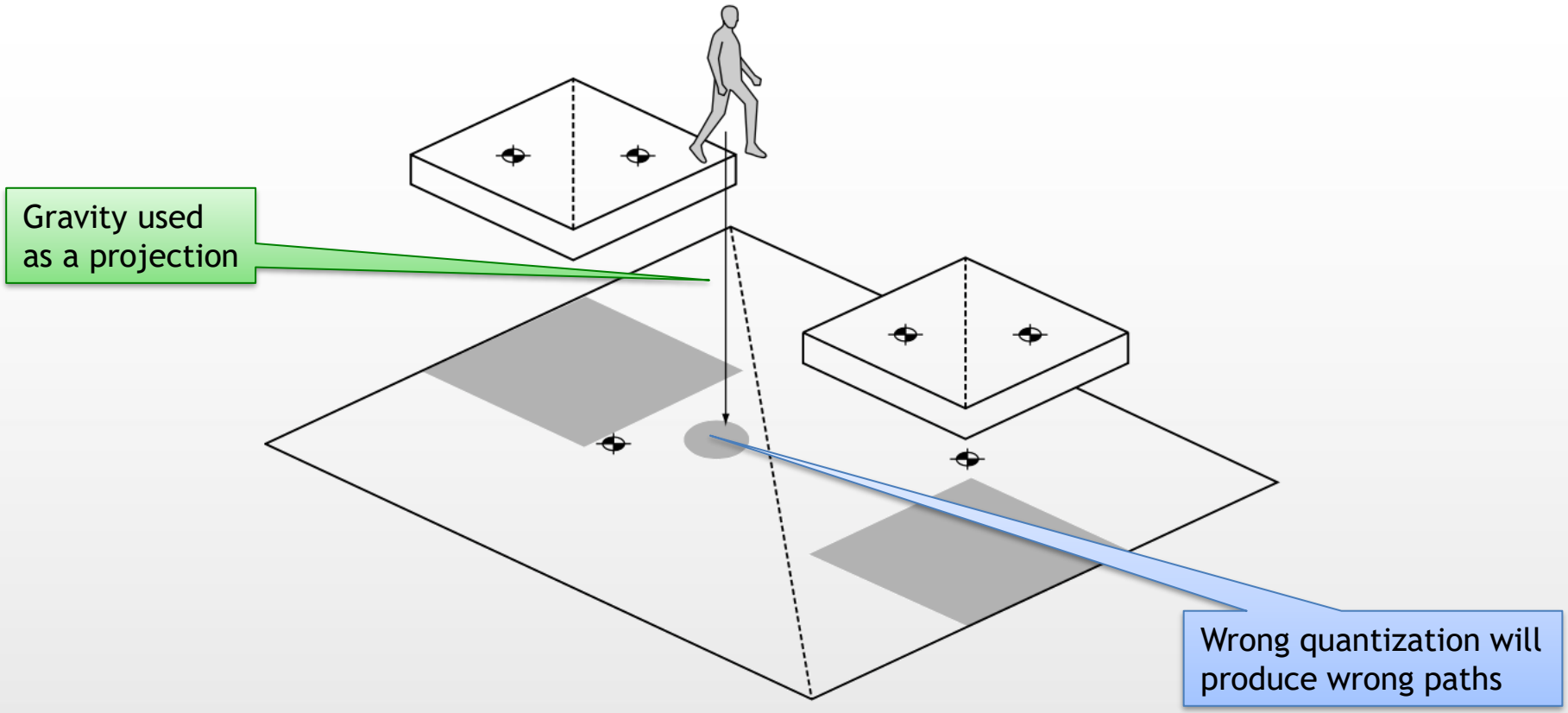
Problems in Quantization and Localization



Problems in Quantization and Localization

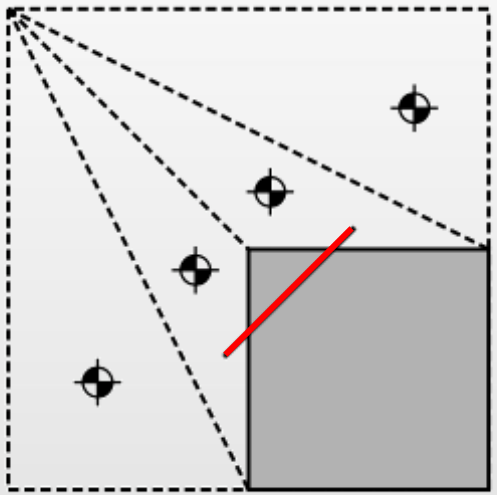


Problems in Quantization and Localization



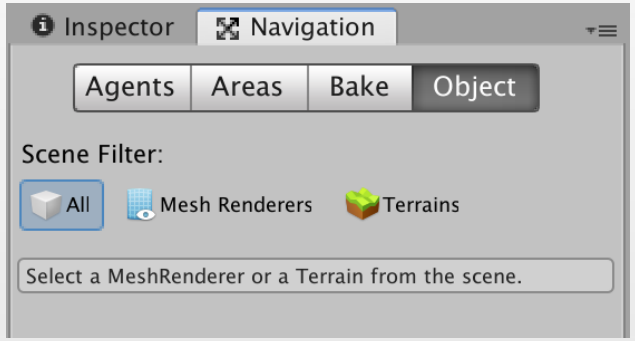
Validity

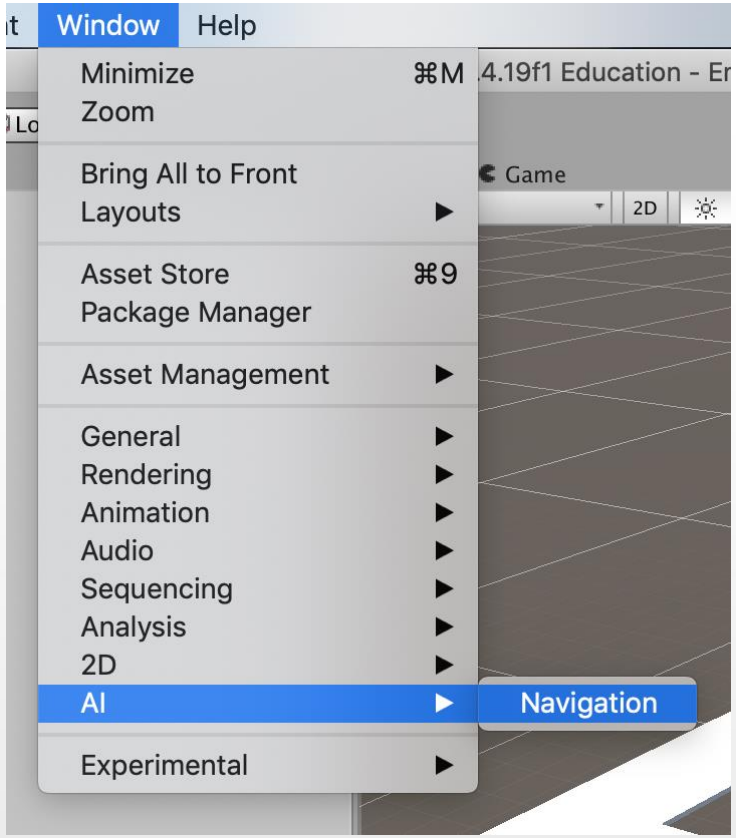
- May be problematic to evaluate
  - Because not all points in a polygon can move to any point of a connected polygon
- Not all floor can be used depending on context
  - E.g., the space under a table
- Size of NPC gets difficult to compute
  - Large NPCs must “stay far away” from walls



Navigation Meshes in Unity

1. Open the dedicated editor window
   - A new tab will pop up in the inspector panel

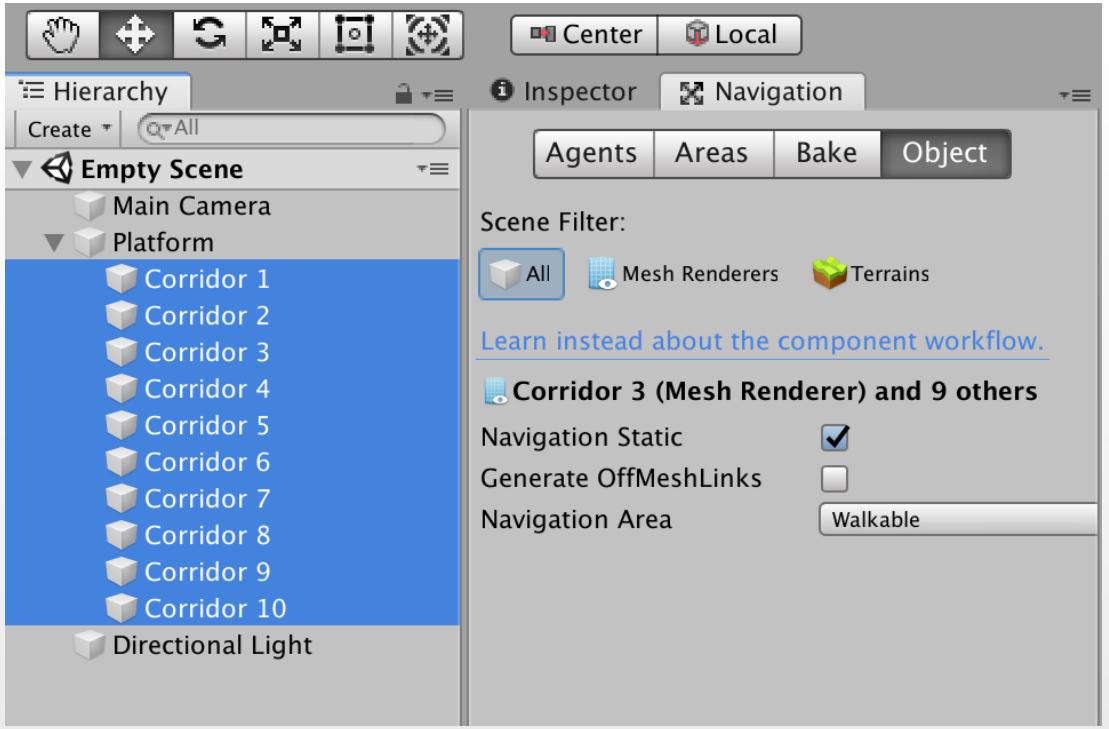




Navigation Meshes in Unity

2. Select all object that you want to be part of the navmesh and set them as “Navigation static”
    2. Try this on scene “Empty Scene” inside the Pathfinding/NavMeshes folder

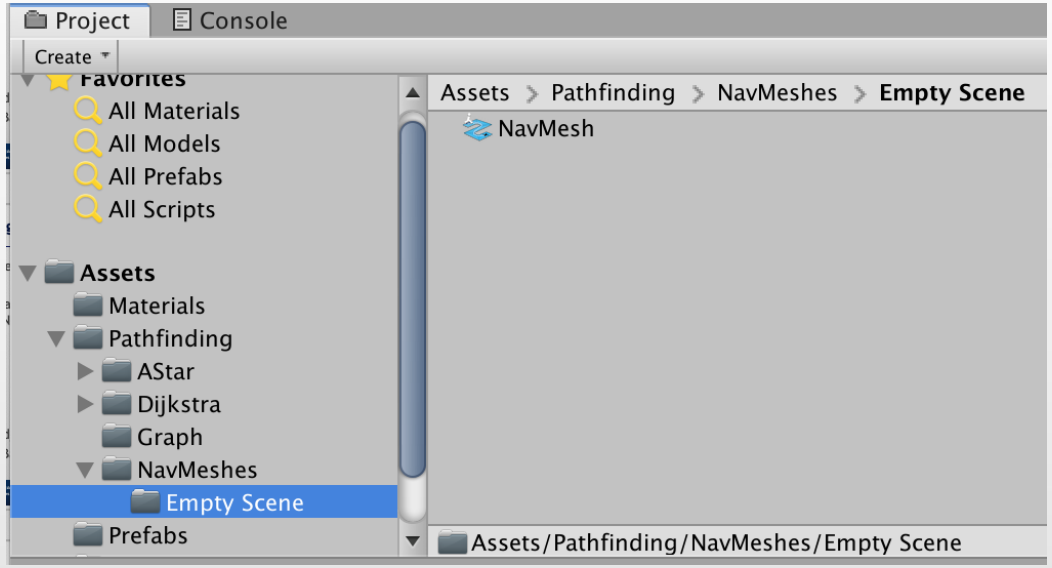
NOTE: you must select the single gameobjects holding the MeshRenderer components. Selecting the platform gameobject is not enough

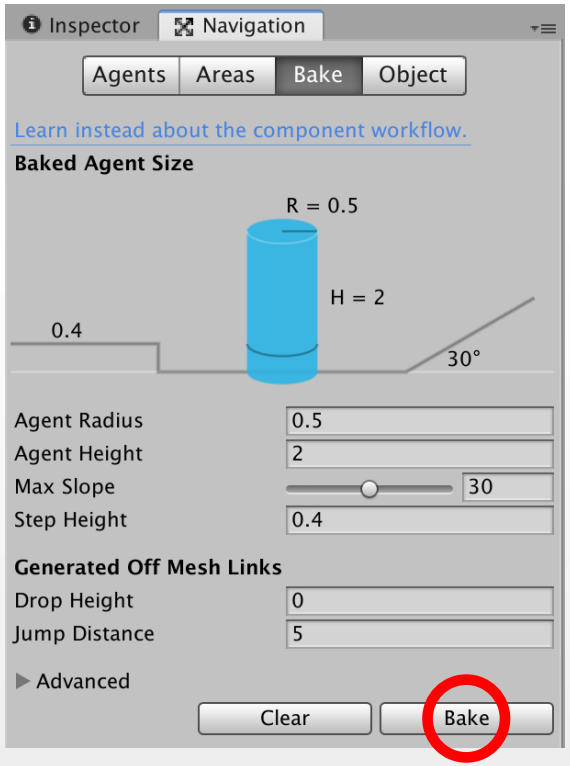


Navigation Meshes in Unity

3. Adjust agents' movement parameters

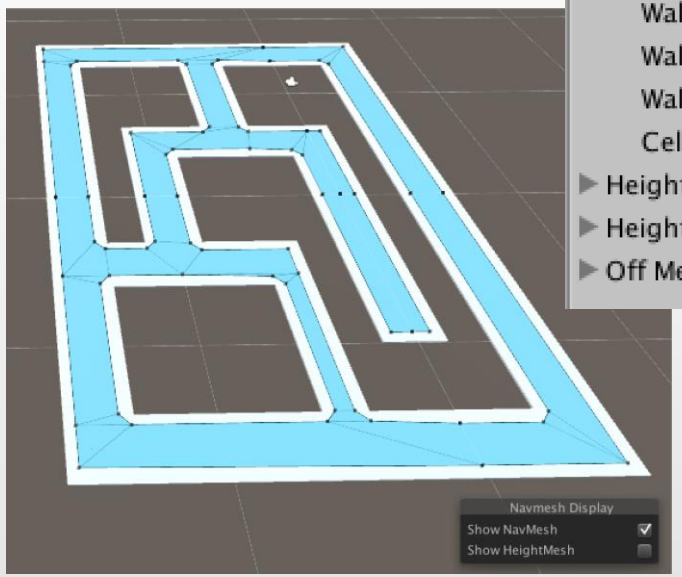
4. "Bake" to create a navigation mesh asset

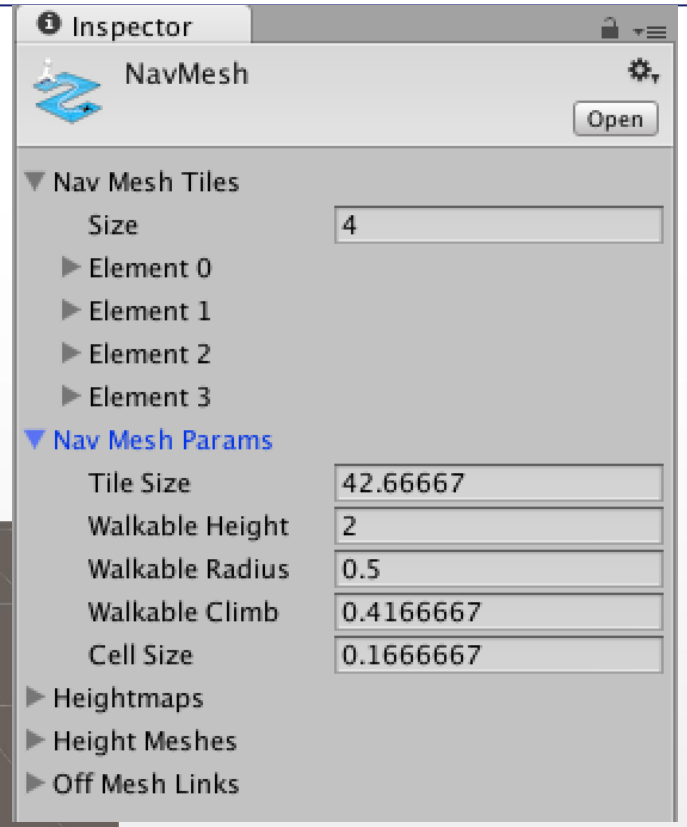




Navigation Meshes as Assets

- Baking will create a new asset that:
  1. Will be created in a subfolder with the same name as the scene
  2. Can be moved in a more convenient location
  3. Is linked to specific gameobjects in your scene



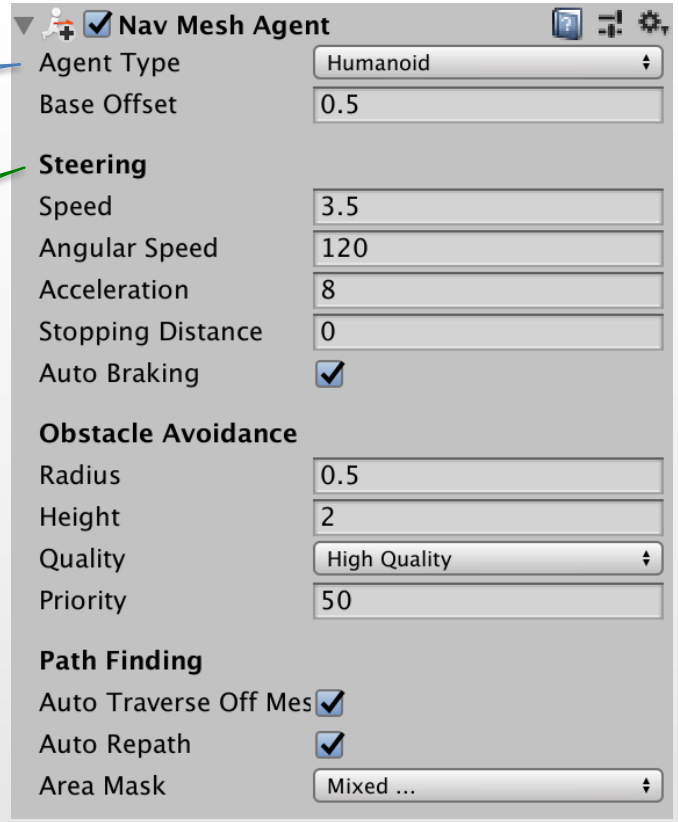


This is NOT Enough

- The navmesh alone is just a data structure
- You need to create an agent to walk it and make some sense out of your scene

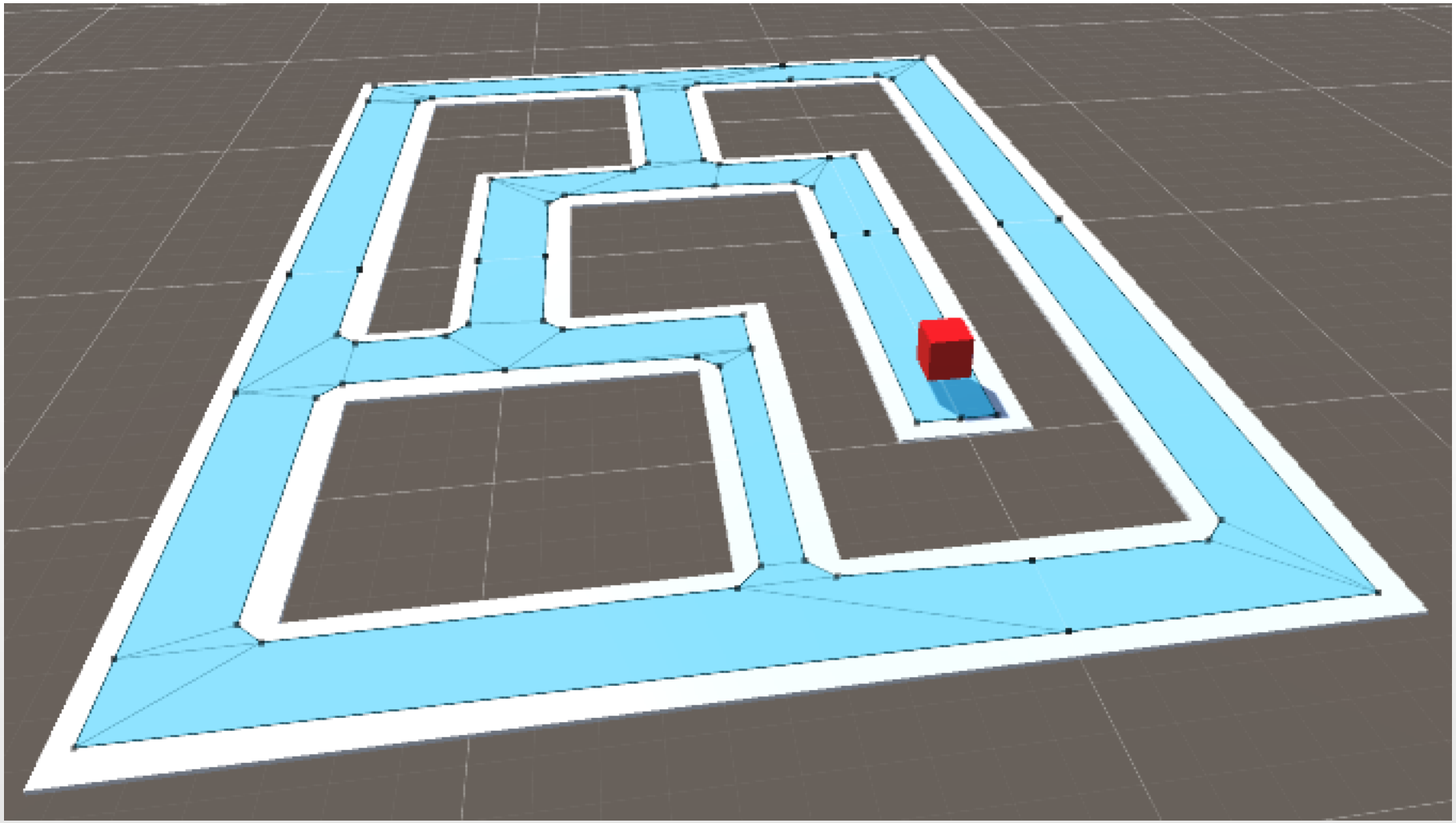
This is a navigation template. It ensures you are using the same physical parameters applied to the baking phase

WOW!
A FREE steering behaviour!



- Create a placeholder for your NPC and put inside a NavMeshAgent component
  - Yes! It is THIS easy :)

A Platform With a Navmesh and an Agent



Then, Give Directions to the Agent

Source: GoSomewhere
Folder: Pathfinding/NavMeshes

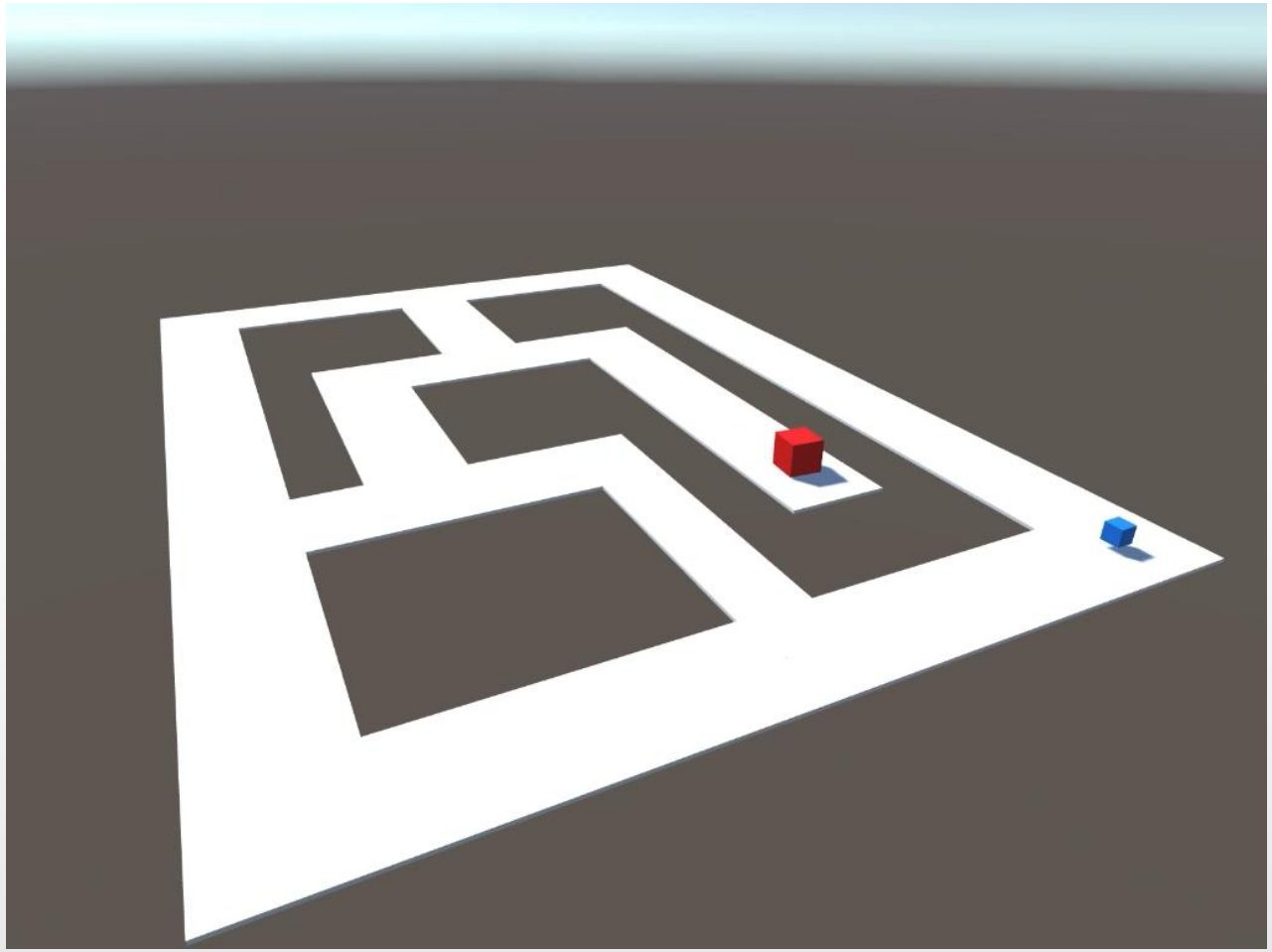
This will make sure we have a NavMeshAgent component available in the gameobject

```
[RequireComponent(typeof(NavMeshAgent))]

public class GoSomewhere : MonoBehaviour {

    public Transform destination;

    void Start () {
        GetComponent<NavMeshAgent>().destination = destination.position;
    }

}
```

The only think we must do is setting a destination to your NavMeshAgent ... and enjoy the show

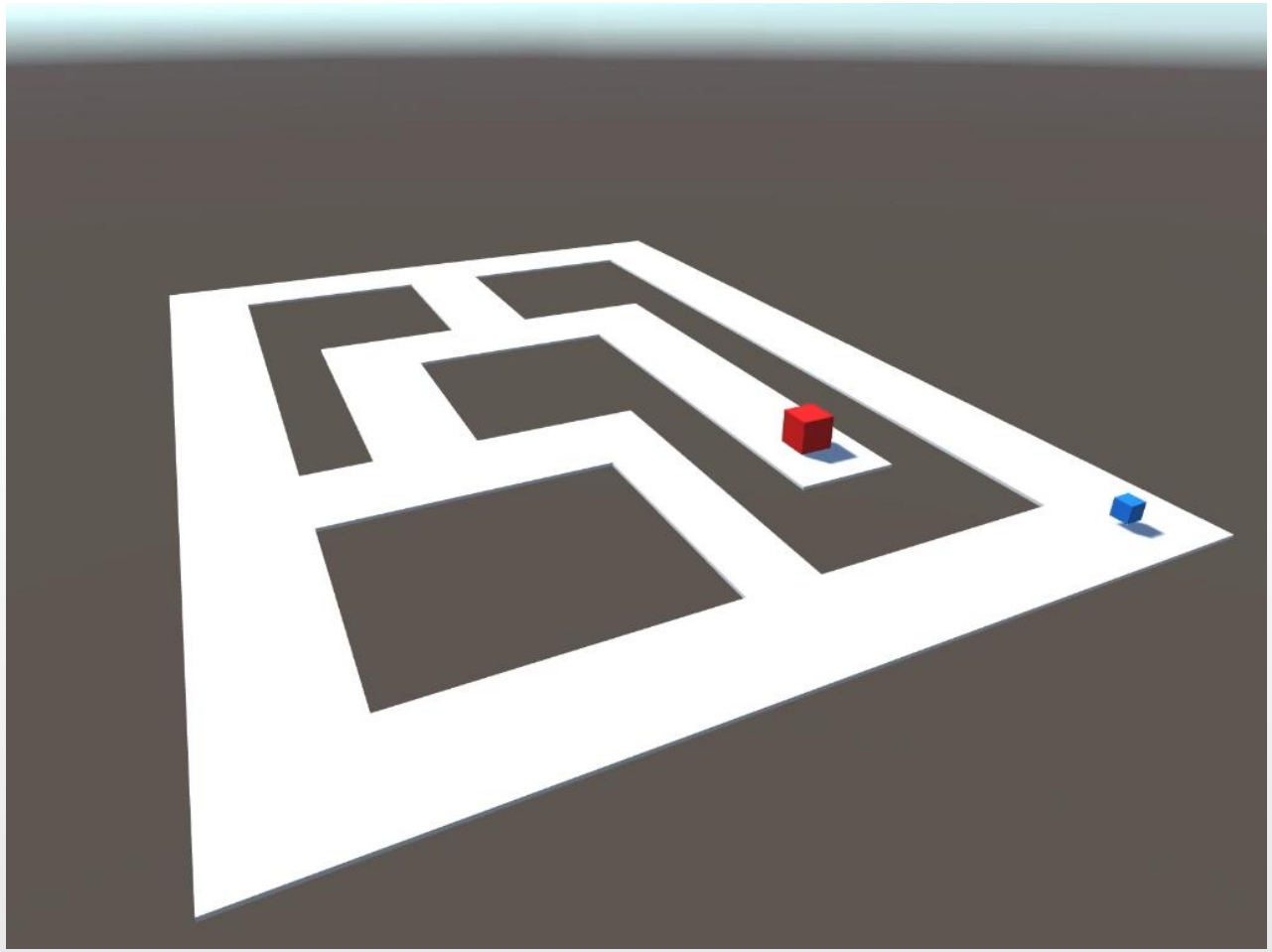
Let’s Have a Look

Scene: Go Somewhere
Folder: Pathfinding/NavMeshes



And Now ... Faster!

Speed set to 14 m/s



Of Course ...

- If we run too fast, we are going to bump into obstacles

... WAIT A SEC!

1. There is no rigidbody, we have no evolution rule providing “bumping”

2. We did not put any wall there! What the hell is the NPC bumping into?

Some Considerations About the NasMeshAgent

- NavMeshAgent will force a RigidBody behaviour on your agent
  - Not 100% real for what I am concerned. Nevertheless, it will ”sit” on the navigation mesh and push other rigidbodies and navmesh agents
- (Invisible) Walls will be inherited from the navmesh boundaries
  - The first turn in the last clip is a clear example for this
- Changing the environment on the fly is also supported
  - Bake the navmesh as NOT “navigation static”
- YOU are in charge to tune the steering behaviour with respect to your level outline

Easy Chasing Behaviour

- You “just” need to

  1. Put a player controller to move the target
  2. Re-calculate destination periodically on the agent

- Yes, once again it is this easy :)

Player Controller (Generic Script)

Source: PlayerController
Folder: Pathfinding/NavMeshes

Make sure there is a RigidBody component. If not, add one

```
[RequireComponent(typeof(Rigidbody))]

public class PlayerController : MonoBehaviour {

    [Range(0.0f, 20.0f)] public float movementSpeed = 4f;
    [Range(0.0f, 180.0f)] public float rotationSensitivity = 90f;

    void Start () {
        ;
    }

    void Update () {
        Rigidbody rb = GetComponent<Rigidbody> ();
        // gas and brake are converted into a translation forward/backward
        rb.MovePosition (transform.position
                         + transform.forward * movementSpeed * (Input.GetAxis ("Vertical") * Time.deltaTime));
        // steering is translated into a rotation
        rb.MoveRotation(Quaternion.Euler(0.0f, rotationSensitivity * (Input.GetAxis ("Horizontal") * Time.deltaTime), 0.0f)
                        * transform.rotation);
    }
}
```

Change position and/or orientation based on input (arrows from keyboard) or thumbsticks from gamepad. Unity will do the mapping for us

Chase the Player

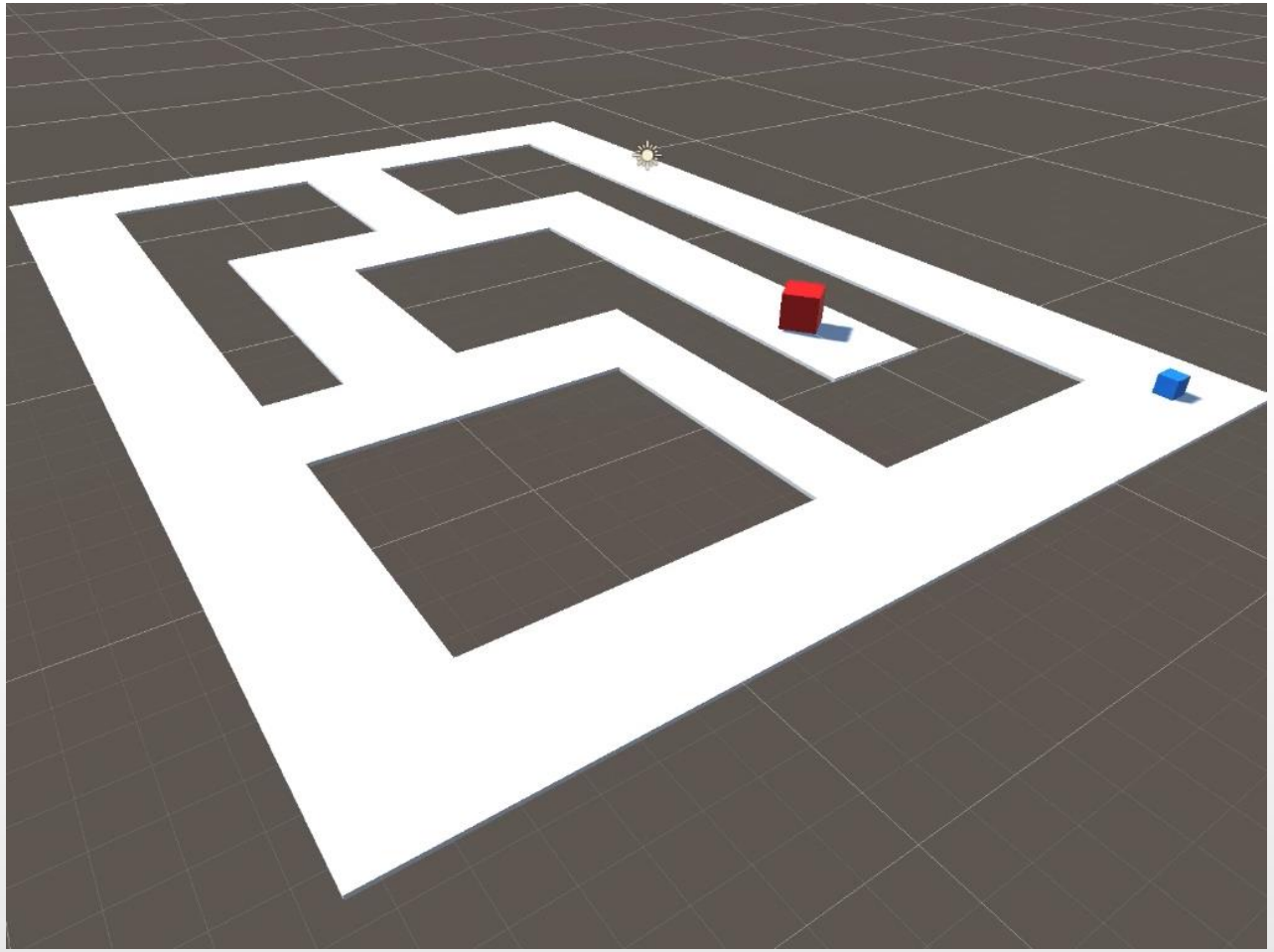
Source: ChaseSomething
Folder: Pathfinding/NavMeshes

```
[RequireComponent(typeof(NavMeshAgent))]

public class ChaseSomething : MonoBehaviour {

    public Transform destination;
    public float resampleTime = 5f;

    void Start () {
        StartCoroutine (GoChasing());
    }

    private IEnumerator GoChasing() {
        while (true) {
            GetComponent<NavMeshAgent> ().destination = destination.position;
            yield return new WaitForSeconds (resampleTime);
        }
    }

}
```

Set the destination every resampleTime second

Let’s Try It

Scene: Chase Player
Folder: Pathfinding/NavMeshes



Why ...

- My view is different?
  - You are looking into the game window. I took the recording from the editor window
- My blue cube is not moving?
  - Because you switched to the editor window after starting the simulation. Unity is not collecting inputs if the game panel.
    To solve the situation, move the game tab in another point of the interface so that it is not sharing the same panel as the editor
- The camera is chasing me?
  - The project is in third person perspective. The camera is attached to the player (see the object hierarchy)

Why ...

- The agent is stopping in some points?
  - Because the navmesh agent destination is set every 5 seconds. When the last sampled position is reached it will wait for the next timeslot to move again. To solve this, just reduce the value of ResampleTime in the Agent inspector
- Did I fall out of the platform?
  - Because there are no walls (remember?)
- The agent stopped when I fell?
  - Because you fell vertically, and it stopped at the closest point to your (planar) position

Chasing with Line of Sight

- The agent has a concept of “last known position” for the chasee

- The target position will get updated only if the chasee is in line of sight
  - This can be done periodically as well as when reaching the destination, this is for you to decide

Chase the Player with LOS

Source: ChaseWithLOS
Folder: Pathfinding/NavMeshes

```
private IEnumerator GoChasing() {
    while (true) {
        Vector3 ray = destination.position - transform.position;
        RaycastHit hit;
        if (Physics.Raycast (transform.position, ray, out hit)) {
            if (hit.transform == destination) {
                GetComponent<NavMeshAgent> ().destination = destination.position;
            }
        }
        yield return new WaitForSeconds (resampleTime);
    }
}
```

Define a ray from agent (chaser) to player (chasee)

This is an output parameter

This if will be verified when the ray it the destination instead of another object in the middle. I.e., there is line of sight between agent and player

This if will be verified when the ray, applied from the agent position will hit something

You can try this by yourself, running the "Chase With LOS scene" (there will be walls!)

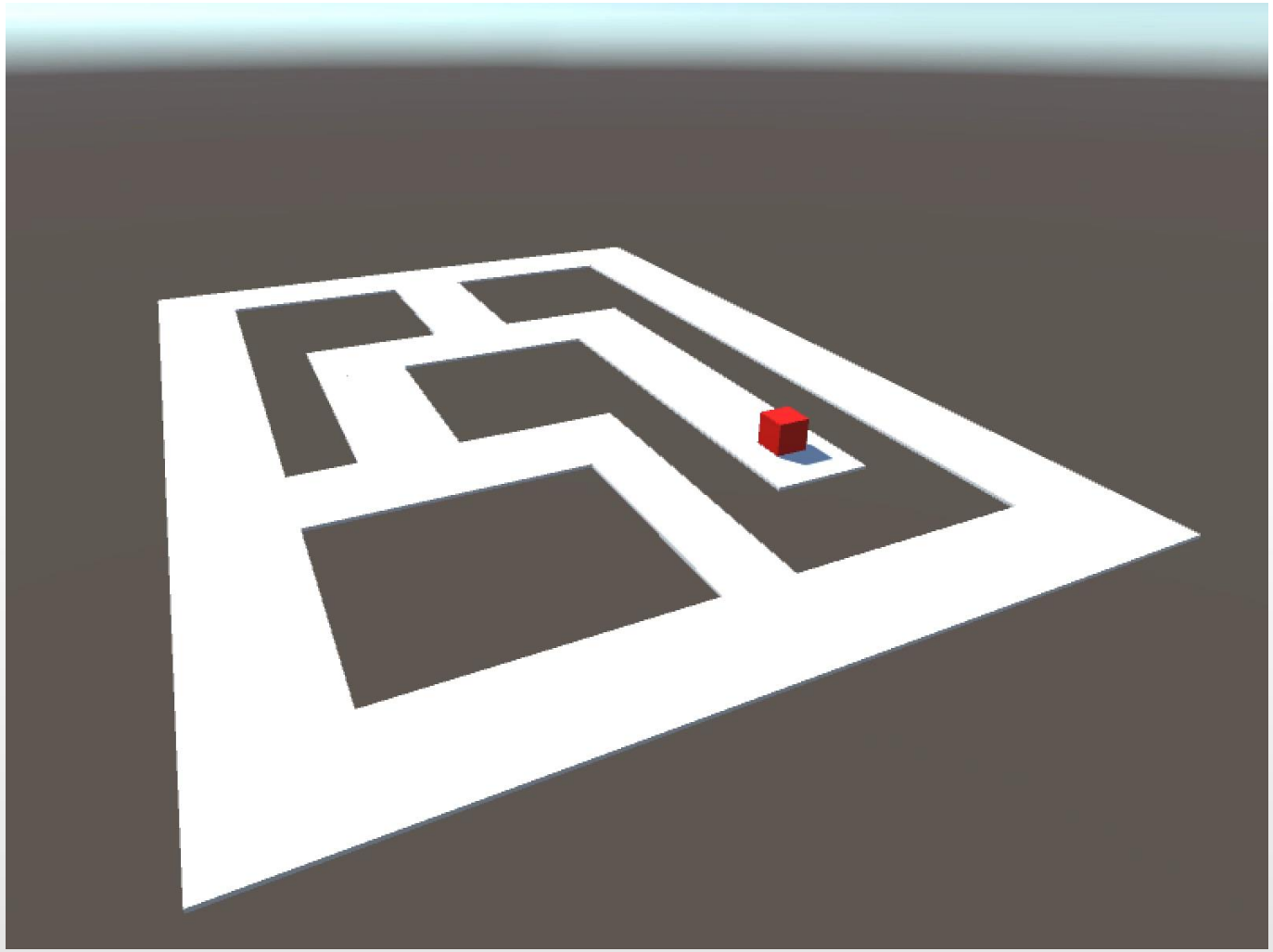
References

- On the book
  - § 4.4.4

- On the web
  - https://docs.unity3d.com/Manual/nav-BuildingNavMesh.html
  - https://docs.unity3d.com/Manual/nav-CreateNavMeshAgent.html

Go Where I Click

Scene: Go To Click
Folder: Pathfinding/NavMeshes



Go Where I Click

Source: GoToClick
Folder: Pathfinding/NavMeshes

```
using UnityEngine;
using UnityEngine.AI;

[RequireComponent(typeof(NavMeshAgent))]

public class GoToClick : MonoBehaviour {

    void Update() {
        if(Input.GetMouseButton(0)) {
            Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
            RaycastHit hit;
            if (Physics.Raycast (ray, out hit)) {
                GetComponent<NavMeshAgent>().destination = hit.point;
            }
        }
    }
}
```

Create a ray from the camera lens to the point where the mouse pointer is projected on the scene

Perform the raycast and where you hit something that point is your destination