COMP4122 GAME DESIGN AND DEVELOPMENT

Navigation Lab

By Dr. Zackary P. T. Sin

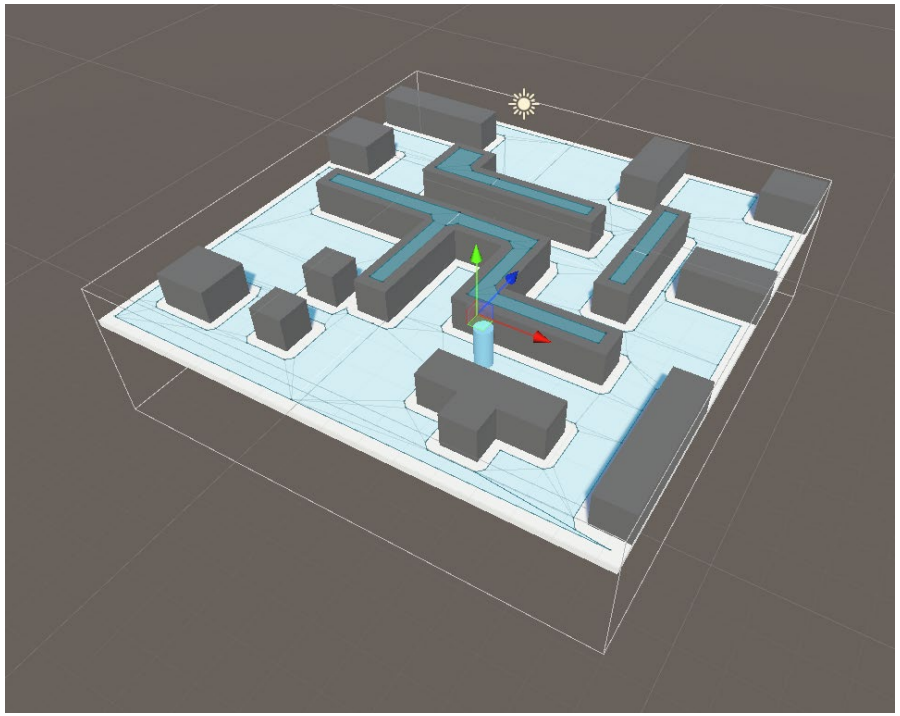
Navigation

Moving things around procedurally

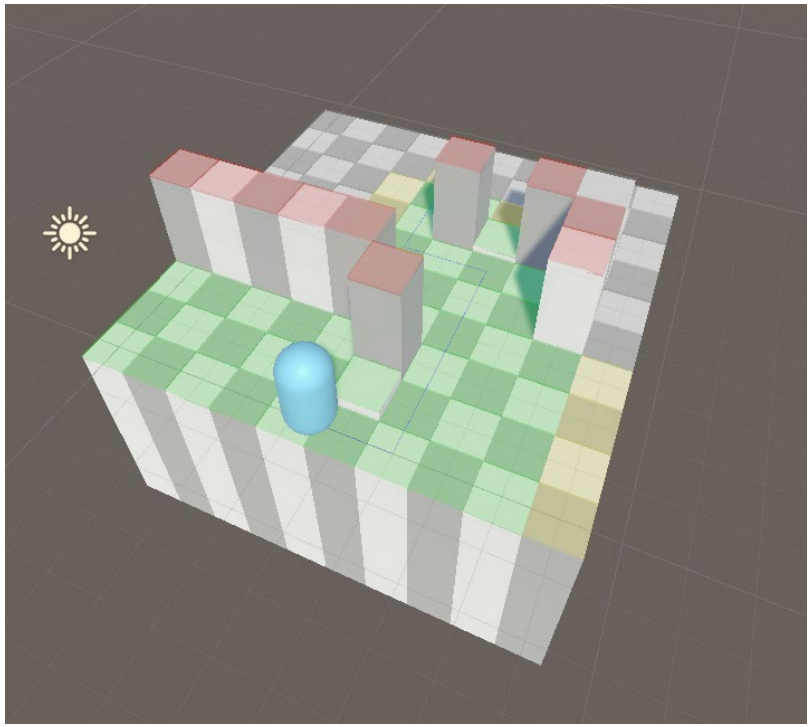
Agenda

- Previously, we tried ways for a human to control its character/agent
- But without human guidance, how can a computer move an agent from one point to another?
- Today, in Unity, we investigate the following…

NavMesh



A* Algorithm

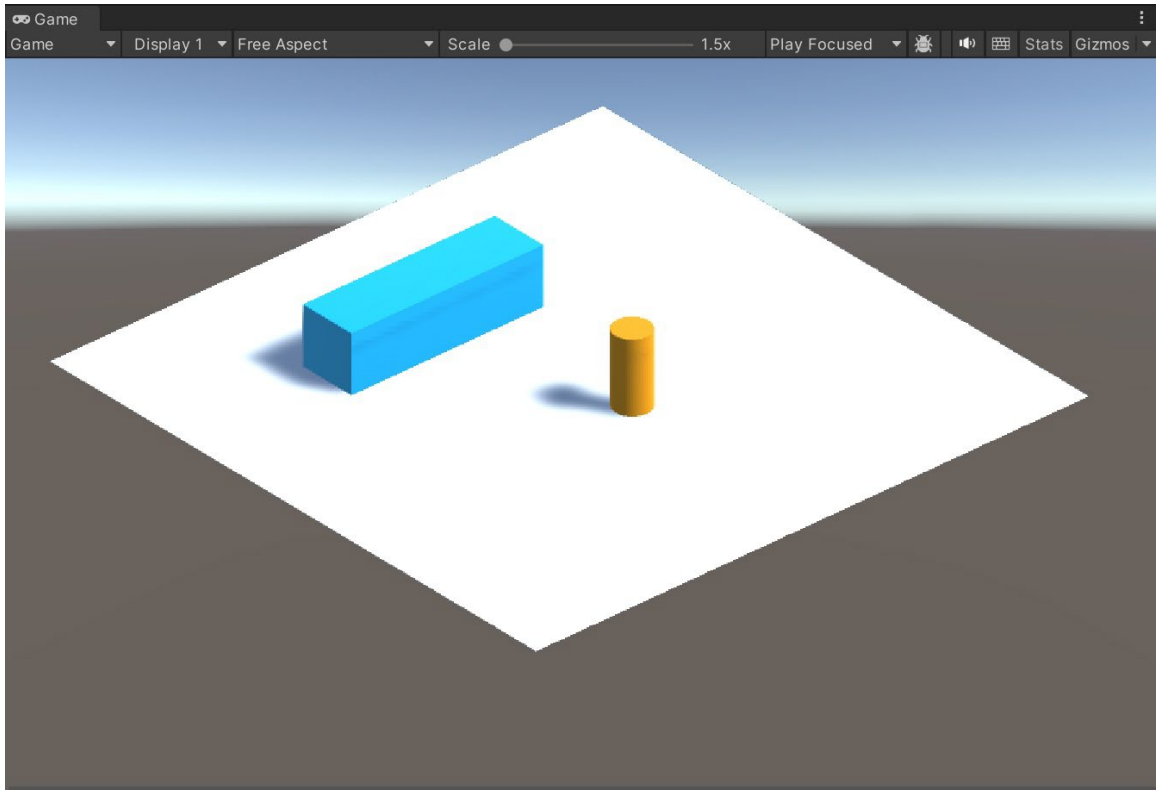


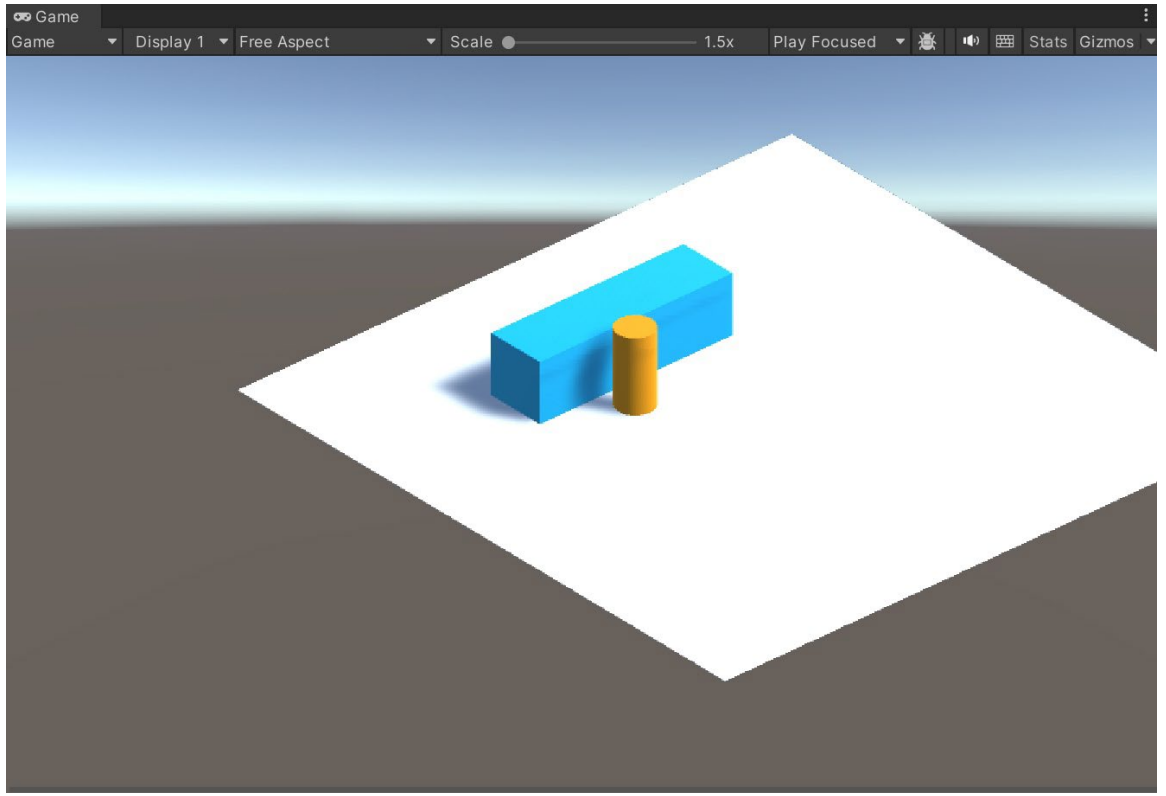
Exercise

How can we move with input again?

5 Exercise on simple movement

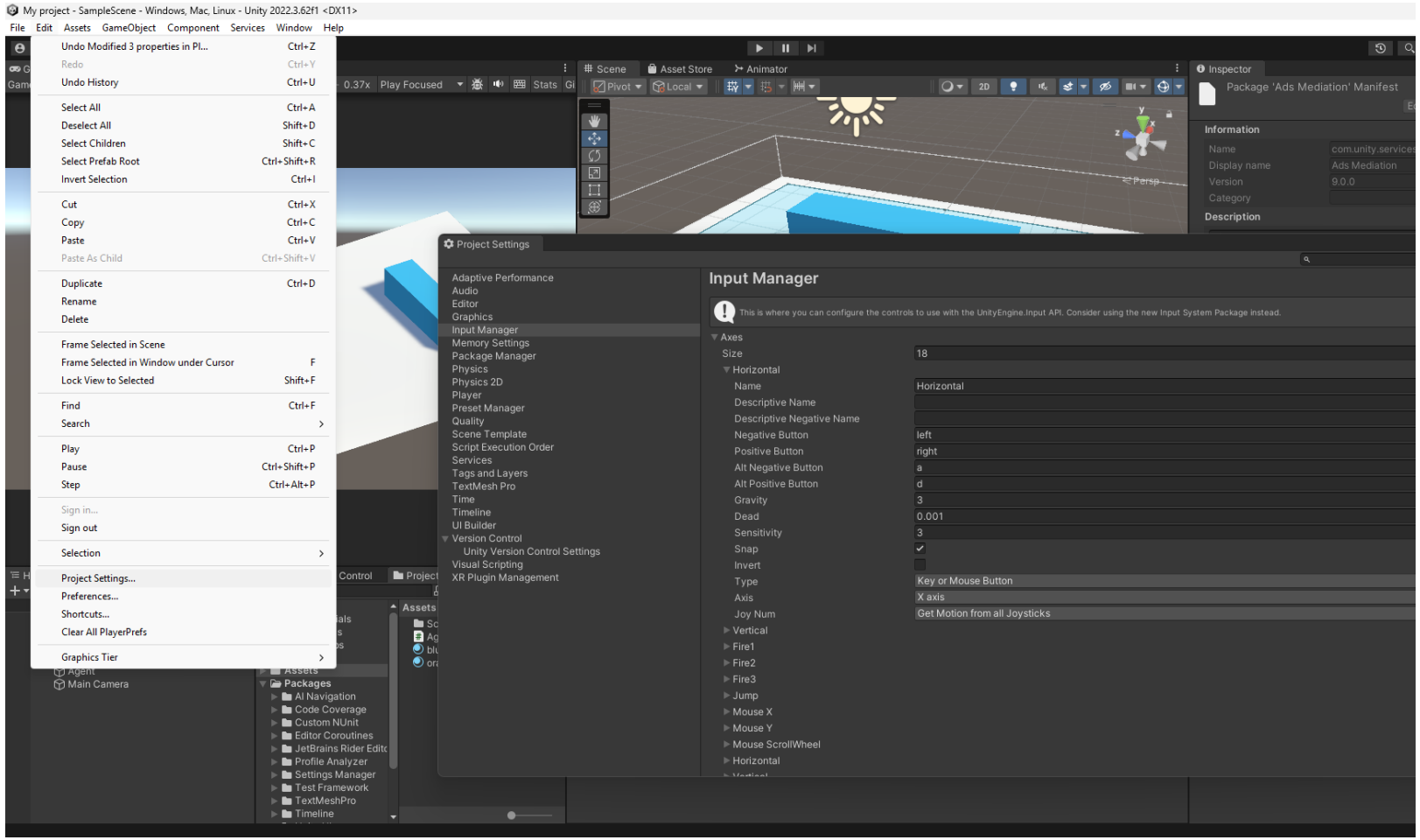
- Implement a simple script where the user can move the cylinder with WASD





6 Exercise on simple movement

- You should try to use InputManager



CODE

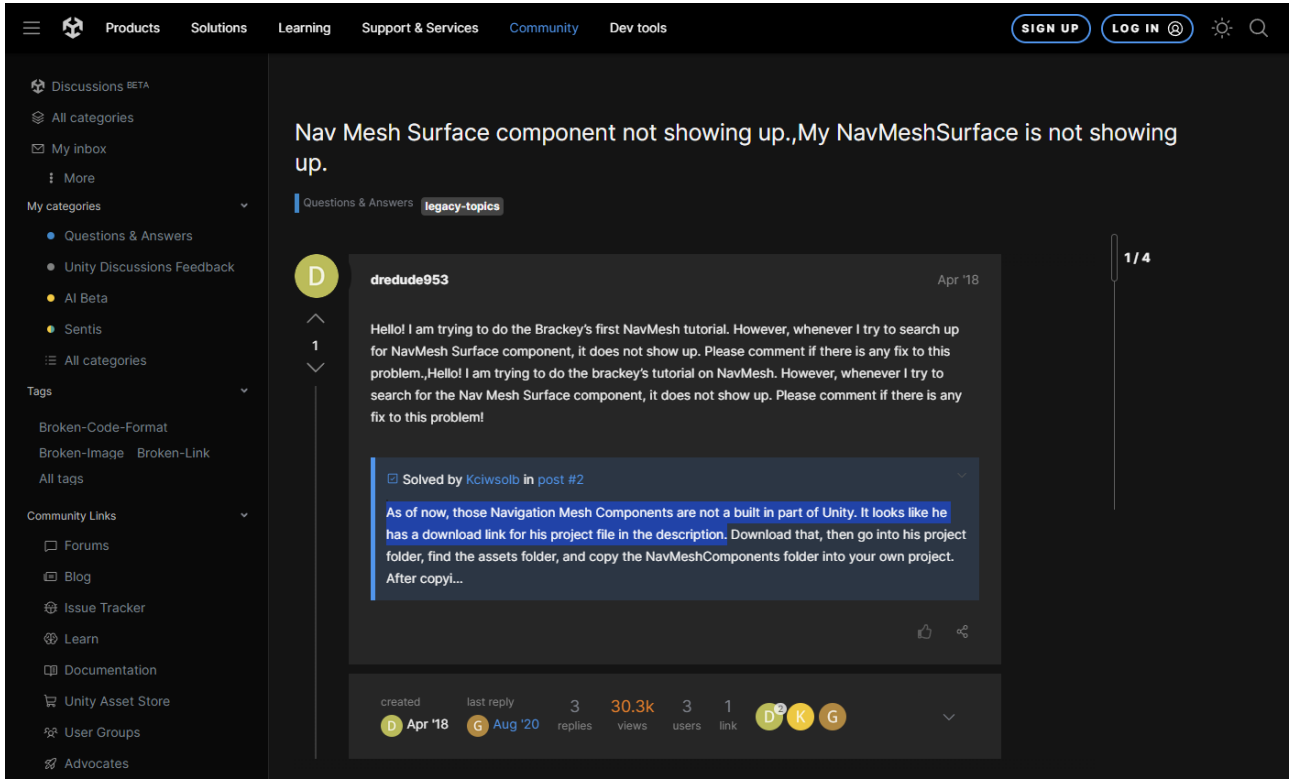
```
[SerializeField] float speed = 0.1f;

 // Update is called once per frame
 void Update()
 {

     Vector3 movement = Vector3.zero;

     movement.x = Input.GetAxis("Horizontal");
     movement.z = Input.GetAxis("Vertical");

     transform.position += movement * speed;
 }
```

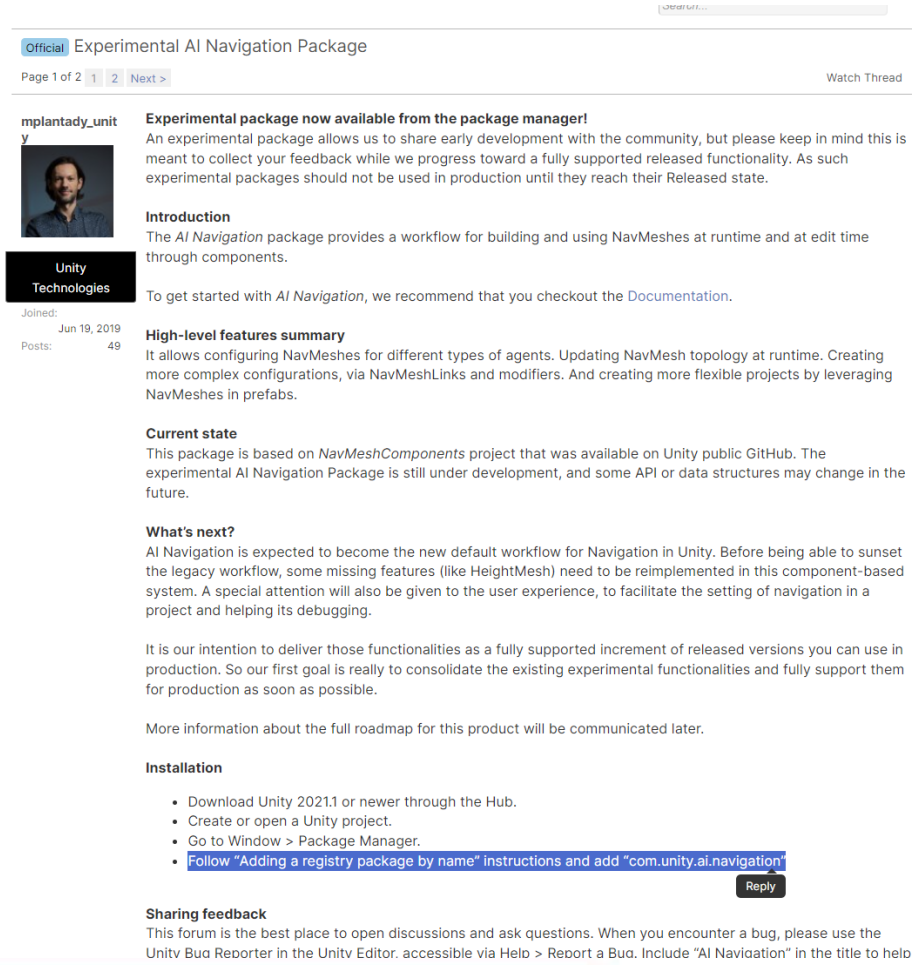
NavMesh

Unity's AI Navigation Asset

8 NavMesh is not part of default package

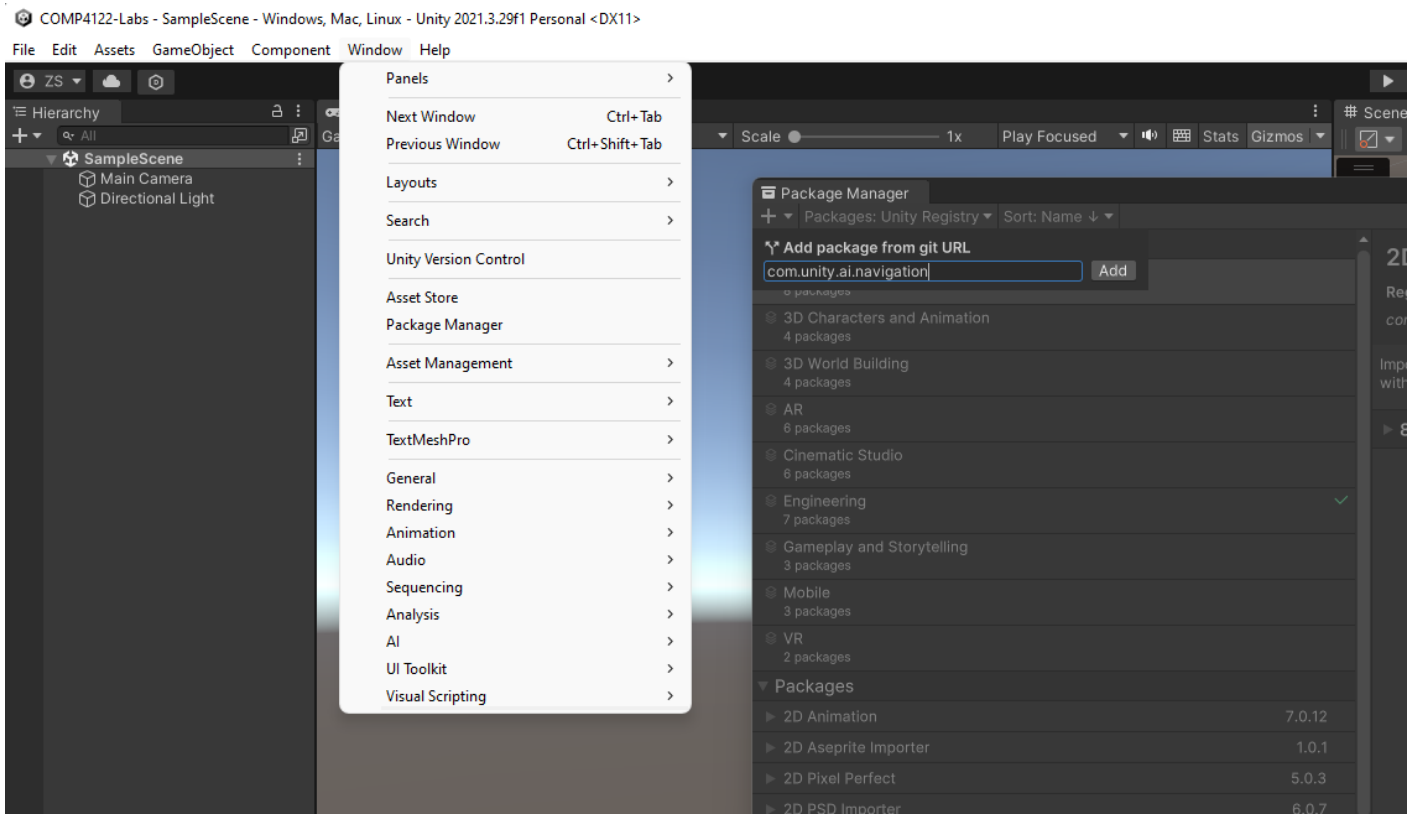
- We will be using NavMesh: https://docs.unity3d.com/Packages/com.unity.ai.navigation@2.0/manual/CreateNavMesh.html
- Always remember, some of Unity's asset is not part of the default package
- Search online and look for clues

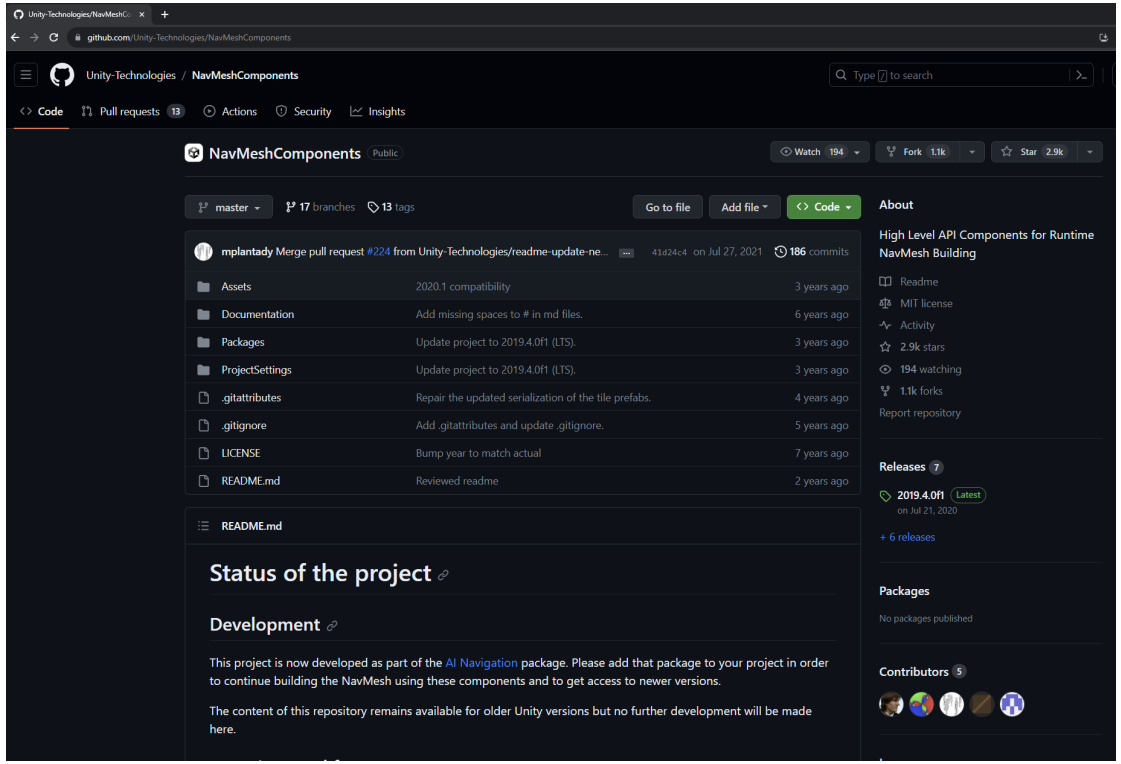




9 Installing AI Navigation package

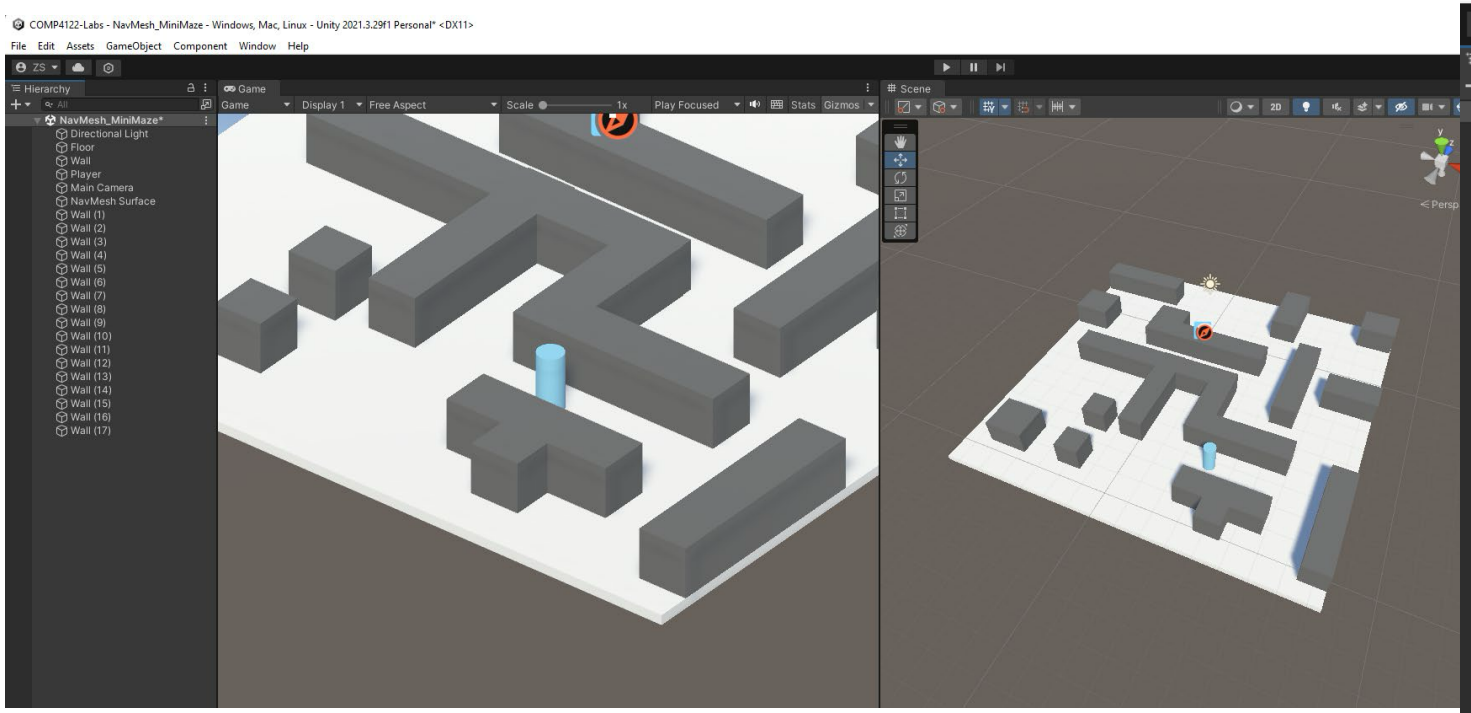
- Either install through the package manager, OR from the Github project
- **We are doing the former**
- Package Manager → Add package from git URL → “com.unity.ai.navigation” → Add

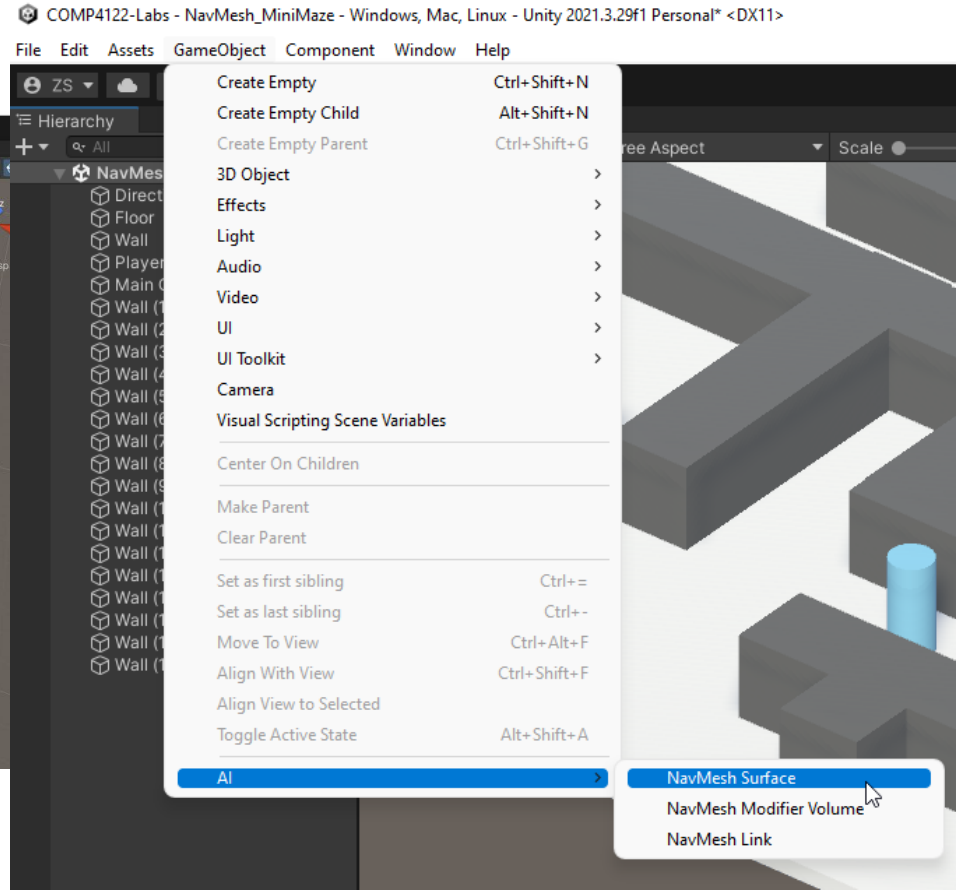




Building the maze

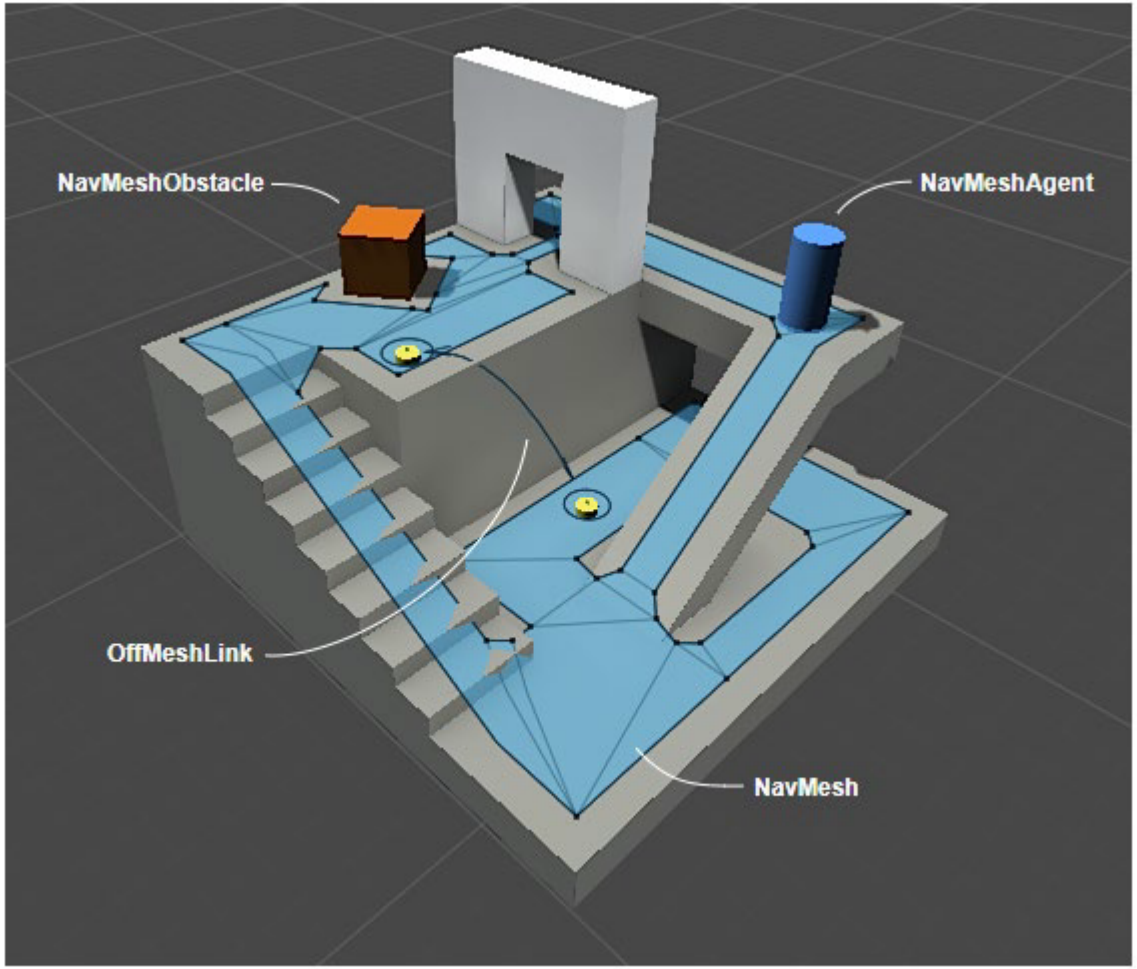
- ADD **Cubes** (GameObjec → 3D Object → Cube) to build a maze
- ADD **Cylinder** to represent the player or an agent
- ADD **NavMesh Surface** (GameObject → AI → NavMesh Surface)
- You can ADD materials to colorize for beautification/clarity





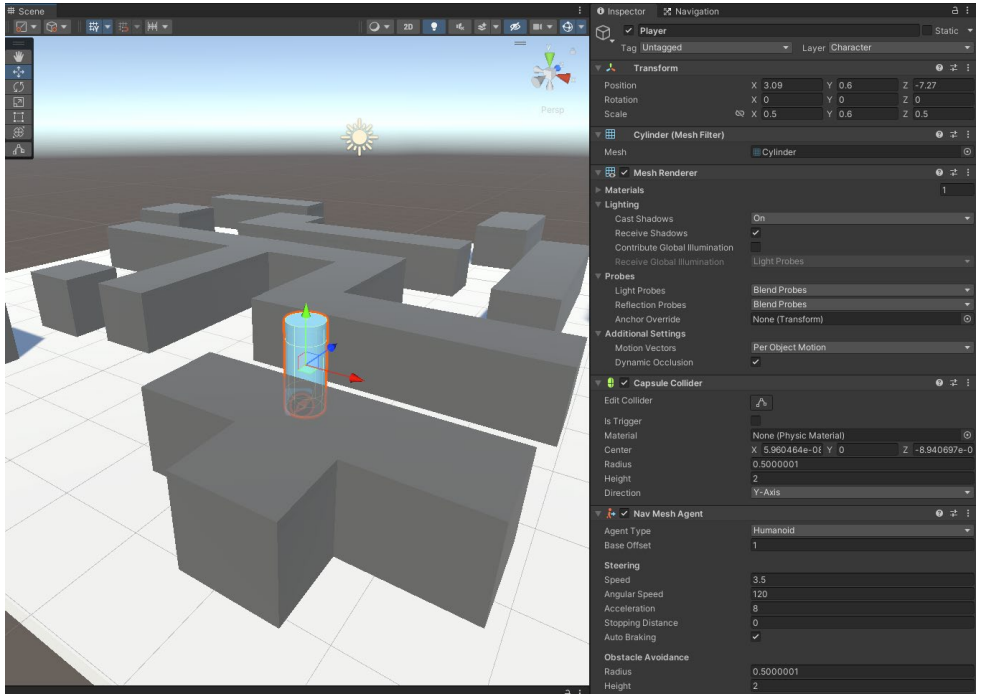
Building the maze

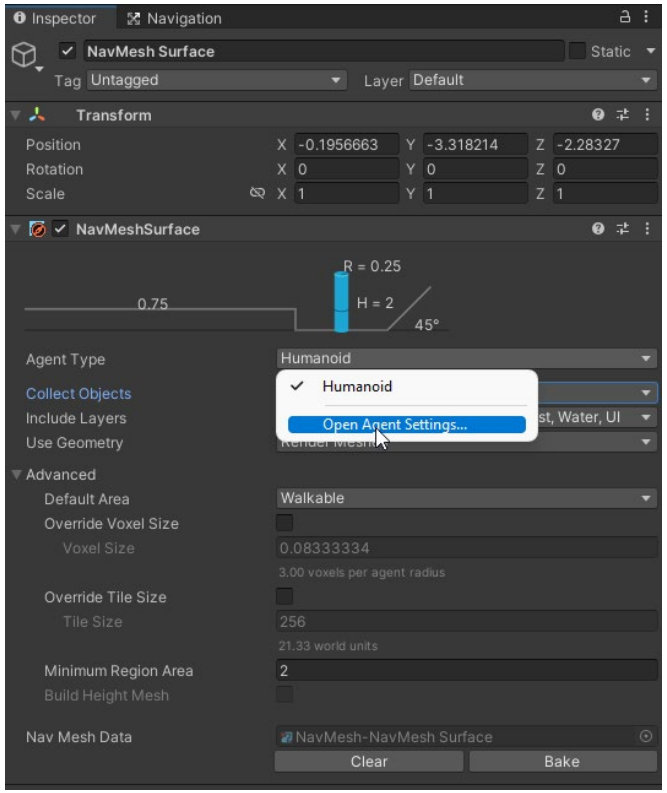
- Consider building a maze that is more fun!
- We can make it work later

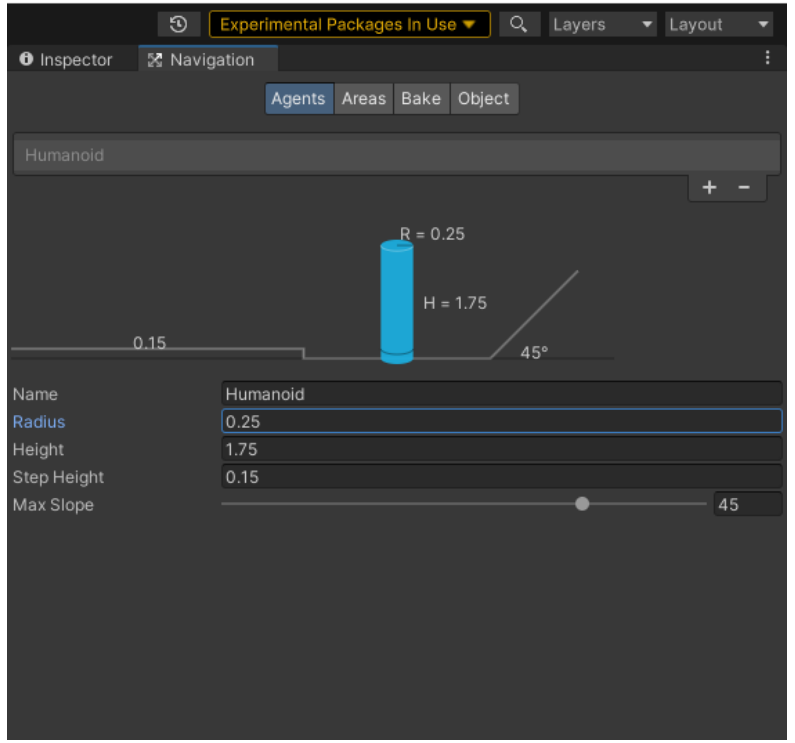


Configure the agent

- For the Cylinder Player, SET the scale as (0.5, 0.6, 0.5)
- For the Cylinder Player again, ADD the **NavMeshAgent** script
- SELECT **NavMesh Surface**, for Agent Type → SELECT Open Agent Settings
- SET Radius as 0.25, **Step Height** as 0.15
- Be observant to the settings and remember them

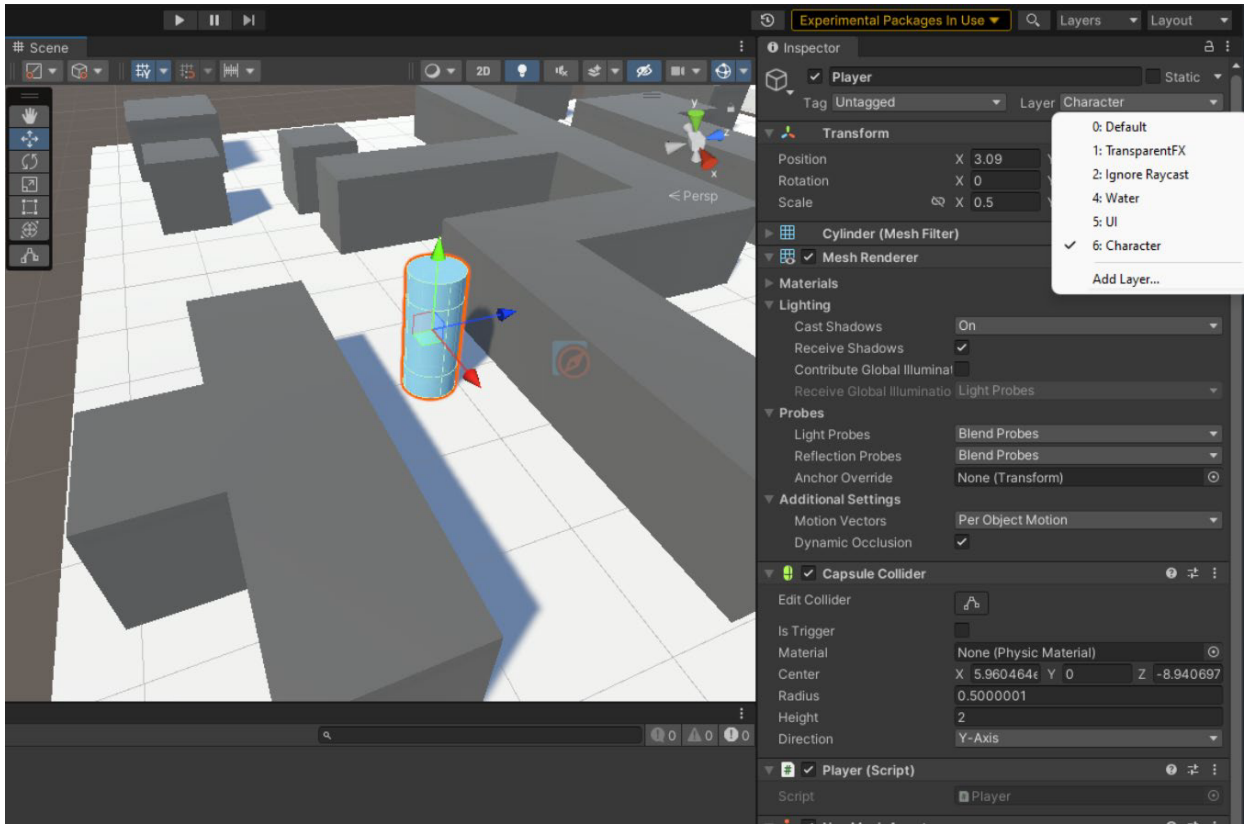


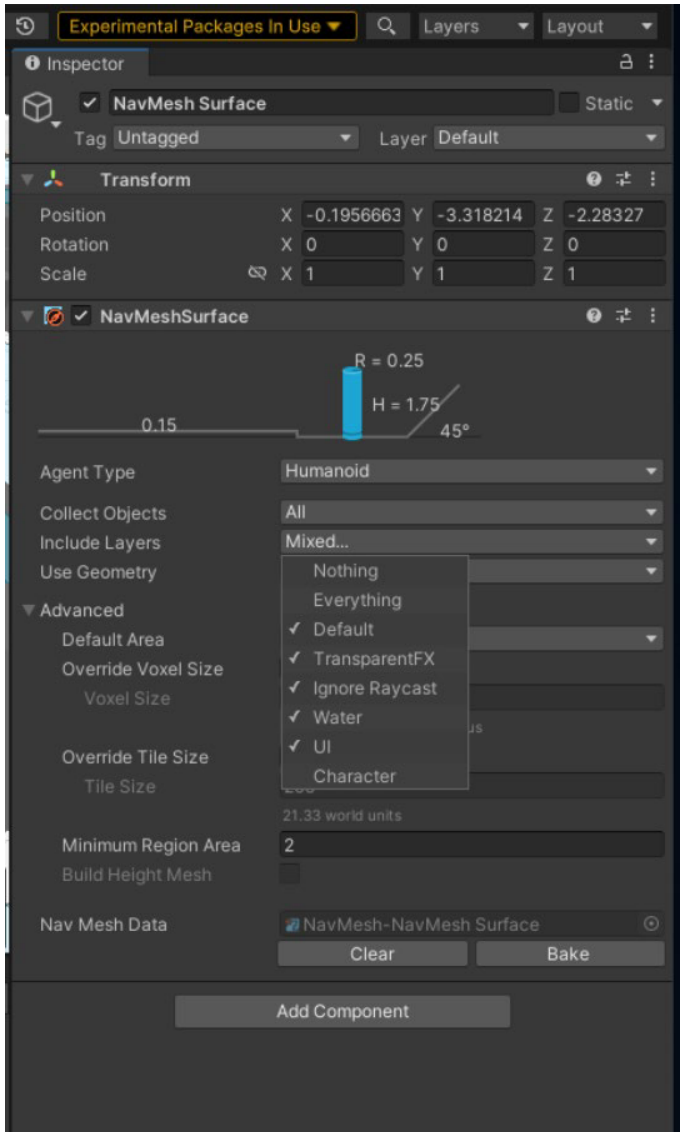




Set the layers

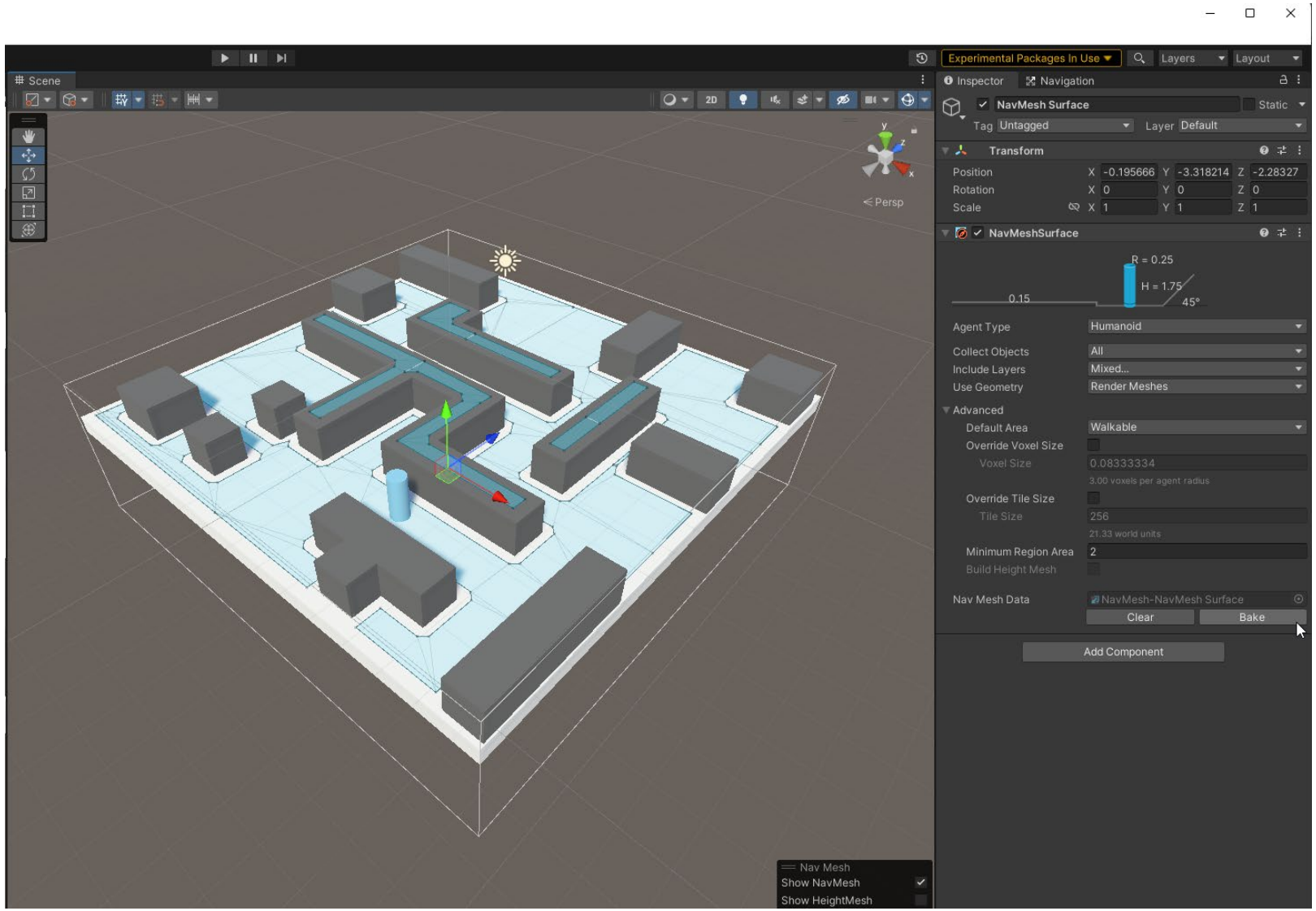
- If you bake now, the player is not on walkable ground.
- You need to tell the NavMesh Surface to ignore the player.
- ADD a new Layer named “Character”
- For Player, SET Layer as Character
- For NavMeshSurface, for Include Layers, UNCHECK Character





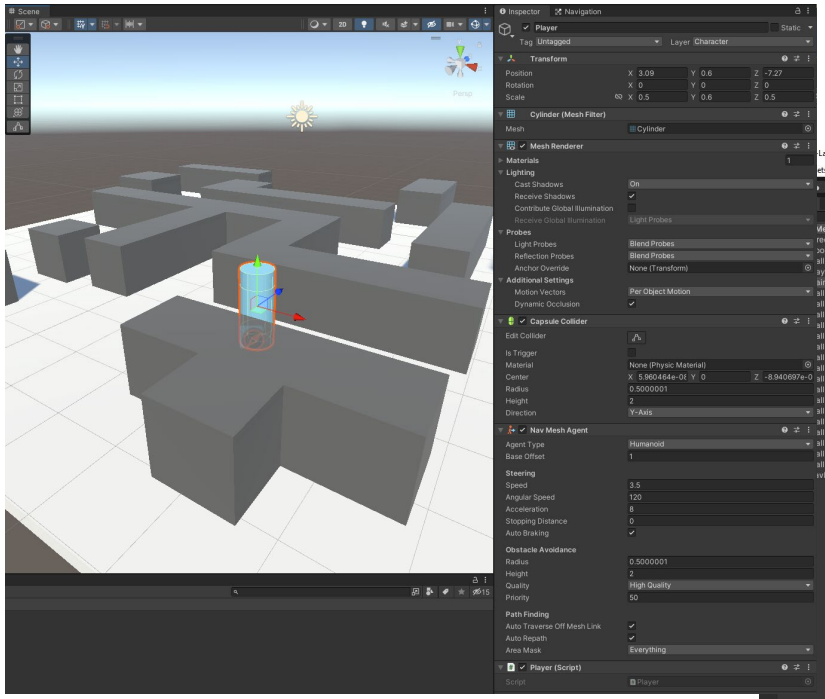
Bake the NavMesh

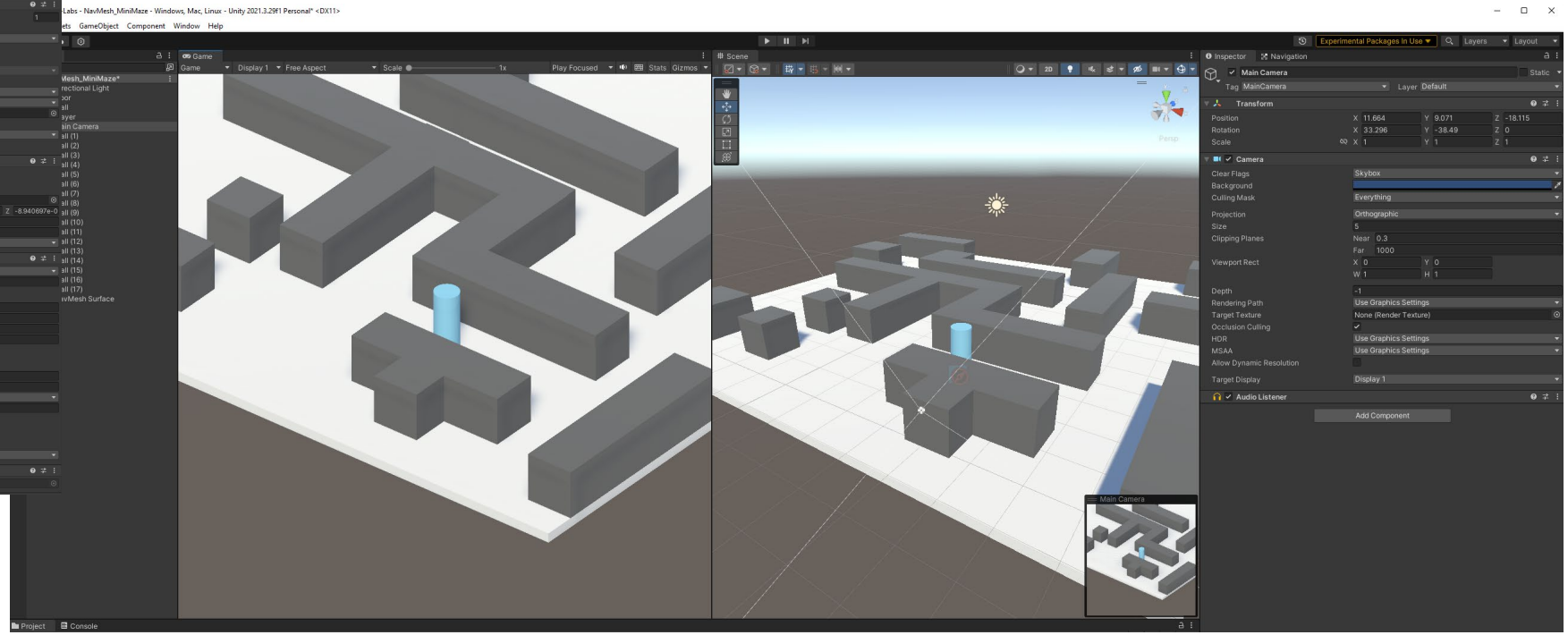
- SELECT Nav Mesh Surface, PRESS Bake
- Notice that the NavMesh is now generated
- Obviously, the blue part is the walkable part.
- Notice there are gaps between the walkable part and the walls
- How can we make the gap smaller?



Add move by click script

- For the Cylinder Player, ADD a script “Player”
- PLACE the camera at a location that is comfortable
- For the camera, FOR Projection → SELECT Orthographic
- SET Size as 5, or whatever you like as long as it can comfortably view the player





The Player script

- COPY the following code to the Player script

CODE

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;


public class MovementScript : MonoBehaviour
{

    public GameObject shootingLocation;
    public GameObject bullet;
    public Camera mainCamera;
    public GameObject barrel;
    public GameObject tankBody;


    public float bulletSpeed = 50.0f;
    [SerializeField] float speed = 0.5f;

    // Start is called before the first frame update
    void Start()
    {

    }
```

The Player script

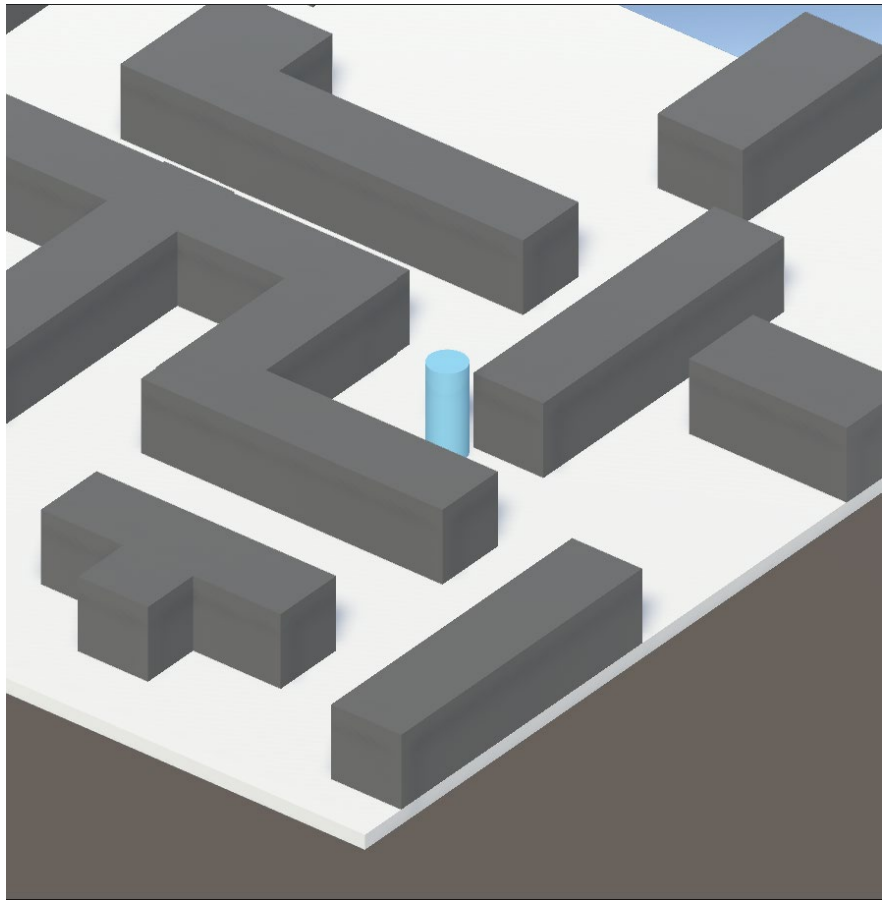
- You should further consider mixing the previous code

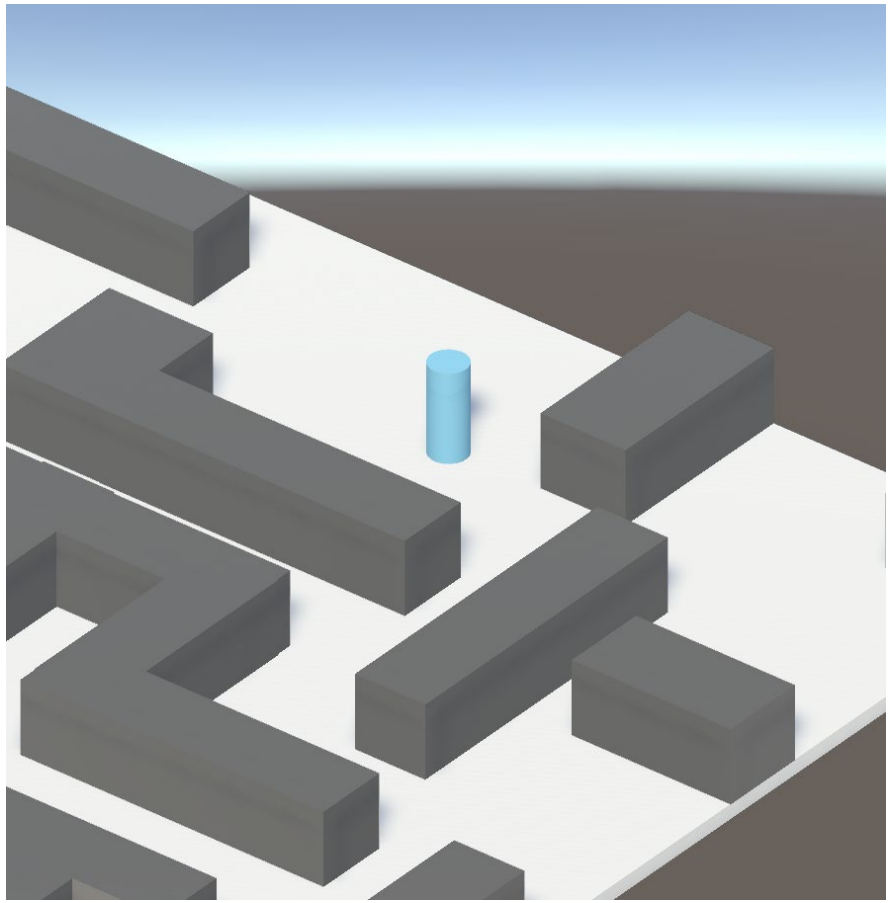
CODE

```
    void Update()
    {
        Ray ray = mainCamera.ScreenPointToRay(Input.mousePosition);
        RaycastHit hit;

        Physics.Raycast(ray, out hit);
        Vector3 targetPosition = hit.point;
        Vector3 direction = targetPosition - transform.position;
        direction.y = 0f;
        Quaternion lookRotation = Quaternion.LookRotation(direction);
        barrel.transform.rotation = Quaternion.Euler(0f, lookRotation.eulerAngles.y, 0f);


        // Tank Movement
        Vector3 movement = Vector3.zero;
        movement.x = Input.GetAxis("Horizontal");
        movement.z = Input.GetAxis("Vertical");

        transform.position += movement * speed;
        // Tank Body Transform
        transform.rotation = Quaternion.LookRotation(movement);
        // Bullets

        if (Input.GetMouseButtonDown(1)) {
            GameObject newbullet = Instantiate(bullet, shootingLocation.transform.position,
shootingLocation.transform.rotation);
            Rigidbody rb = newbullet.GetComponent<Rigidbody>();
            rb.AddForce(newbullet.transform.forward * bulletSpeed, ForceMode.Impulse);
        }
    }
```

Try!

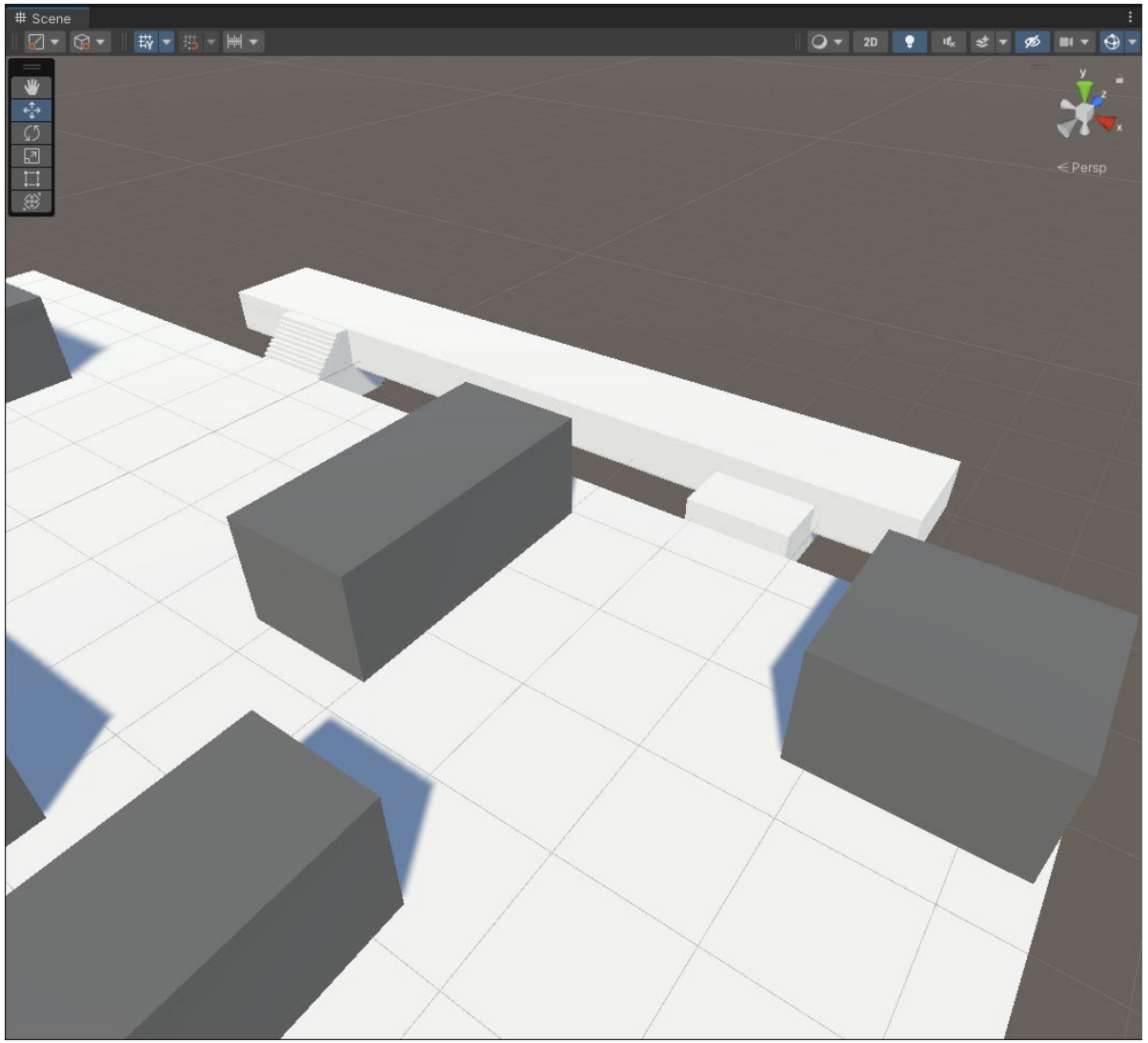
- Press PLAY and try by clicking





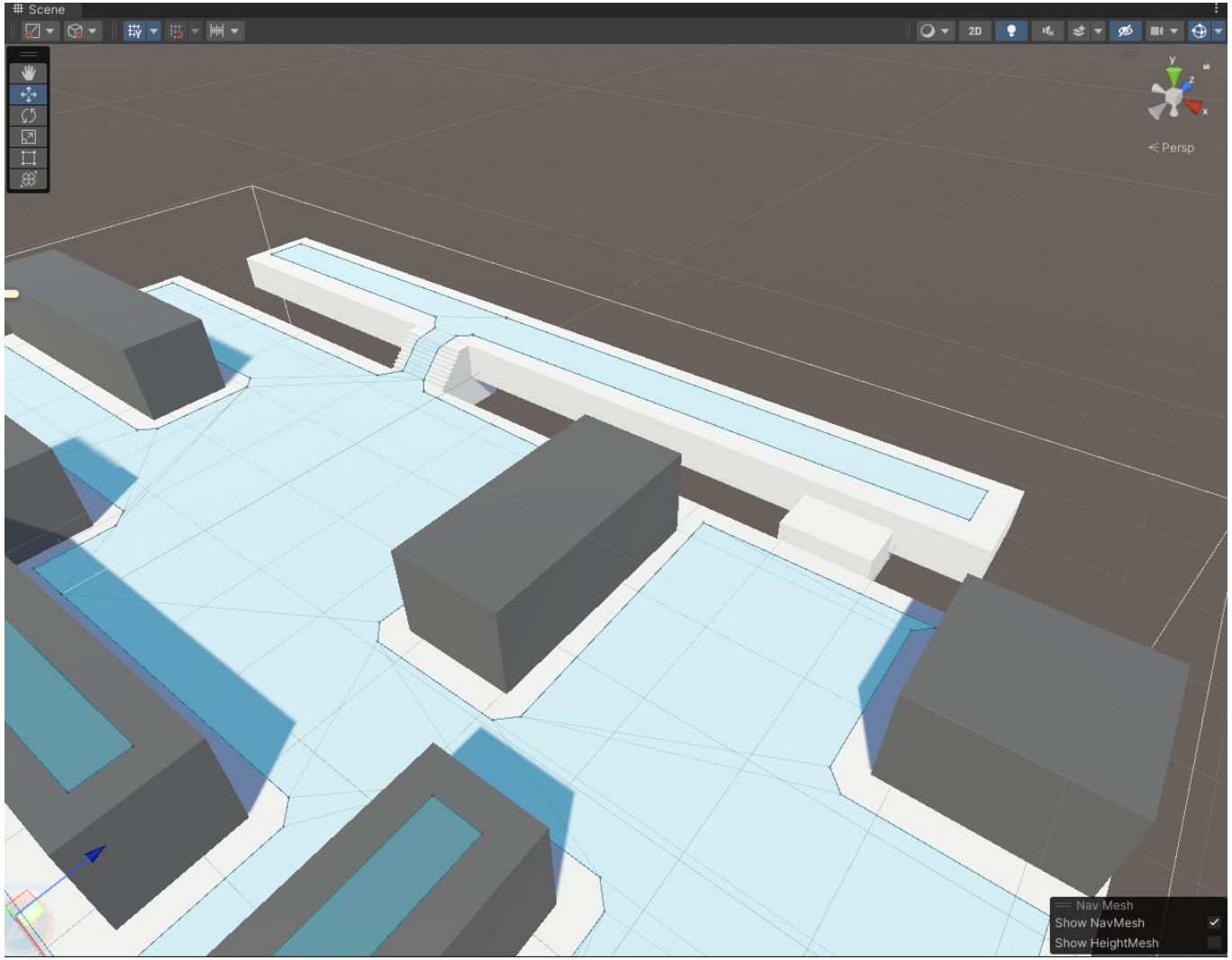
Placing staircases

- Build the two staircases like so
- One with shorter steps
- The other with taller steps



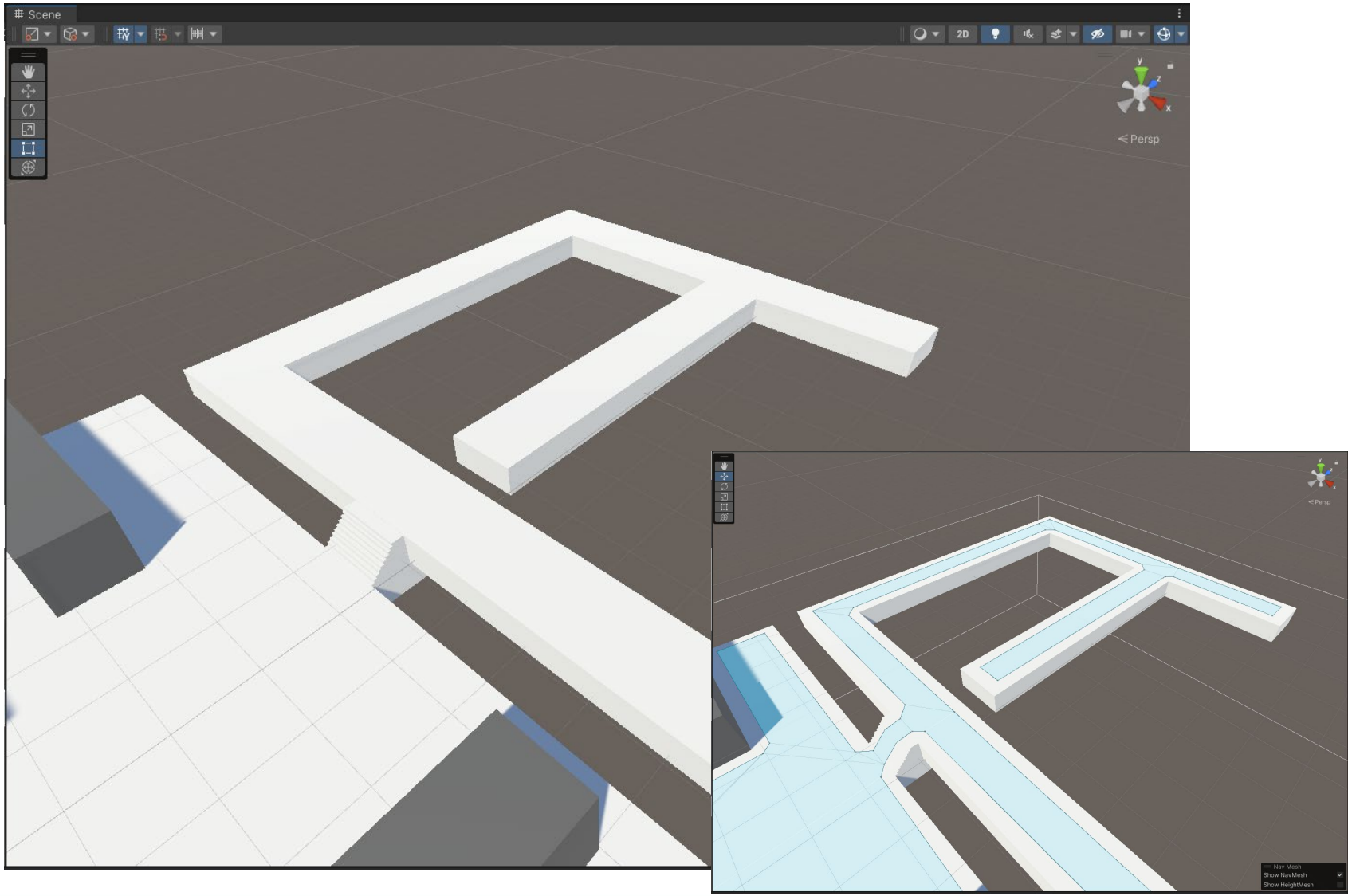
(Un)passable staircases

- BAKE the NavMesh again
- Notice that only one staircase can be crossed
- Why?
- Will NavMesh work with uneven surface?



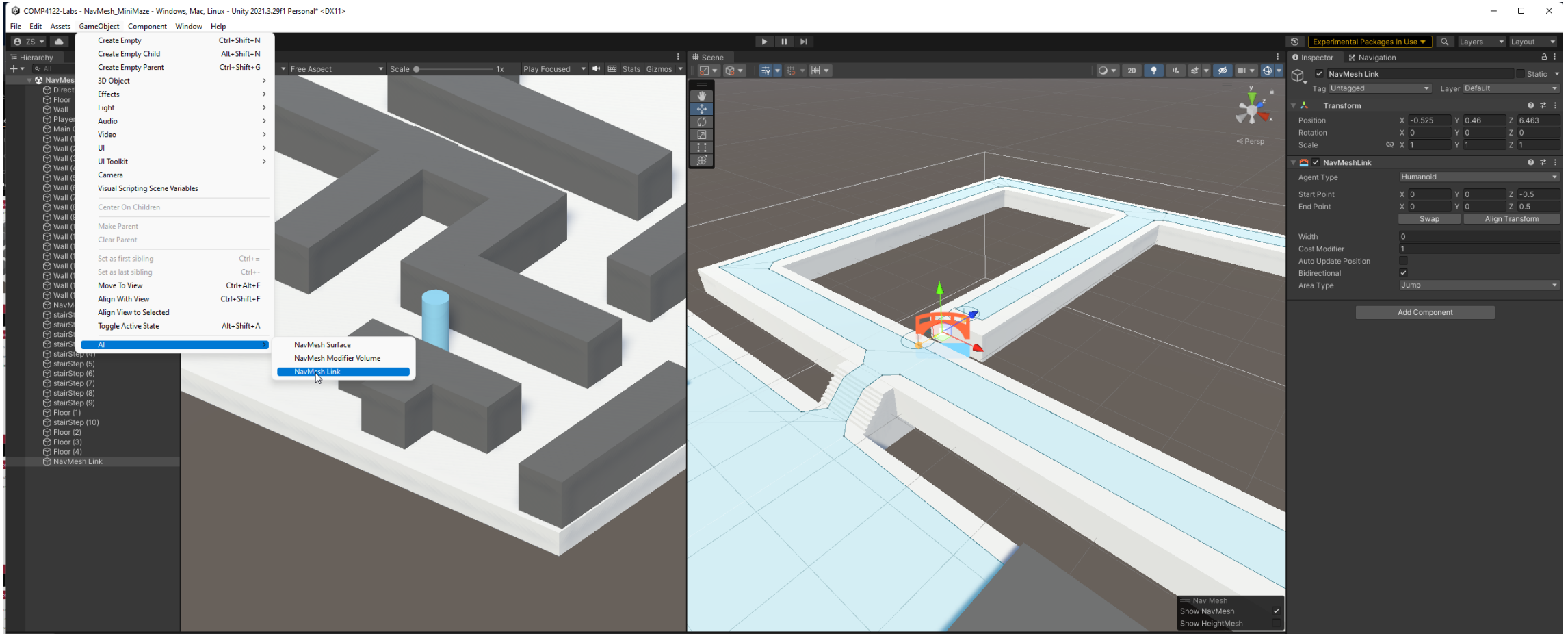
Walkable and jumpable routes

- Extend the maze with two routes
- One which is directly walkable
- The other have a gap
- Obviously, the one with a gap, cannot be crossed now



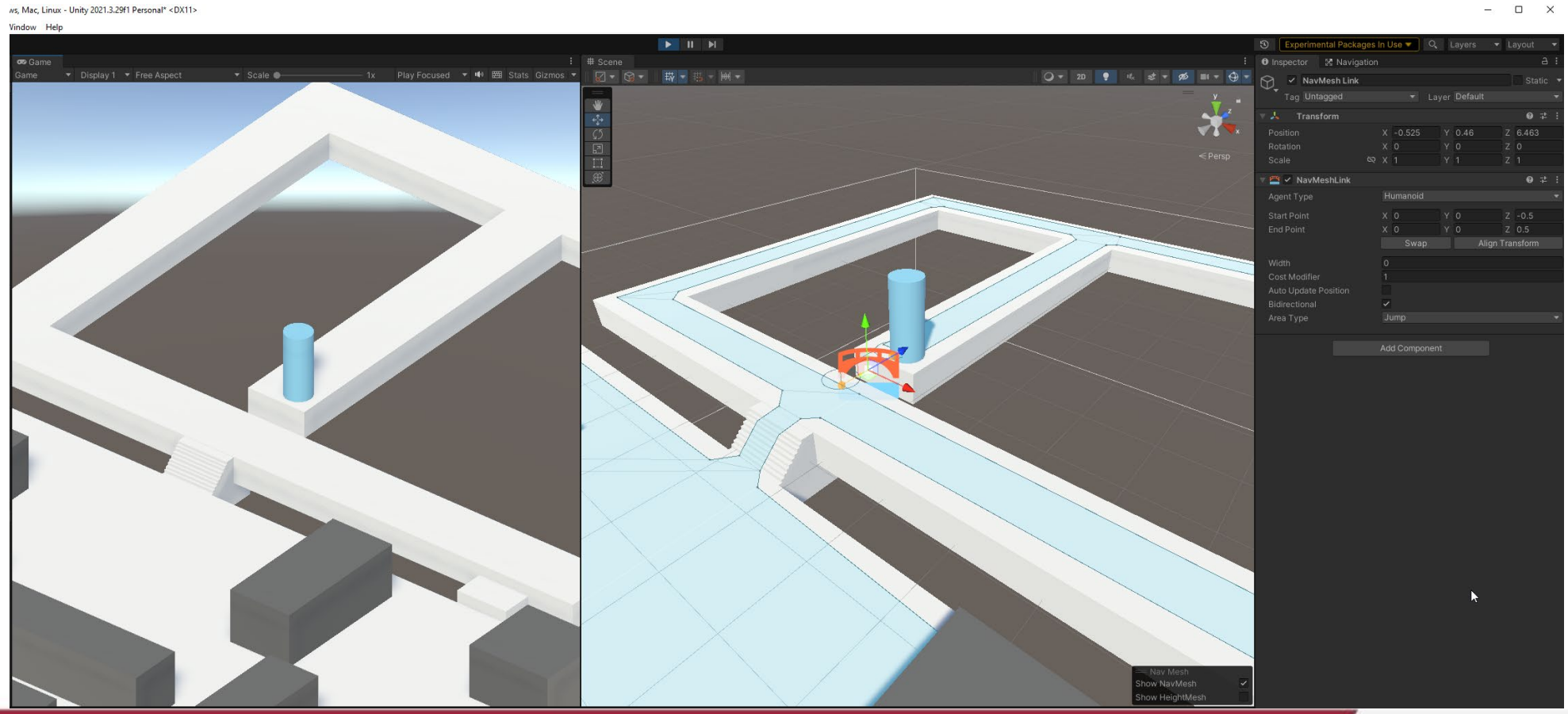
NavMesh Link

- ADD NavMeshLink (GameObject → AI → NavMesh Link)
- Place the NavMeshLink at the gap
- BAKE NavMesh again



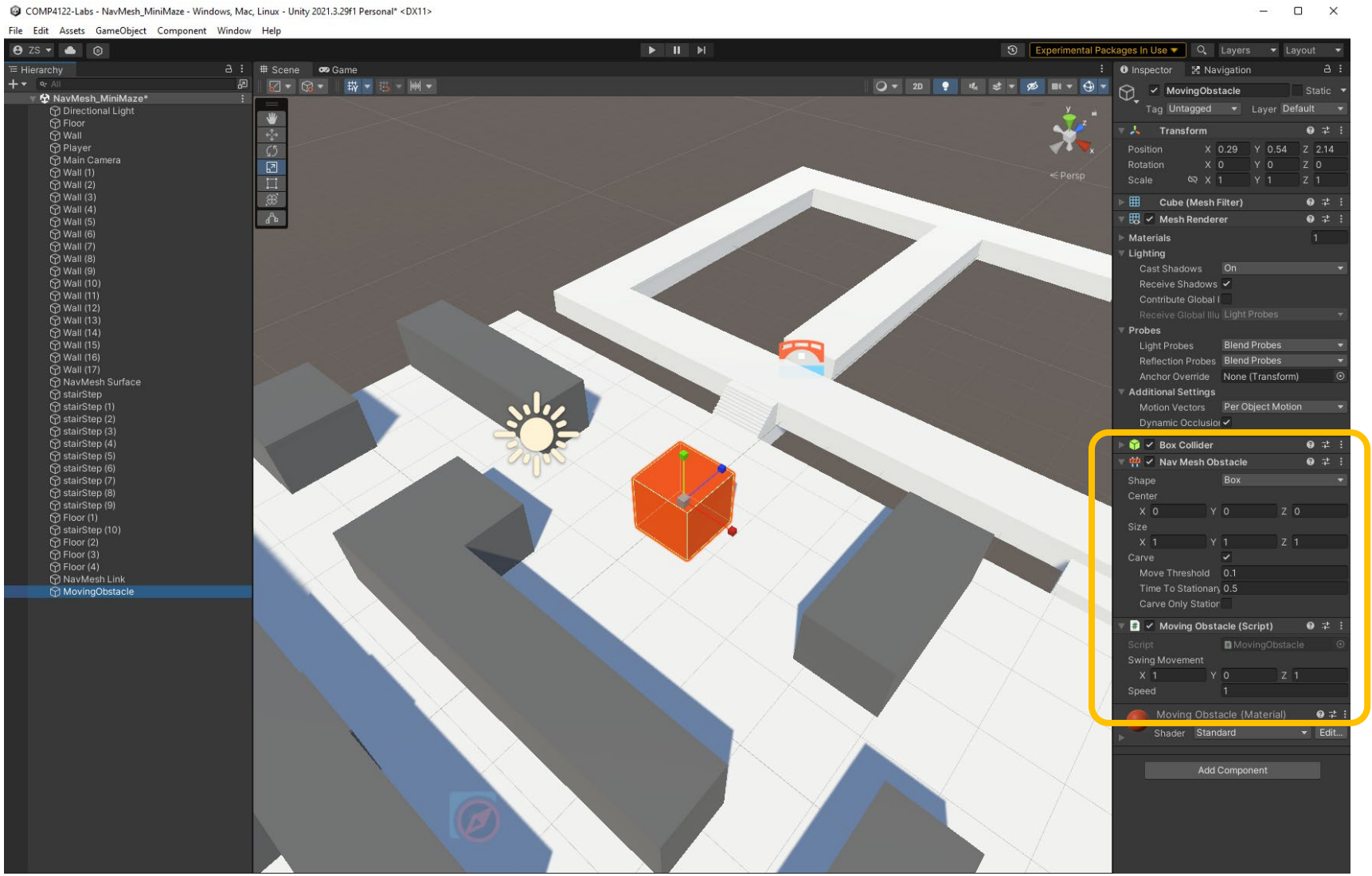
NavMesh Link's cost

- SET NavMeshLink's Cost Modifier as 1
- Press PLAY
- Works! But how can we change the "animation" when the player is jumping across the gap?
- Try different Cost Modifier value to see when the player will get cross the gap



Dynamic NavMesh with obstacle

- What if we have a moving obstacle?
- ADD a Cube, named it “Moving Obstacle”
- ADD NavMeshObstacle
- ADD a new script named MovingObstacle
- Be mindful of the parameters for the components



Changing the map

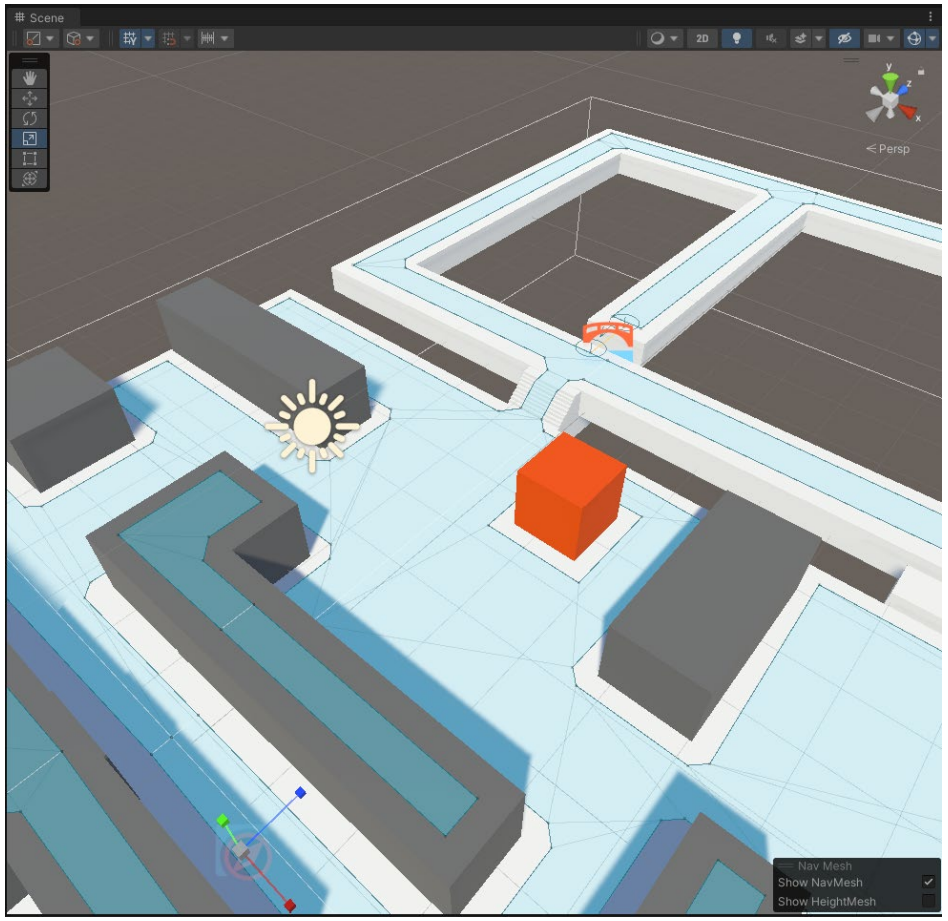
- COPY the following code to the MovingObstacle script

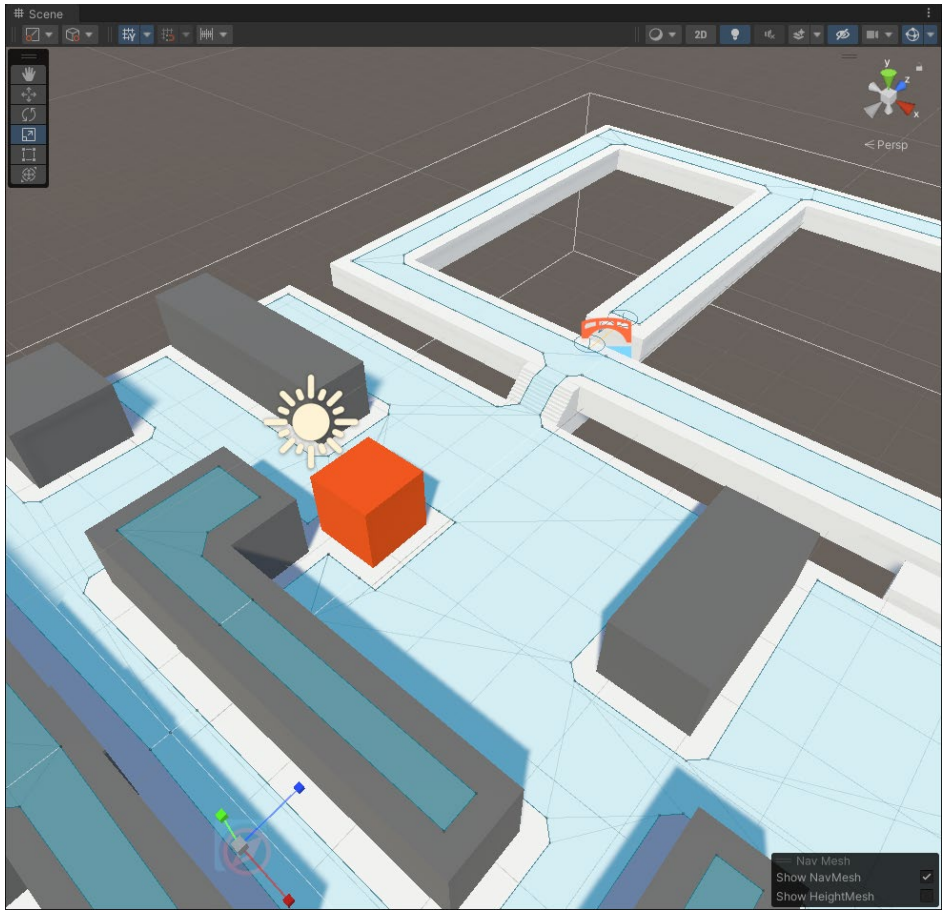
CODE

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MovingObstacle : MonoBehaviour
{
    [SerializeField]
    Vector3 swingMovement = Vector3.one;

    [SerializeField]
    float speed = 1;

    Vector3 startLocation;

    float samplePoint = 0;

    private void Awake()
    {
        startLocation = transform.position;
    }

    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        samplePoint += speed * Time.deltaTime;

        transform.position = startLocation + Mathf.Sin(samplePoint) * swingMovement;
    }
}
```

Dynamic NavMesh with obstacle

- Press PLAY
- Notice how the agent react.



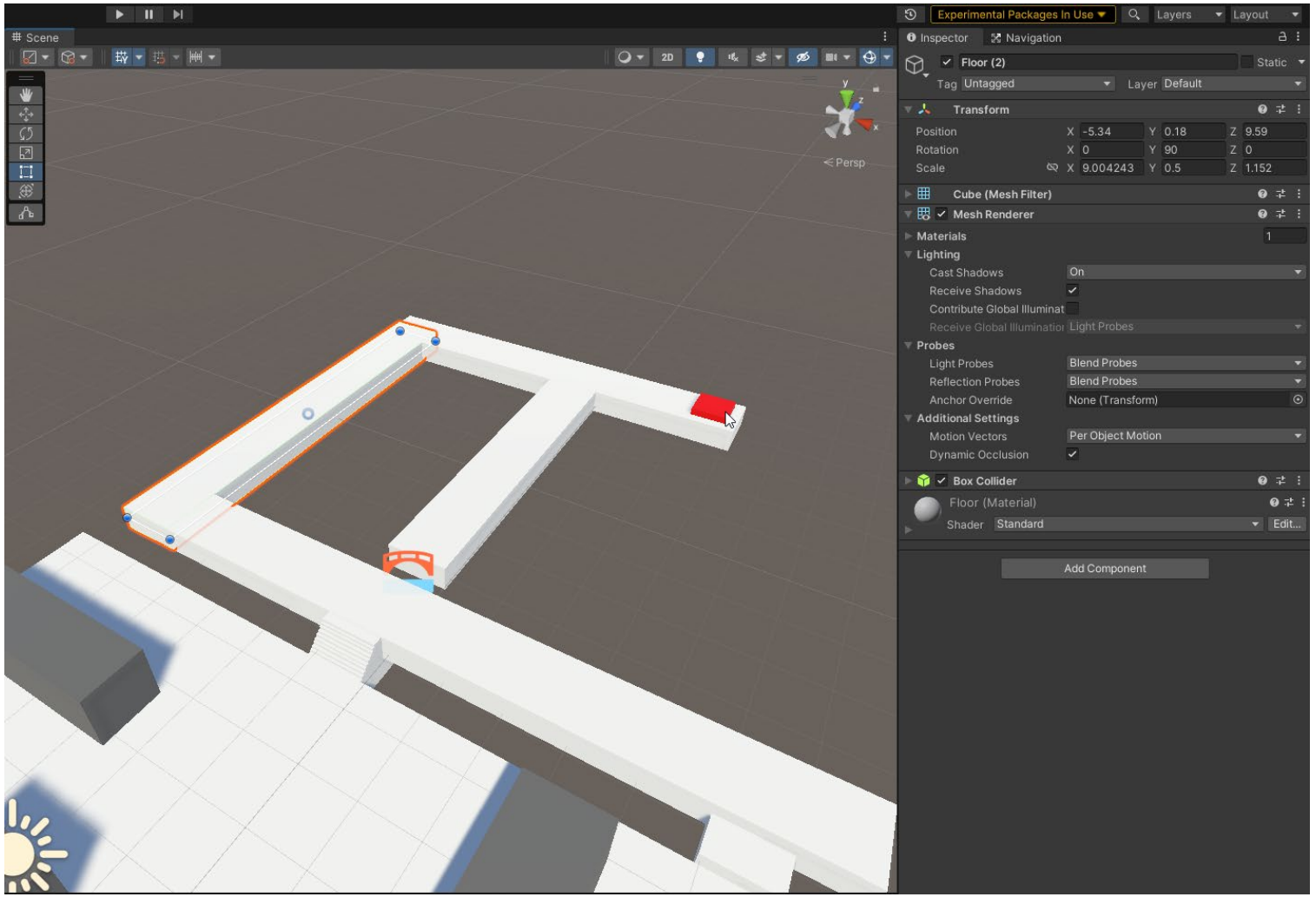


Exercise

Or for your to think about

Changing the map

- Consider the following…
- ADD a Red Button such that when the character walks on it, the walkable path on the left will be gone
- How can we re-bake the NavMesh procedurally?



Changing the map

- Check the code for NavMeshSurface
- There is a function BuildNavMesh()
- So, the idea is that when the Red Button is triggered, call this function.
- Is it a good approach with this exercise? Actually no ☺
- Why? Can you think of other ways?
- But it is suitable for procedurally generated map, which we will discuss in a later lab.
- You may also want to check out “NavMeshModifierVolume”

NavMeshSurface.cs

Miscellaneous Files — Unity.AI.Navigation.NavMeshSurface

```
231             }
232
233             buildSettings.minRegionArea = minRegionArea;
234
235             return buildSettings;
236         }
237
238         /// <summary> Builds and instantiates this NavMesh surface. </summary>
239         public void BuildNavMesh()
240         {
241             var sources = CollectSources();
242
243             // Use unscaled bounds - this differs in behaviour from e.g. collider components.
244             // But is similar to reflection probe - and since navmesh data has no scaling support - it is the right choice here.
245             var surfaceBounds = new Bounds(m_Center, Abs(m_Size));
246             if (m_CollectObjects == CollectObjects.All || m_CollectObjects == CollectObjects.Children)
247             {
248                 surfaceBounds = CalculateWorldBounds(sources);
249             }
250
251             var data = NavMeshBuilder.BuildNavMeshData(GetBuildSettings(),
252                     sources, surfaceBounds, transform.position, transform.rotation);
253
254             if (data != null)
255             {
256                 data.name = gameObject.name;
257                 RemoveData();
258                 m_NavMeshData = data;
259                 if (isActiveAndEnabled)
260                     AddData();
261             }
262         }
```

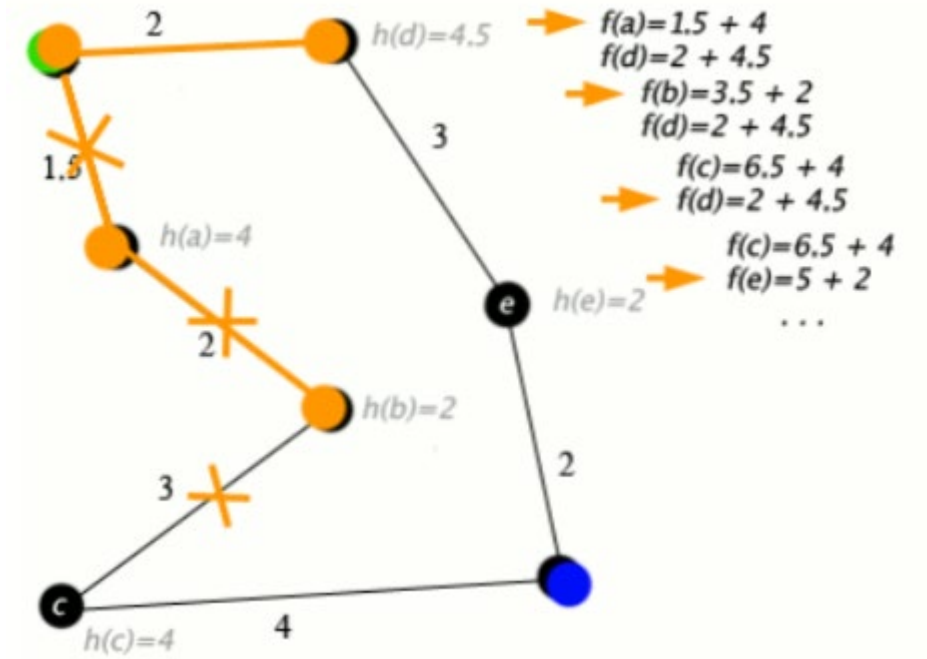
A* Algorithm

A quick prototype for learning only

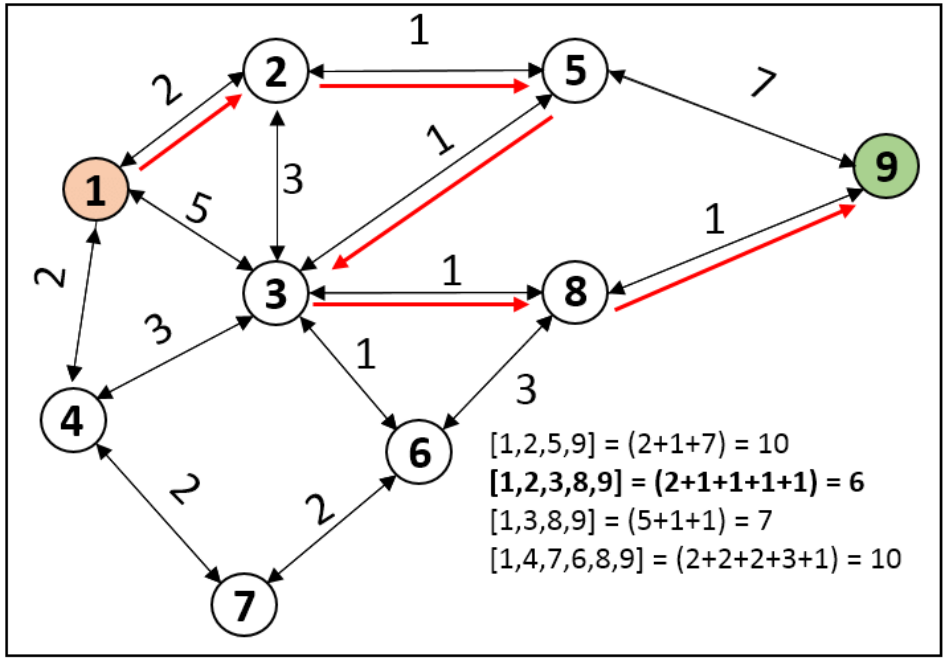
A* Algorithm vs Dijkstra's Algorithm

- Before we start, a question.
- What is the difference between A* algorithm and Dijkstra's algorithm?

A* Algorithm



Dijkstra's Algorithm



Pseudo code for A* Algorithm

- Let’s look at the pseudo code first
- This one is from Wikipedia
- https://en.wikipedia.org/wiki/A*_search_algorithm
- Two things to bare in mind
- First, the distance cost
- Second, the heuristic function
- We will write a prototype for learning only

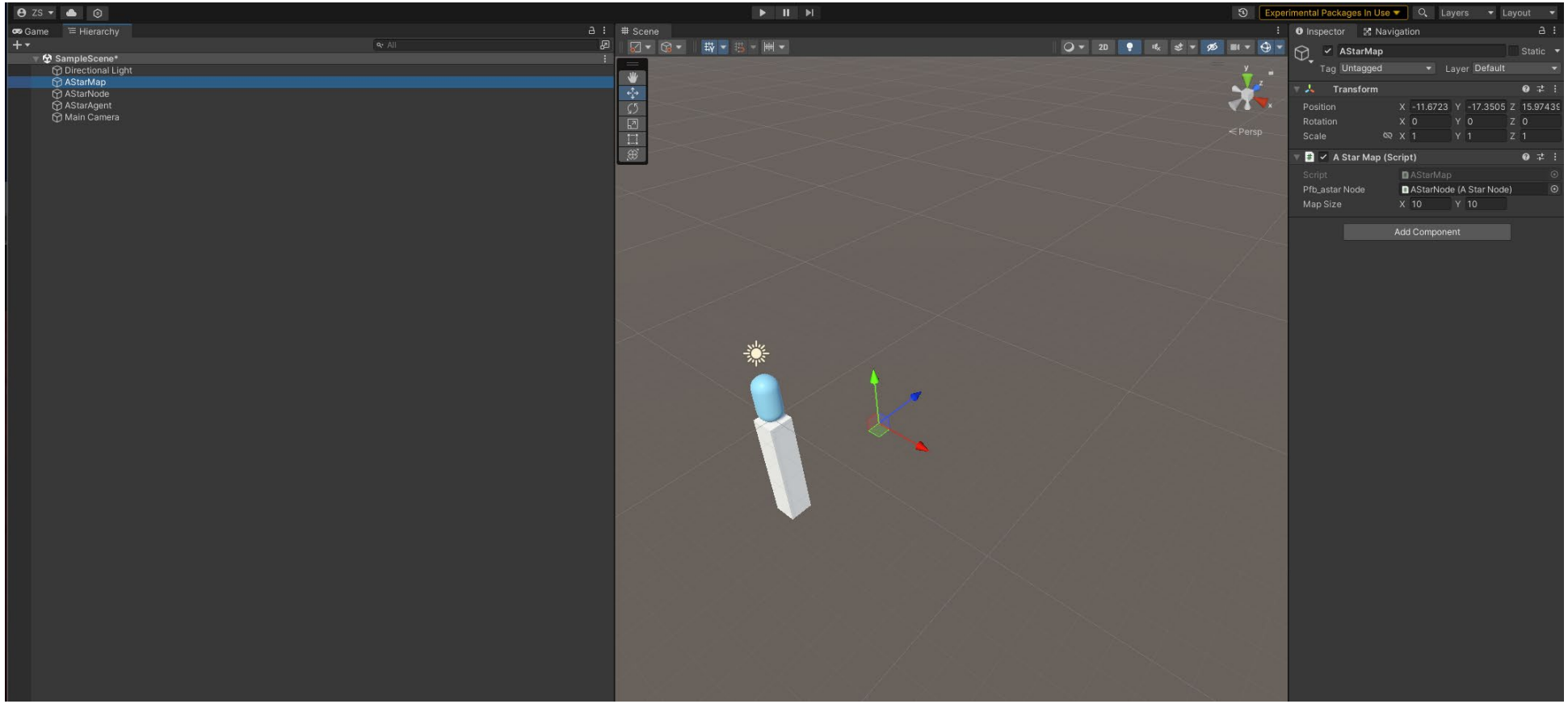
Pseudocode [edit]

The following pseudocode describes the algorithm:

```
function reconstruct_path(cameFrom, current)
    total_path := {current}
    while current in cameFrom.Keys:
        current := cameFrom[current]
        total_path.prepend(current)
    return total_path

// A* finds a path from start to goal.
// h is the heuristic function. h(n) estimates the cost to reach goal from node n.
function A_Star(start, goal, h)
    // The set of discovered nodes that may need to be (re-)expanded.
    // Initially, only the start node is known.
    // This is usually implemented as a min-heap or priority queue rather than a hash-set.
    openSet := {start}

    // For node n, cameFrom[n] is the node immediately preceding it on the cheapest path from the start
    // to n currently known.
    cameFrom := an empty map

    // For node n, gScore[n] is the cost of the cheapest path from start to n currently known.
    gScore := map with default value of Infinity
    gScore[start] := 0

    // For node n, fScore[n] := gScore[n] + h(n). fScore[n] represents our current best guess as to
    // how cheap a path could be from start to finish if it goes through n.
    fScore := map with default value of Infinity
    fScore[start] := h(start)

    while openSet is not empty
        // This operation can occur in O(Log(N)) time if openSet is a min-heap or a priority queue
        current := the node in openSet having the lowest fScore[] value
        if current = goal
            return reconstruct_path(cameFrom, current)

        openSet.Remove(current)
        for each neighbor of current
            // d(current,neighbor) is the weight of the edge from current to neighbor
            // tentative_gScore is the distance from start to the neighbor through current
            tentative_gScore := gScore[current] + d(current, neighbor)
            if tentative_gScore < gScore[neighbor]
                // This path to neighbor is better than any previous one. Record it!
                cameFrom[neighbor] := current
                gScore[neighbor] := tentative_gScore
                fScore[neighbor] := tentative_gScore + h(neighbor)
                if neighbor not in openSet
                    openSet.add(neighbor)

    // Open set is empty but goal was never reached
    return failure
```

Generate the map

- First, let us generate a map for us to do A* on
- ADD empty game object, named it “AStarMap”, ADD a new script AStarMap on it
- ADD a Cube, named it “AStarNode”, ADD a new script AStarNode on it, sized it (1, 5, 1) and place it at (0, -2.5, 0)
- ADD a Capsule, named it “AStarAgent”, ADD a new script AStarAgent on it, place it at (0, 1, 0).



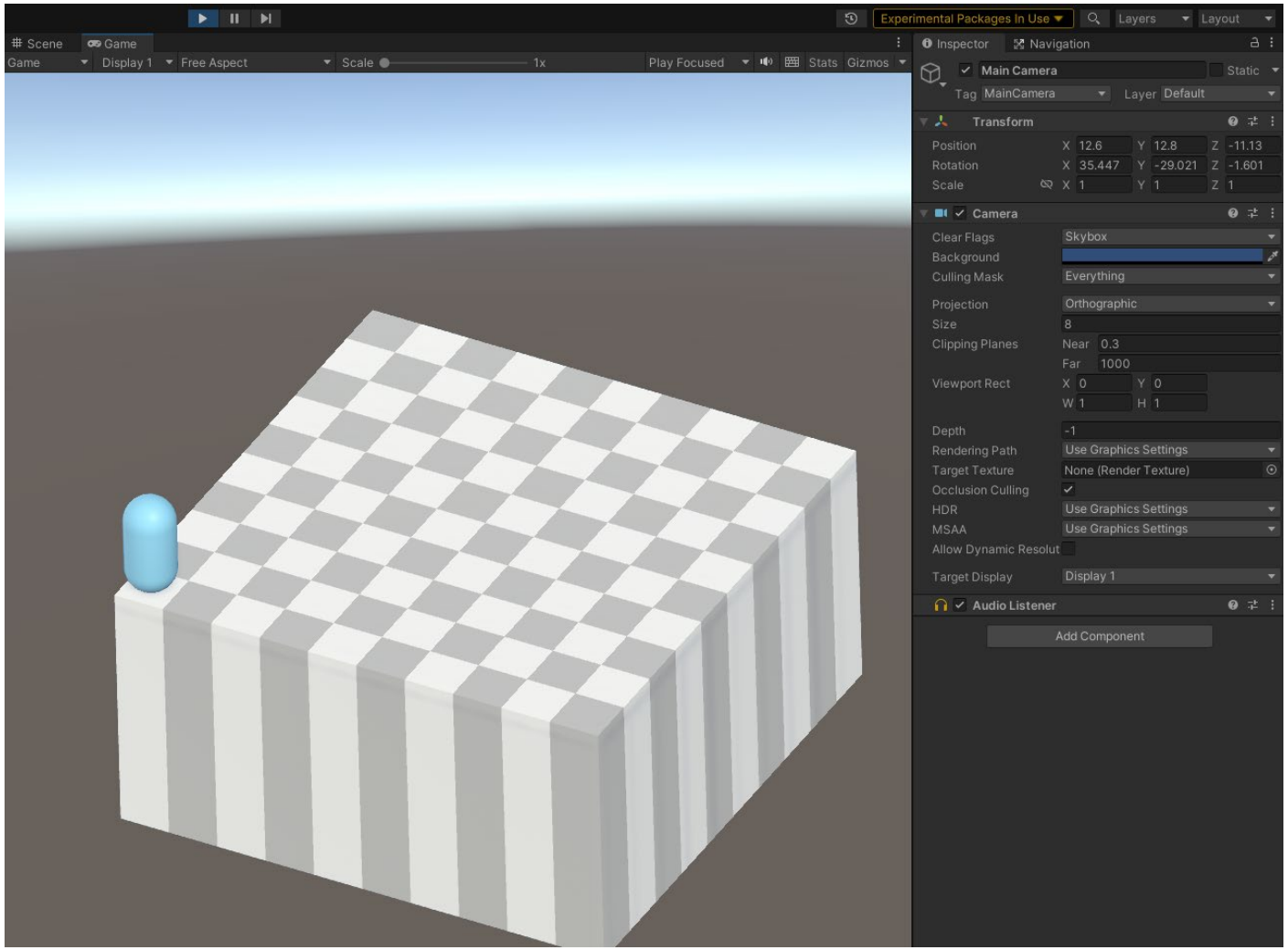
The AStarMap script

- COPY the following code to the AStarMap script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AStarMap : MonoBehaviour
{
    public AStarNode pfb_astarNode;

    [SerializeField]
    Vector2Int mapSize = new Vector2Int(10, 10);

    public AStarNode[,] astarNodeMap;
    public List<AStarNode> astarNodes;

    void Awake()
    {
        astarNodeMap = new AStarNode[mapSize.x, mapSize.y];

        for (int x = 0; x < mapSize.x; x++)
        {
            for (int y = 0; y < mapSize.y; y++)
            {
                AStarNode n = Instantiate<AStarNode>(pfb_astarNode);
                n.transform.position = new Vector3(x, pfb_astarNode.transform.position.y, y);
                n.gameObject.name += string.Format("_{0}_{1}", x, y);
                if (x % 2 != y % 2) { // to create a chessboard feel
                    MeshRenderer mRend = n.GetComponent<MeshRenderer>();
                    Material m = mRend.material; // create a copy of the material. Check the documentation
                    m.color = new Color(0.8f, 0.8f, 0.8f);
                    mRend.material = m;
                }
                astarNodeMap[x, y] = n;
                astarNodes.Add(n);
            }
        }

        pfb_astarNode.gameObject.SetActive(false);
    }
}
```

Generate the map

- Set up a Orthogrphic camera
- Press PLAY
- You should see the following
- In the following slides, ADD the scripts as specified
- The added scripts are implemented the A* Algorithm



The AStarMap script for A*

- ADD the following code to the AStarAgent script

```
    [SerializeField]
    float heightCostMultiplier = 2.0f;
    [SerializeField]
    float maxHeightDifference = 0.5f;

    void Awake()
    {
        …

        Vector2Int[] neigbors = new Vector2Int[] { new Vector2Int(1, 0), new Vector2Int(-1, 0), new Vector2Int(0, 1), new Vector2Int(0, -1) };
        for (int x = 0; x < mapSize.x; x++)
        {
            for (int y = 0; y < mapSize.y; y++)
            {
                AStarNode n = astarNodeMap[x, y];
                foreach (Vector2Int nOffset in neigbors)
                {
                    int _x = x + nOffset.x;
                    int _y = y + nOffset.y;
                    if (_x >= 0 && _x < mapSize.x && _y >= 0 && _y < mapSize.x)
                        n.neighbors.Add(astarNodeMap[_x, _y]);
                }
            }
        }

        pfb_astarNode.gameObject.SetActive(false);

    }
…
```

The AStarMap script for A*

- Continue…

```
float computeDistance(AStarNode from, AStarNode to)
{
    Vector3 flatDelta = to.transform.position - from.transform.position;
    flatDelta.y = 0;

    float flatDist = flatDelta.magnitude;

    float heightDifference = Mathf.Abs((to.transform.position - from.transform.position).y);

    if(heightDifference > maxHeightDifference)
    {
        return float.PositiveInfinity;
    }

    return flatDist + heightDifference * heightCostMultiplier;
}
```

The AStarMap script for A*

- Continue…

```
    public List<AStarNode> openSet;
    public List<AStarNode> Compute(AStarNode from, AStarNode to)
    {
        openSet = new List<AStarNode>(); // you should use a better data structure. Specifically, you find an equivalent of PriorityQueue: https://learn.microsoft.com/en-us/dotnet/api/system.collections.generic.priorityqueue-2?view=net-7.0

        astarNodes.ForEach(n => {
            n.distance = float.MaxValue;
            n.from = null;
        });

        from.distance = 0;
        from.heuristicCost = (to.transform.position - from.transform.position).magnitude;
        openSet.Add(from);
        AStarNode crt = null;
        while (openSet.Count > 0)
        {
            crt = openSet[0];
            openSet.RemoveAt(0);
            if (crt == to)
                break;

            foreach (AStarNode nbr in crt.neighbors)
            {
                float nbrDist = crt.distance + computeDistance(crt, nbr);

                if (nbrDist < nbr.distance && nbrDist < float.PositiveInfinity)
                {
                    nbr.from = crt;
                    nbr.heuristicCost = nbrDist + (to.transform.position - nbr.transform.position).magnitude;

                    int insertIdx = openSet.Count;
                    for (int i=0; i < openSet.Count; i++)
                    {
                        if (nbr.heuristicCost < openSet[i].heuristicCost)
                        {
                            insertIdx = i;
                            break;
                        }
                    }
                    openSet.Insert(insertIdx, nbr);
                }
                nbr.distance = nbrDist;
            }
        }
```

The AStarMap script for A*

- Continue…

```
        List<AStarNode> path = new List<AStarNode>();
        if (crt == to)
        {
            while (crt != from)
            {
                path.Add(crt);
                crt = crt.from;
            }

            path.Add(from);
            path.Reverse();

        }
        else
        {
            Debug.LogWarning("Fail to find path to target");
            path = null;
        }

        return path;
    }
```

The AStarAgent script for A*

- COPY the following code to the AStarAgent script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AStarAgent : MonoBehaviour
{
    AStarNode crtNode;
    List<AStarNode> pathToWalk;

    [SerializeField]
    float speed = 3;

    [SerializeField]
    Camera cam;

    AStarMap map;

    public Vector3 SurfacePoint
    {
        get
        {
            return transform.position - new Vector3(0, transform.localScale.y, 0);
        }
    }

    private void Awake()
    {
        map = Object.FindObjectOfType<AStarMap>();
    }

    // Start is called before the first frame update
    void Start()
    {
        crtNode = map.astarNodeMap[0, 0]; // why this is at Start() not Awake()?
    }
…
```

The AStarAgent script for A*

- Continue…

```
…
    void Update()
    {
        if (Input.GetMouseButtonDown(0))
        {
            Ray r = cam.ScreenPointToRay(Input.mousePosition);
            RaycastHit hit;
            if(Physics.Raycast(r, out hit))
            {
                AStarNode target = hit.collider.GetComponent<AStarNode>();
                if(target != null)
                {
                    pathToWalk = map.Compute(crtNode, target);
                }
            }
        }
        if (pathToWalk != null)
        {
            if ((pathToWalk[0].SurfacePoint - SurfacePoint).magnitude < 0.1f) // reach target
            {
                crtNode = pathToWalk[0];
                crtNode.transform.up = Vector3.up;
                pathToWalk.RemoveAt(0);

                if (pathToWalk.Count == 0)
                {
                    pathToWalk = null;
                }
            }

            if (pathToWalk != null)
            {
                Vector3 delta = pathToWalk[0].SurfacePoint - SurfacePoint;
                transform.forward = delta.normalized;

                transform.position += transform.forward * speed * Time.deltaTime;
            }
        }
    }
}
```

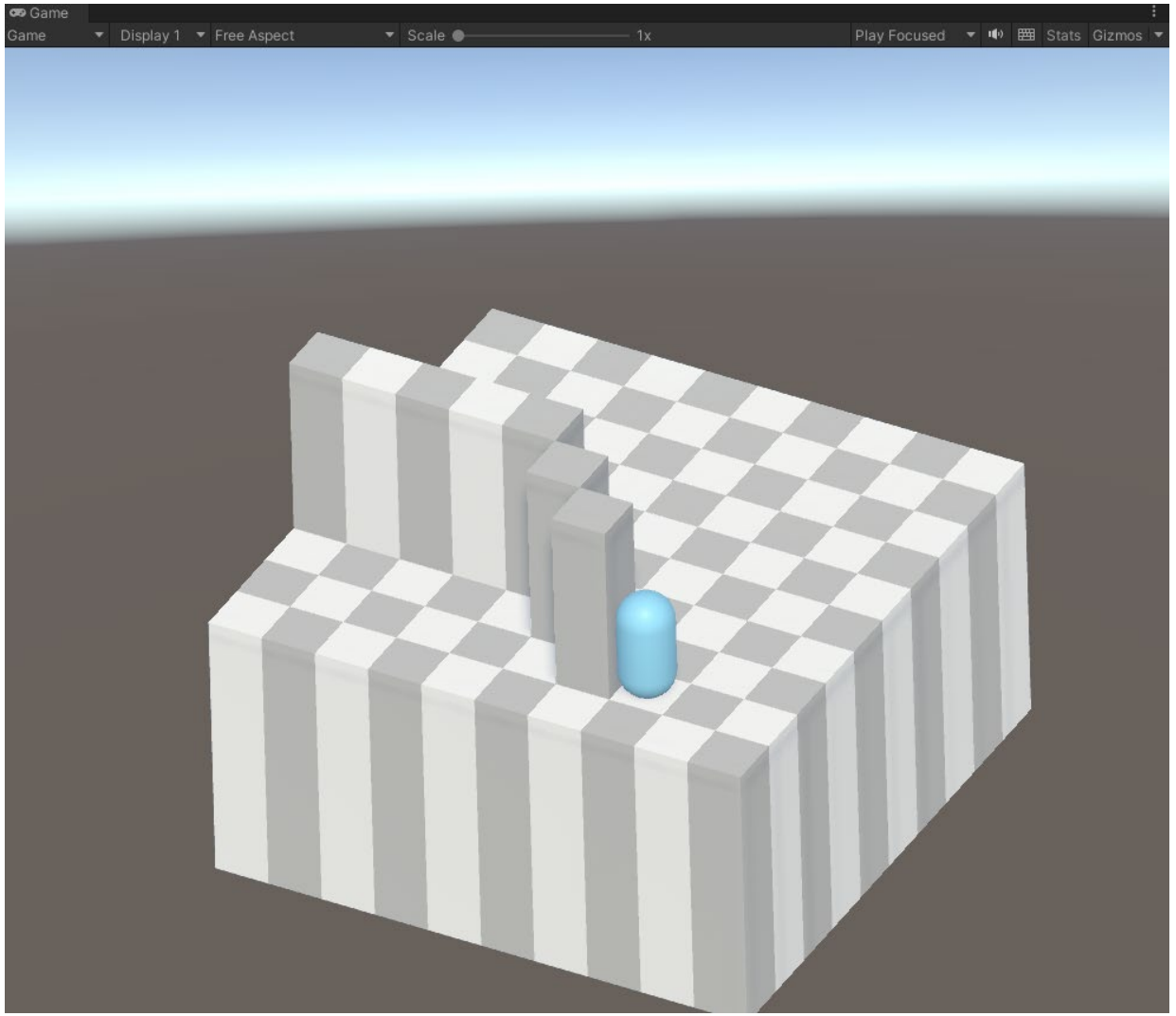
The AStarNode script for A*

- COPY the following code to the AStarAgent script

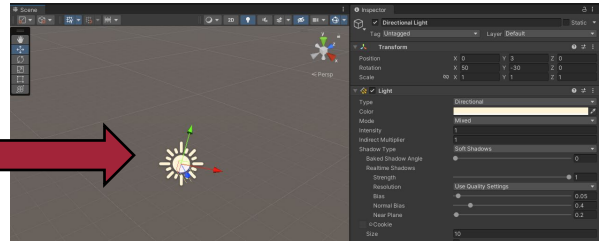
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AStarNode : MonoBehaviour
{
    public AStarNode from;
    public float distance;
    public float heuristicCost;

    public List<AStarNode> neighbors = new List<AStarNode>();

    AStarMap map;

    void Awake()
    {
        map = Object.FindObjectOfType<AStarMap>();
    }

    public Vector3 SurfacePoint
    {
        get {
            return transform.position + new Vector3(0, transform.localScale.y * 0.5f, 0); // why we need this?
        }
    }
}
```

Generate the map

- By this point, you should be able to click a nodes
- And the agent will move
- You can try to manually move some of the nodes vertically to change the movable area
- However, we want to better visualize
- We can use **Gizmos** (https://docs.unity3d.com/ScriptReference/Gizmos.html)
- You have actually seen them for awhile!



The light icon is a gizmo



The AStarNode Gizmos script

- ADD the following code to the AStarAgent script

```
private void OnDrawGizmos()
{
    Color c = Color.grey;

    if (map.openSet != null && map.openSet.Contains(this)) // what is "this"?
    {
        c = Color.yellow;
    }
    else if (distance >= float.PositiveInfinity)
    {
        c = Color.red;
    }
    else if (distance < float.MaxValue)
    {
        c = Color.green;
    }
    c.a = 0.4f;

    Gizmos.color = c;
    Gizmos.DrawCube(SurfacePoint, new Vector3(1, 0.1f, 1));
}
```

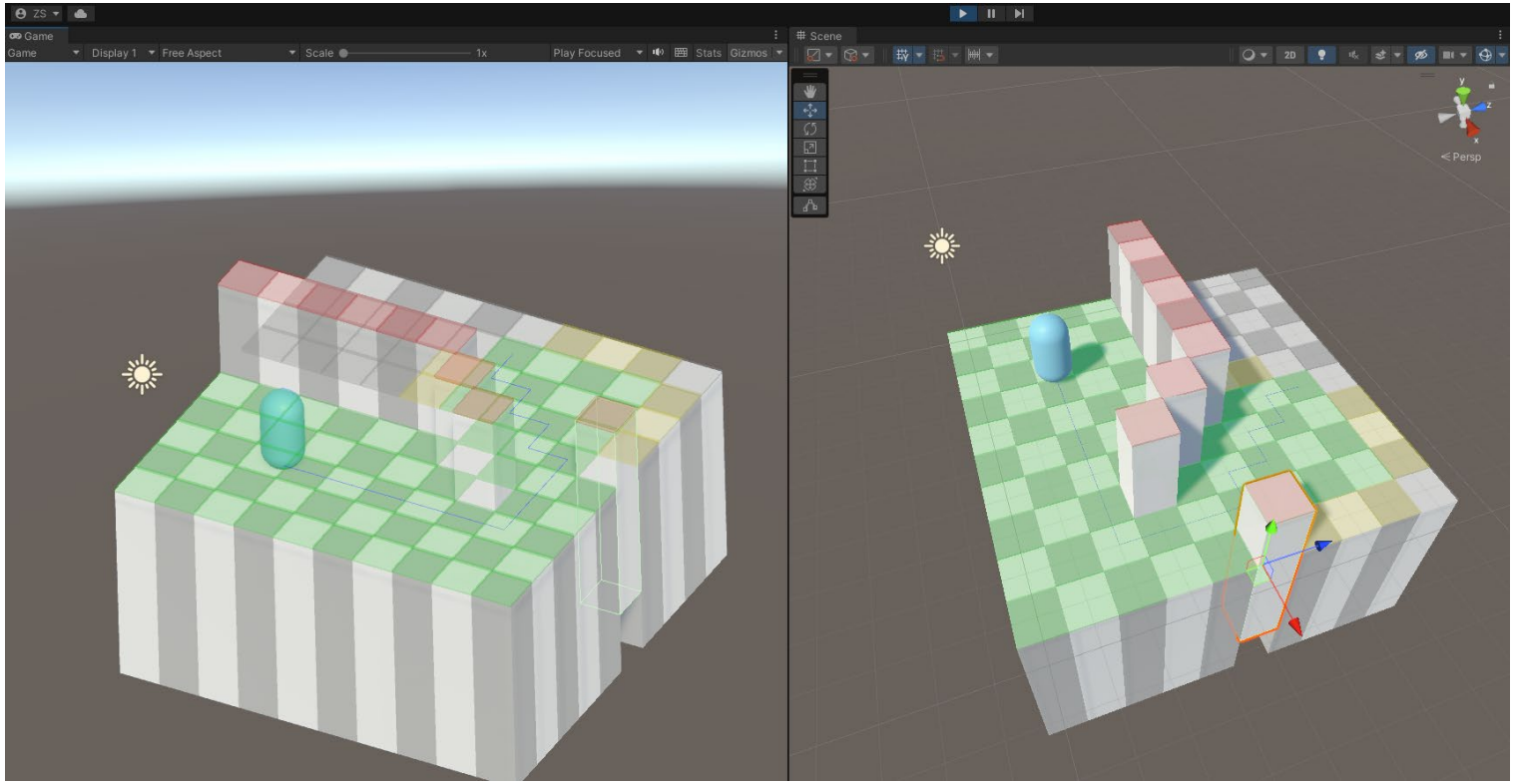
The AStarAgent Gizmos script

- ADD the following code to the AStarAgent script

```
private void OnDrawGizmos()
{
    Gizmos.color = new Color(0, 0, 1, 0.5f);
    for(int i=0; pathToWalk != null && i < pathToWalk.Count; i++)
    {
        Vector3 prev = i == 0 ? SurfacePoint : pathToWalk[i - 1].SurfacePoint;
        Gizmos.DrawLine(prev, pathToWalk[i].SurfacePoint);
    }
}
```

Generate the map

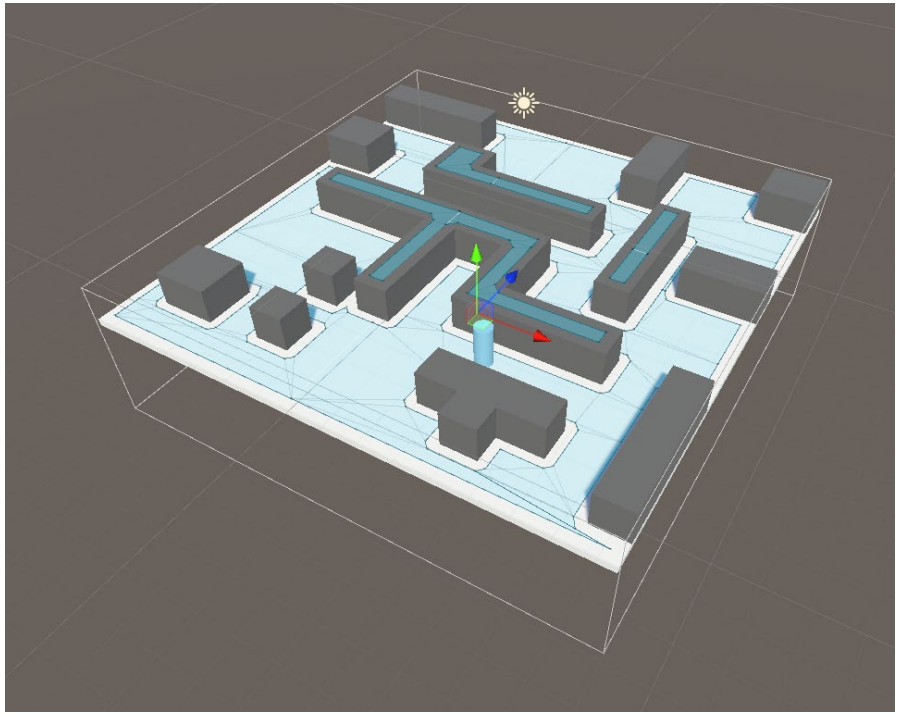
- Press PLAY
- You can press the Gizmos button in both Game and Scene window to show or hide the gizmos.
- Try playing with the settings, particularly HeightCostMultiplier
- Does it ring a bell with NavMeshLink?



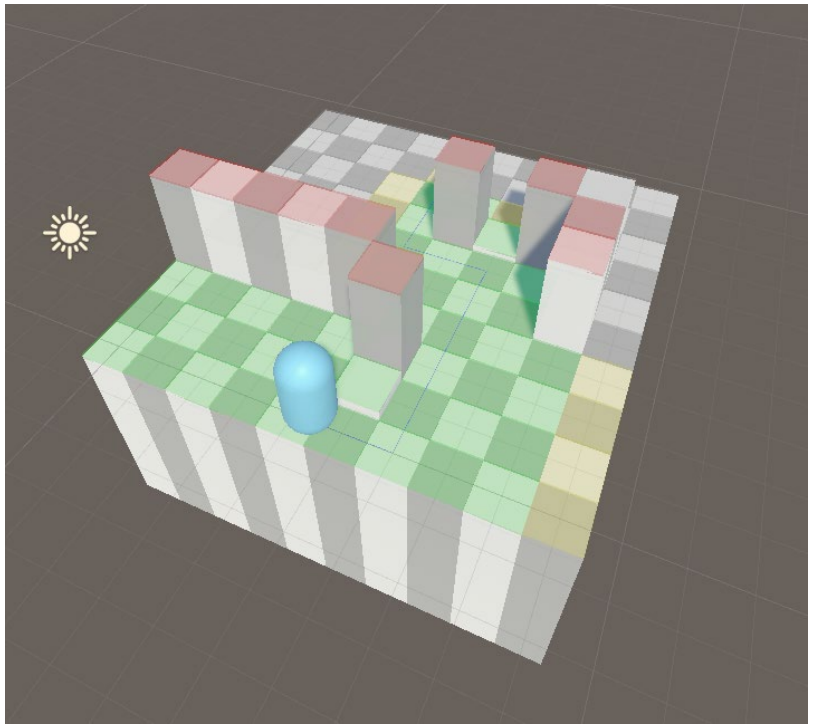
For your thought

- What are their differences?
- Can we mix them together?

NavMesh



A* Algorithm



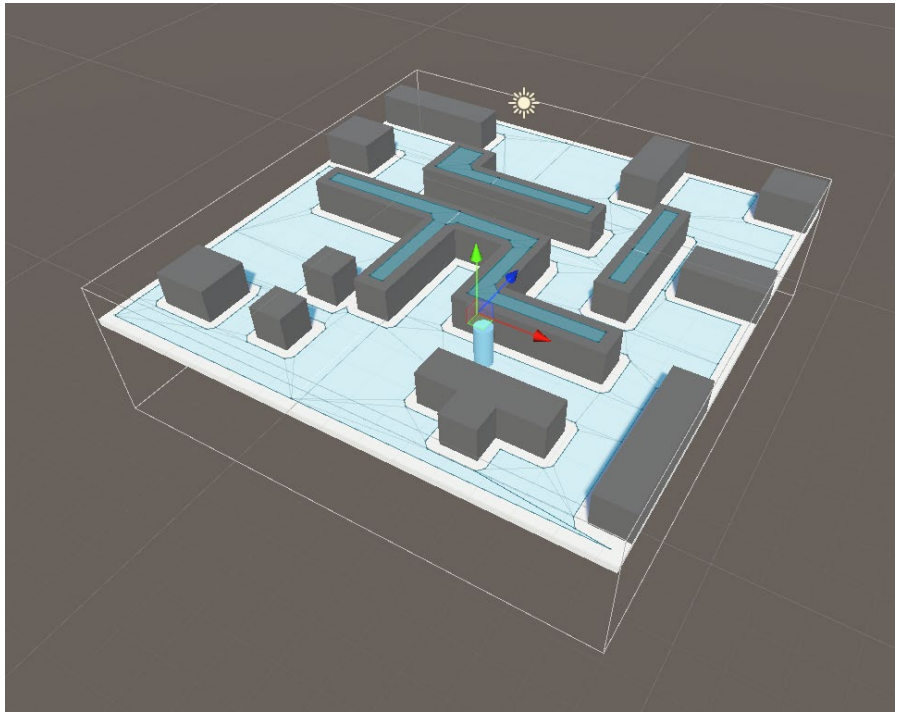
Bonus

Questions and additional materials

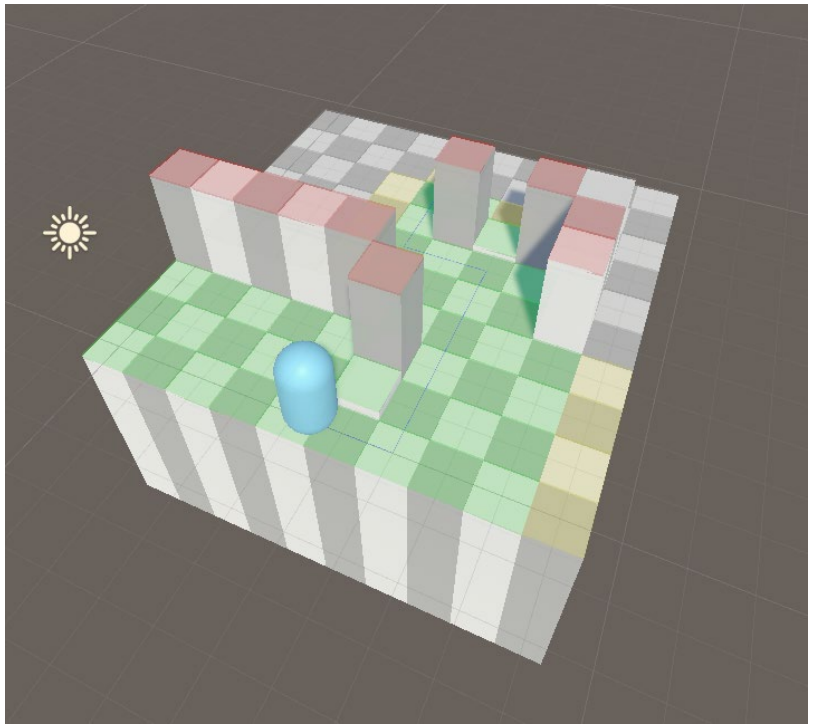
For your thought

- What are their differences?
- Can we mix them together?

NavMesh



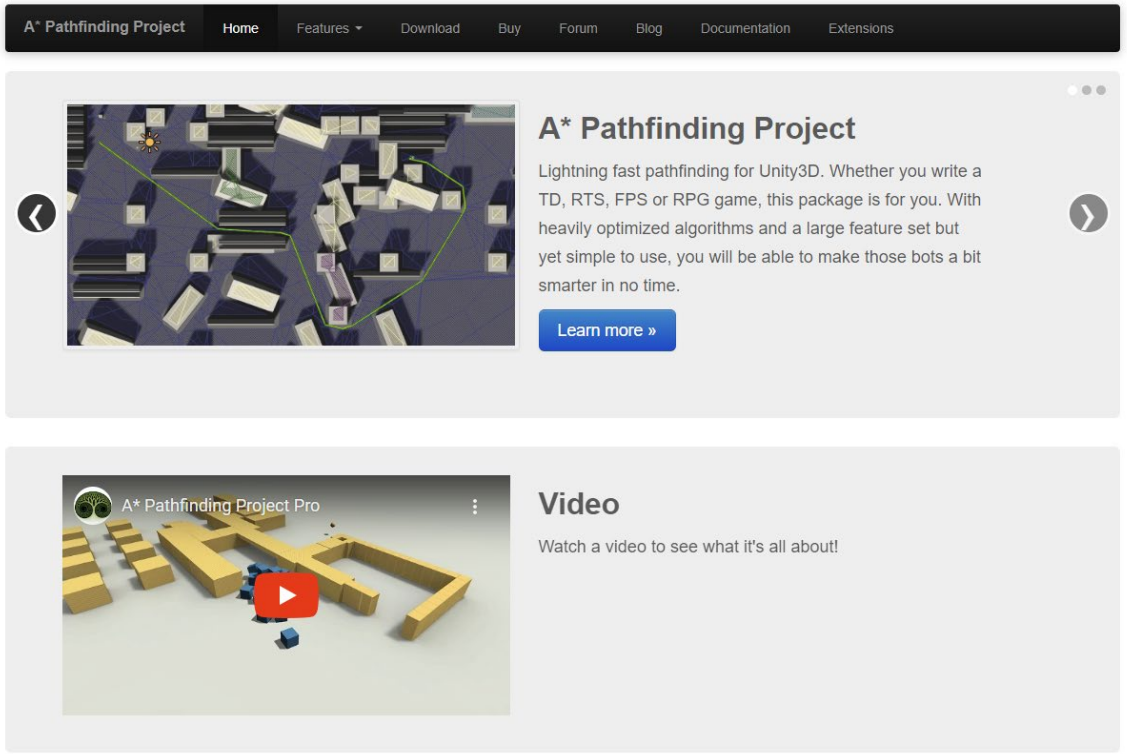
A* Algorithm

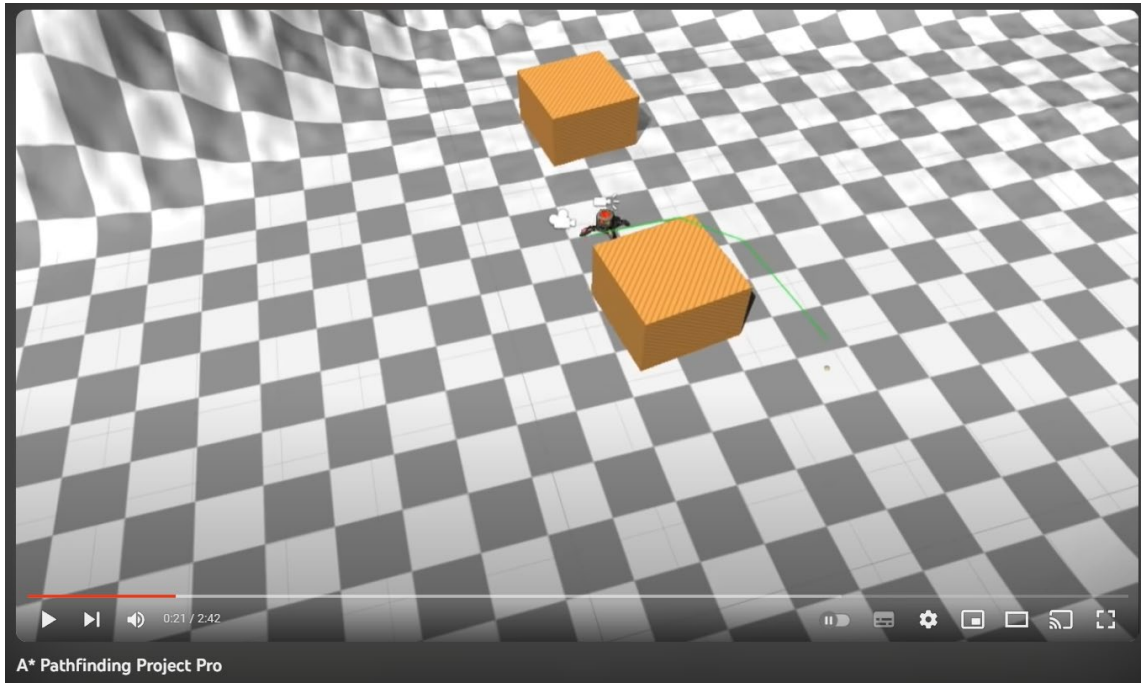


Important asset to consider

- Don’t implement everything on your own → there are existing assets/projects to use
- Don’t always rely on assets as well → existing assets may not be suitable for your case (e.g. 2D game)

A* Pathfinding Project



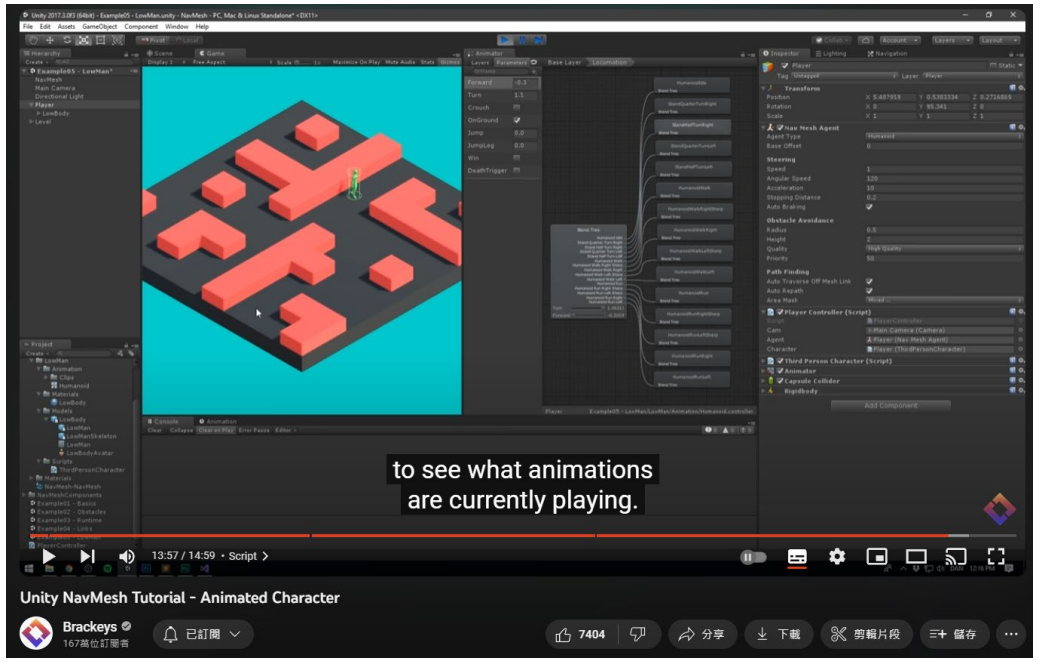


- https://arongranberg.com/astar/

More for your own learning

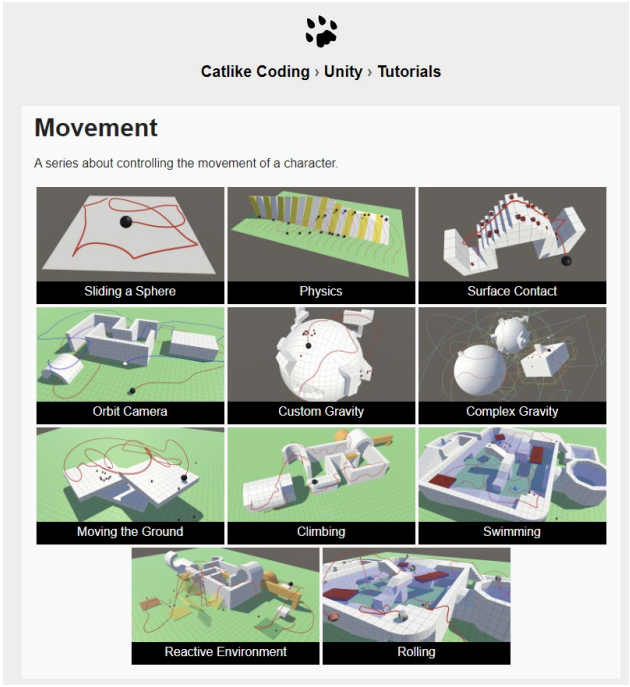
- The labs are meant for you to kickstart your learning
- And the TAs and I are here for you to ask questions
- There are many materials outside of the lab → find the best one for you and learn more

NavMesh



- https://www.youtube.com/watch?v=CHV1ymlw-P8

Catlike Coding



- https://catlikecoding.com/unity/tutorials/movement/

Thank you ☺