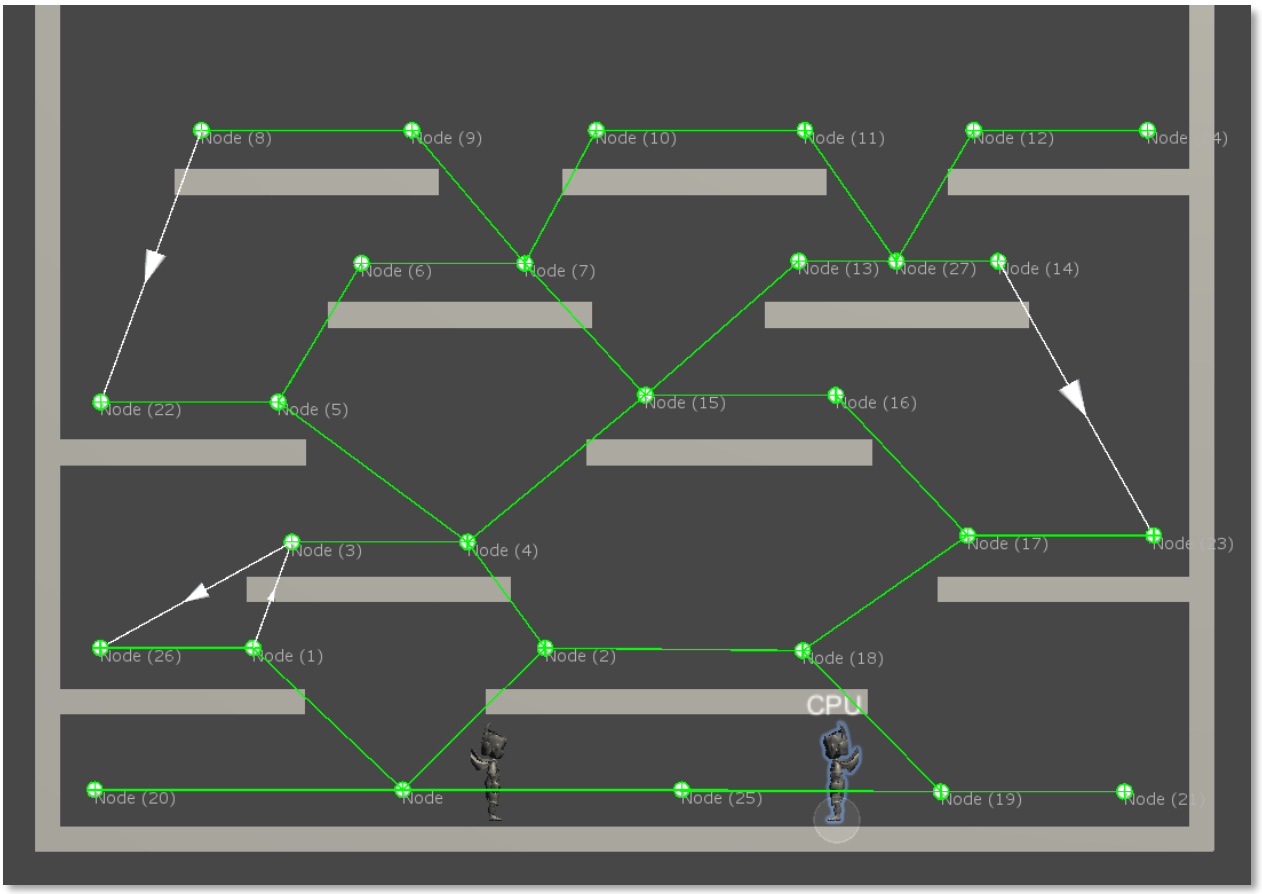
MESH PATHFINDING FOR PLATFORMER

DOCUMENTATION

First of all, thank you for purchasing this asset through Unity Assets Store. This documentation will guide you through the usage in order for this tool to work properly in your game project.

WHAT IS MESH PATHFINDING FOR PLATFORMER?

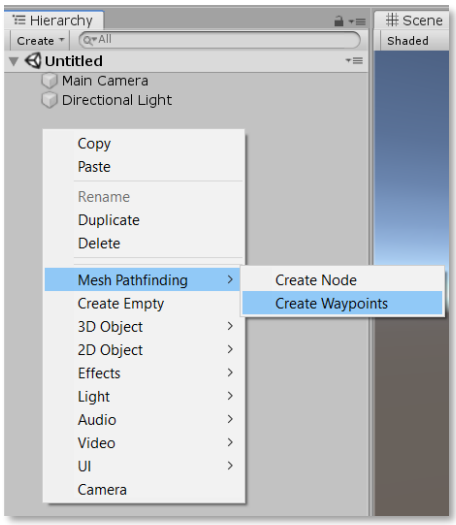


Mesh Pathfinding for Platformer is an easy to use pathfinding system and tool for Unity. This tool provides mesh graph creation instead of grid, especially made to create specific waypoints of platformer game character. The system itself is very robust, and it will not drive you to use a defined movement method, but instead you can implement it on your own movement method!

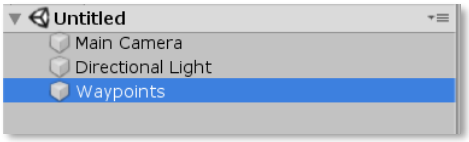
HOW TO SET UP

1. Creating Mesh Waypoints

Select **GameObject** from menu bar, or right click on hierarchy, and select **Mesh Pathfinding -> Create Waypoints.**

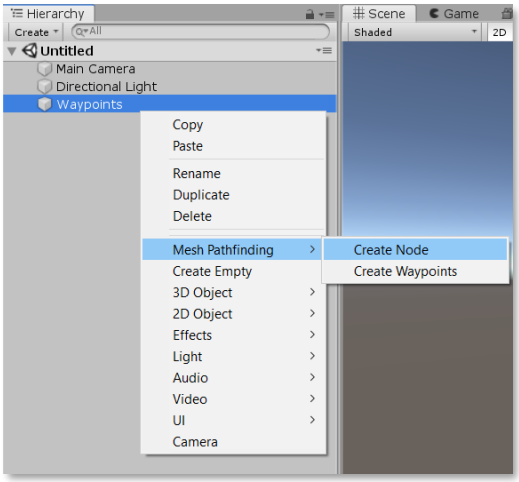


The Waypoints GameObject will be created on the scene. This will be the root GameObject that stores all the Node GameObjects that we'll create later on.

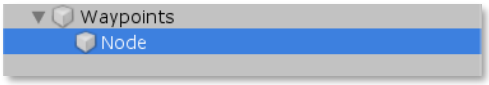


2. Creating Node

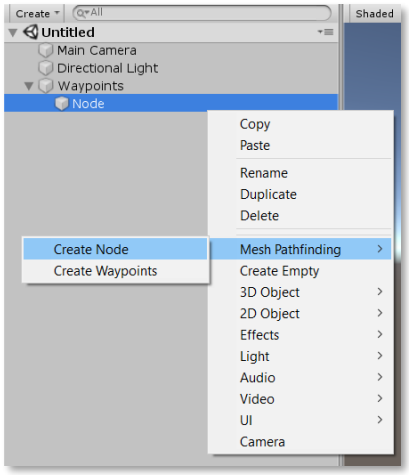
Select **GameObject** from menu bar, or right click on hierarchy, and select **Mesh Pathfinding -> Create Node.**

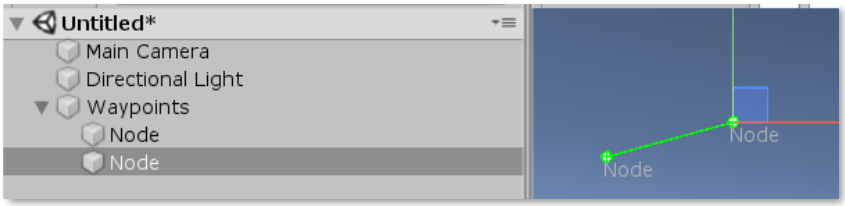


A Node will be created, and will automatically search for Waypoints GameObject even if you're not currently selecting the Waypoints GameObject. Otherwise if there is no Waypoints GameObject was created in the scene, then it will be created automatically.

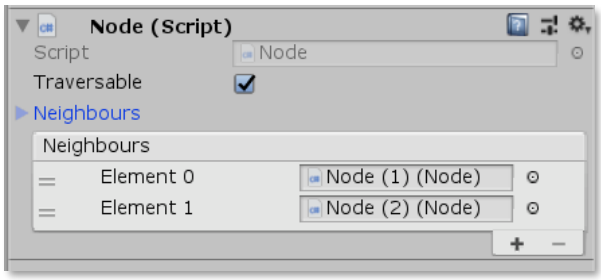


If you click on Create Node again while selecting another Node, then that newly created Node will be automatically connected each other with the previously selected Node, as their neighbour.





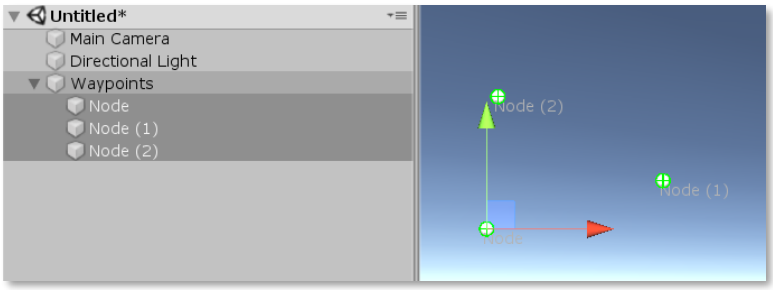
You can also just duplicate the previously created nodes (**Ctrl + D**) and set the connections manually in inspector.

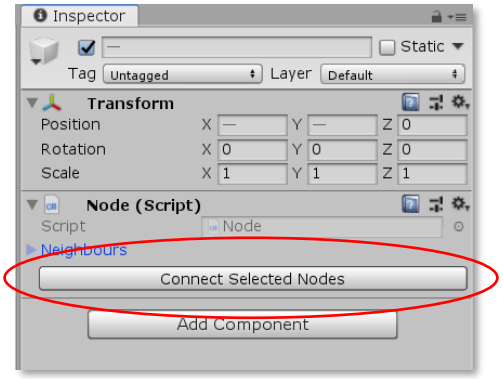


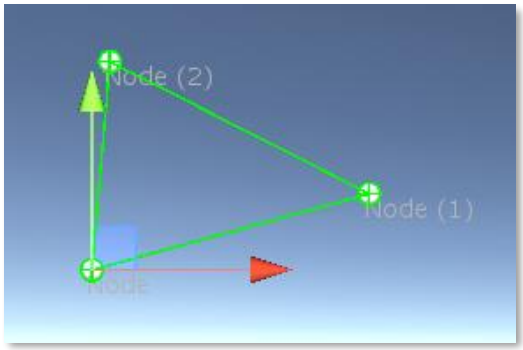
Traversable is set true by default. If it is disabled, then the node will be ignored in path calculation. For example, when there's an obstacle in runtime, that blocked the node, then you can set traversable to false, so that no pathfinding user can go through the obstacle.

3. Connecting Nodes to Each Other

After you created several Nodes, you can connect them all to each other by selecting all of them, and then click on Connect Selected Nodes button in the inspector.



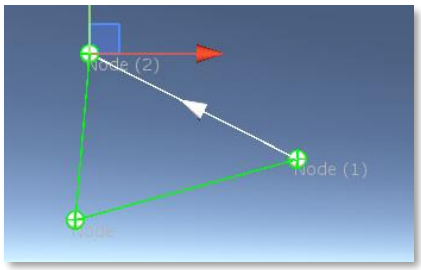




You can also manually assign the neighbours connection by modifying the neighbours list in the inspector.

4. Connecting Nodes in One Direction

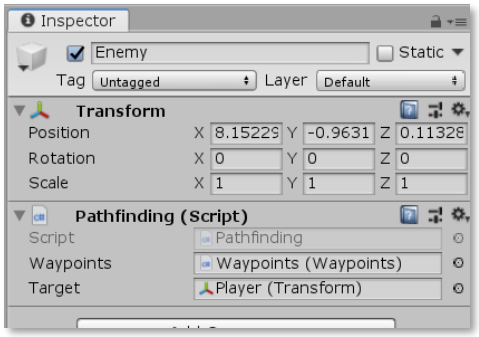
A node connection can also be one-directional (can't traverse back) by manually assigning a specific node as a neighbour in a specific node. On the scene window, you can see that one-directional connection is visualized differently with an arrow that shows which direction it goes to.



For example as we can see on the image above, the pathfinding user can traverse from Node (1) to Node (2), but it can't go back from Node (2) to Node (1).

5. Creating Pathfinding User

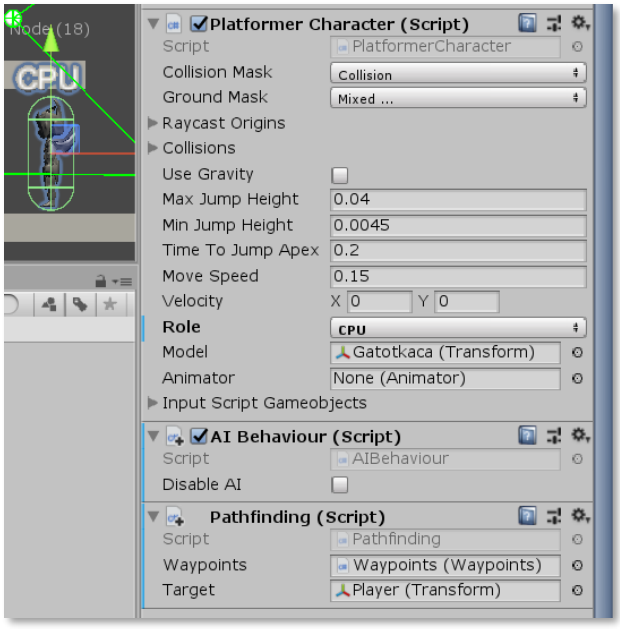
In order to use the pathfinding, select the GameObject that you want use, and add **Pathfinding** component in inspector.



Assign the waypoints to be used.

Specify the target transform that's going to be the goal of the calculated path.

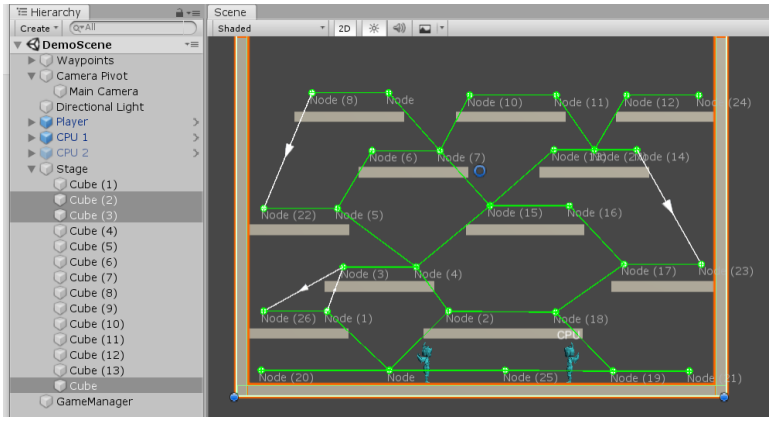
In the demo scene, we have provided an example of how to utilize the generated path result to be used along with the movement script in a file called **AIBehaviour.cs**.



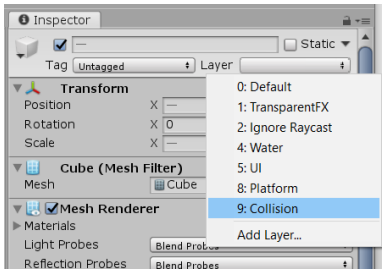
It's worth noting that a Waypoints GameObject can be used by multiple Pathfinding users. In the demo scene we've also provided an example, by enabling the **CPU 2** GameObject in the demo scene.

Setting Up Layers in Demo Scene

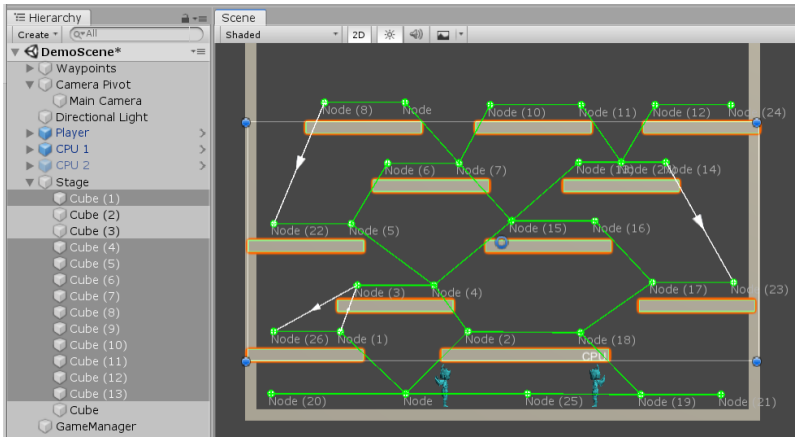
Select the longest cubes that wrap around the stage, just like the picture below.



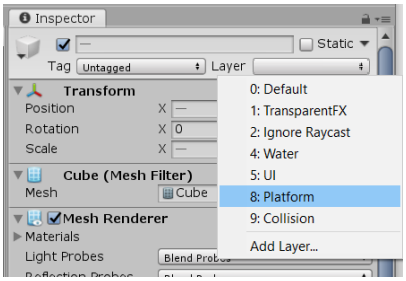
Set the layer of those objects to be **Collision**. You may have to Add Layer first before you can select it.



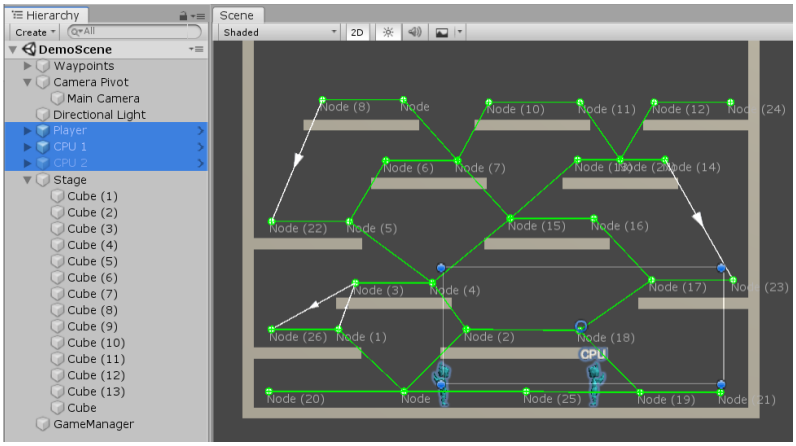
Select the all the flying cubes that used as platforms.



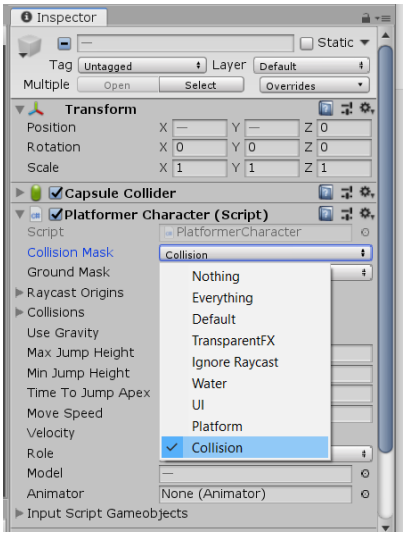
Set the layer of those objects to be **Platform**. You may have to Add Layer first before you can select it.



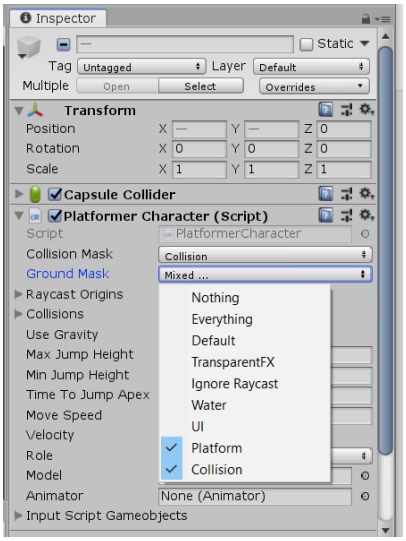
Now that we have assigned the objects in layer Collision and Platform, we have to specify the layer mask in each character. Select **Player**, **CPU 1**, and **CPU 2** gameobject in hierarchy pane.



Specify the collision layer mask by clicking the dropdown and select **Collision**.



Specify the ground layer mask by clicking the dropdown and select **Platform** and **Collision**.



Now you can try to play the demo scene!

Implement in Your Own Way!

To implement the pathfinding on your own movement script, just create a component script, and inherit the class PathfindingUserBase. Set the target using SetTarget method, and call StartFindPath method to start generating the path to the target. Alternatively, you can also call StartFindPath(false) to prevent the looping path generation, and specify when to call it manually.

```
using UnityEngine;
using Calcatz.MeshPathfinding;

public class PathfindingUserExample : PathfindingUserBase {
    public Transform target;

    void Start() {
        pathfinding.SetTarget(target);
        pathfinding.StartFindPath();
    }

    void FixedUpdate() {
        Node[] path = pathfinding.GetPathResult();
        if (path != null) {
            //Do something with the path result
            //For example, move towards path[0].transform.position

        }
    }
}
```

You can add your own movement logic using the path information provided. For example, how can your character reach the closest node? Should he just run towards it? Or should he jump because it's too high, or can he just fly towards it?

SUPPORT

If you have any questions or difficulties regarding this tool, you can send an email to affan@calcatz.com and we'll gladly help you. Thank you for having this asset, cheers!