

**Proce
Path - 3D**




Documentation

ProcePath

Setup  p.3

PathFindingComponent  p.4

PathSettingOnCharacter  p.7

How to debug  p.10

Setup

To start using ProcePath you'll need to have in your scene at least the prefab <<PathFindingController>> Or you can set up this by opening **Window > ProcePath > Setup**

This is the only thing you need to be able to use it.

You can take a look on the sample scene "SamplePathfinding3D".

It contain a sample character "SampleIA" using "ProcePath" by clicking on the ground on "Play"



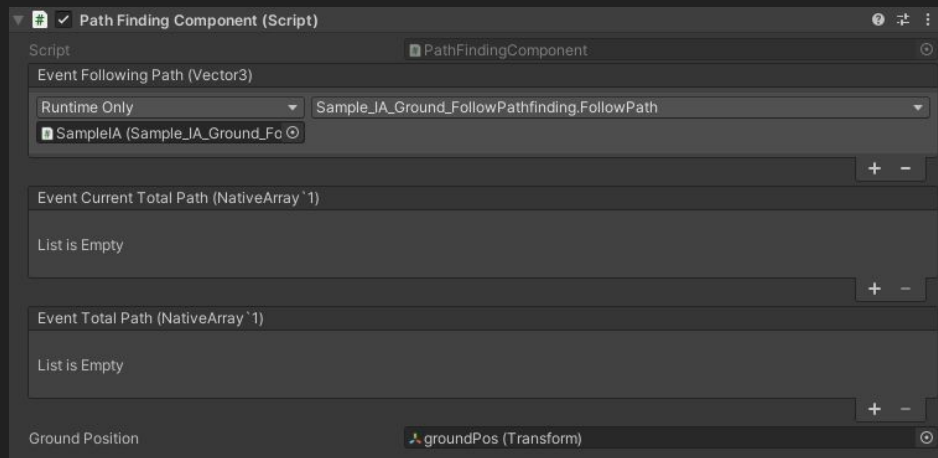
On his inspector you can find this variables : "MaxNodeTotalCalculOnFrame", it's corresponding of maximum nodes calcul on a frame. Bigger is this number, more the performance may be required (depending of the distance).

Max Node Total Calcul On Frame

3500

PathFindingComponent (1/3)

To help you using “ProcePath” on your “Character”, you can add the component “PathFindingComponent”, it will help you to get all the path needed to the target and debugging:



- “EventFollowPath” is an event that will send the current direction between the current position to the next position of the path
- “EventCurrentTotalPath” will send an “native<Vector3>” of the current path position to the target already calculated
- “EventTotalPath” will send an “native<Vector3>” of the total path position to the target

Ground Position corresponding to a transform at the bottom of your character.

PathFindingComponent (2/3)

On your own script, the minimum that your script have to contain is something like this :

- You'll have to get your "PathFindingComponent" (variable or GetComponent)
- You'll need to have a "PathSettingOnCharacter" variables

```
[SerializeField] private PathSettingOnCharacter characterPathSetting; // PathCharacterSetting.asset
[SerializeField] private PathFindingComponent path; // SampleIA (PathFindingComponent)

private void CreateNewTargetPointToGo(Vector3 positionTarget)
{
    Vector3 transformPos = transform.position;
    path.CreateNewPath(characterPathSetting, startPosition: transformPos, endPosition: positionTarget, CallbackPathTravel);
}
```

Now you can create a path by calling "CreateNewPath" from PathFindingComponent, by referencing

- A "PathSettingOnCharacter" variables
- The current character position and the Target position

You can stop the current calcul by calling "ResetPathComponent" from "PathFindingComponent"

PathFindingComponent (3/3)

When you use “[CreateNewPath](#)” you can also put a method callback to be update of the current status of the calcul for the pathfinding, like “[CallBackPathTravel](#)”.

```
private void CreateNewTargetPointToGo(Vector3 positionTarget)
{
    Vector3 transformPos = transform.position;

    pathProcess = PathFindingController.PathProcess.WIP;
    path.CreateNewPath(characterPathSetting, startPosition: transformPos, endPosition: positionTarget, CallBackPathTravel);
}

Frequently called 1 usage
private void CallBackPathTravel(PathFindingController.PathProcess obj)
{
    pathProcess = obj;
    path.ResetPathComponent(); // sample
}
```

There is 4 different status : Done - Wip - Failed, and FailedDueToMapSize, the last one is telling you that the distance of the travel is bigger than your “map size”. But don’t worry, in that case it will continue to calculating by creating new calcul until it reach the target (it keep the last point nearest to the target).

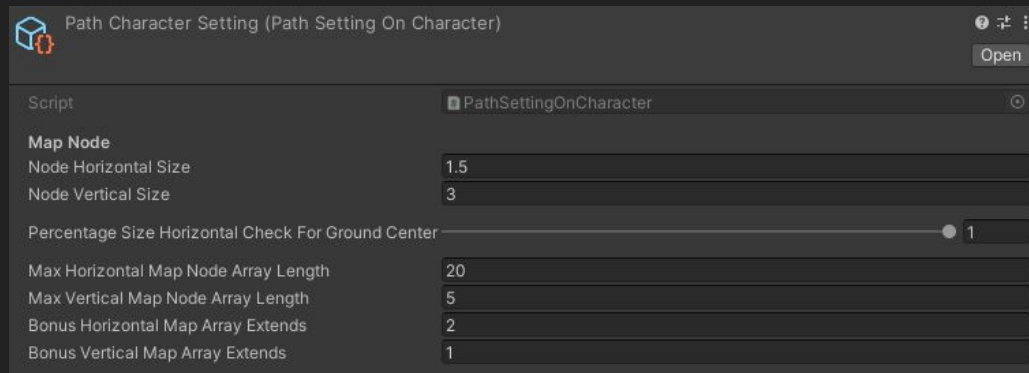
If you need to stop the calcul of the pathfinding you can call “[ResetPathComponent](#)” like the sample above.

PathSettingOnCharacter (1/3)

PathSettingCharacter is a scriptable, you can create new one on project by going to :
"Asset -> create -> scriptable -> ProcePath -> CharaSetting

This first part is to configure your "nodes":

- Size of horizontal and vertical node (better if it's equal or bigger of your character collider)
- The percentage of nodes size to check the ground of the nodes
- The maximum size of the map in horizontal and vertical way
- And a bonus size in case



Note that the map size is corresponding to the distance of a point A and B (character to target). If the map size is smaller (with the node size) than the distance, it will create multiple map to reach the target (bigger map size will have a better precision but it will cost more performance, you will have to find the balance you need).

If the map size is smaller than his maximum, it will add your Bonus size, otherwise it will clamp to the maximum.

PathSettingOnCharacter (2/3)

On second you can configure your “Path Update”

Path to Target	
Distance To Check Between Position And Next Node	0.1
Delay Checking Node Path State	0.5
Max Case Distance To Check	10

- You can configure the maximum distance you need before going to the next node of your path (using for the event “**EventFollowingPath**” on your “**PathFindingComponent**”)
- A delay to check if the nodes of your path have new status or not (if obstacle have appeared on your path for example). It will automatically recalculate a new path if needed.
- And the maximum of Nodes above the current node you want to check.

PathSettingOnCharacter (3/3)

The third and last part, is for your character “abilities”

“On Player”, you can select what kind of move your character can do. (Note that “**PathNodeDirectionType**” is using for the event “**EventFollowingPath**” on your “**PathFindingComponent**”, it will give the direction between the current node to the next one; or the direction between your ground position to the next node)

Layer Setting is to set what kind of layer your character can walk, and what kind of layer is considered obstacles

On Ground settings (not really useful with “**CharacterCanFly**” at true) you can configure what kind of angle your character can walk; the maximum vertical distance that your character can auto climb like edges (only detection, it’s mean that you’ll manage this on your script). “**PreciseVerticalDetectionBetweenTwo**” is the precision (in percentage) between two nodes to check their maximum difference vertical ground (bigger is this number, more it may require performance).

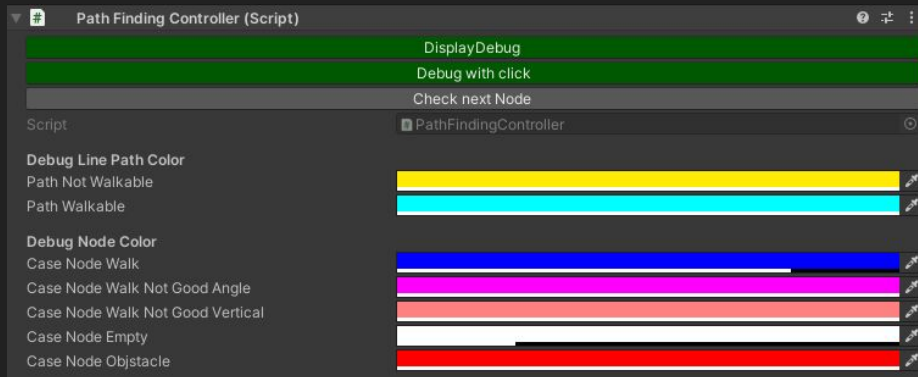
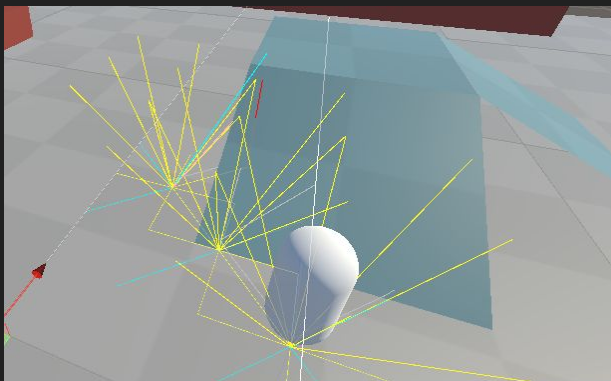
And Fall settings (using if “**CharacterCanFall**” is set at true) to configure how many node case your character can fall⁹

Player Move	
Character Can Use Diagonal Move	<input checked="" type="checkbox"/>
Character Can Use Vertical Path	<input checked="" type="checkbox"/>
Character Can Fly	<input type="checkbox"/>
Character Can Fall	<input checked="" type="checkbox"/>
Path Node Direction Type	Between Last Node To Next Node
Layer Setting	
Ground Layer	Ground
Obstacle Layer	Obstacle
Ground settings	
Max Angle Available To Walk Up	25
Max Angle Available To Walk Down	25
Max Vertical Distance Player Can Auto Climb	0.25
Precise Vertical Detection Between Two Case	11
Fall settings	
Ma Vertical Case Character Can Fall	10

How to Debug (1/3)

There is three ways to Debug the pathfinding, firstable on “**PathFindingController**” on his inspector :

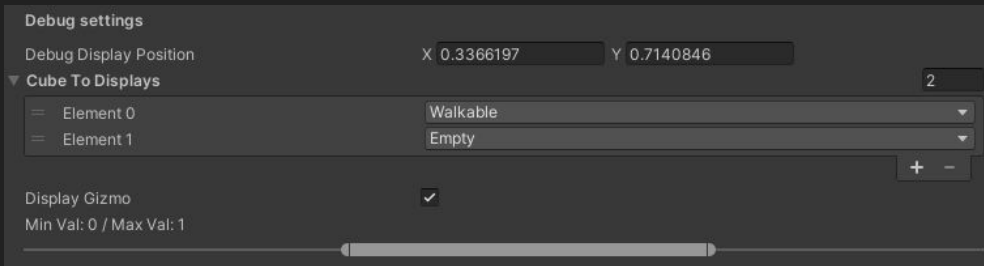
You can see “**Display Debug**”, if it set at true, it will display “**Debug With Click**”, it will allow you to see point by point what the pathfinding it doing (by clicking “**Check next Node**” it will go on the next calcul, etc..). It will give you something like this :



It will show you what direction is a “Walkable” path or not corresponding to the color “**PathNotWalkable**” and “**PathWalkable**”

How to Debug (2/3)

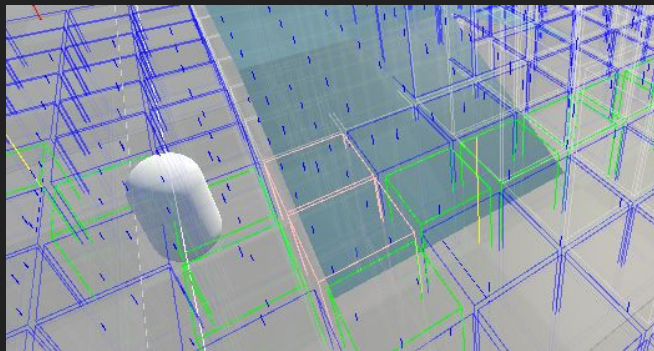
On the second parts, on your “**PathFindingComponent**” on your character you can this on the inspector, this :



Cube to display corresponding of what node type you want to see (with display gizmo set at true), it will show you something like this :

The cube color is corresponding to “**DebugNodeColor**” on “**PathFindingController**”. Except green, corresponding of the path your character have to follow to reach the target point.

The slider on the inspector is to define a “cursor” to see only a parts of all cubes or the totality you set “0 to 1” (quite useful when there is a lot of cubes)

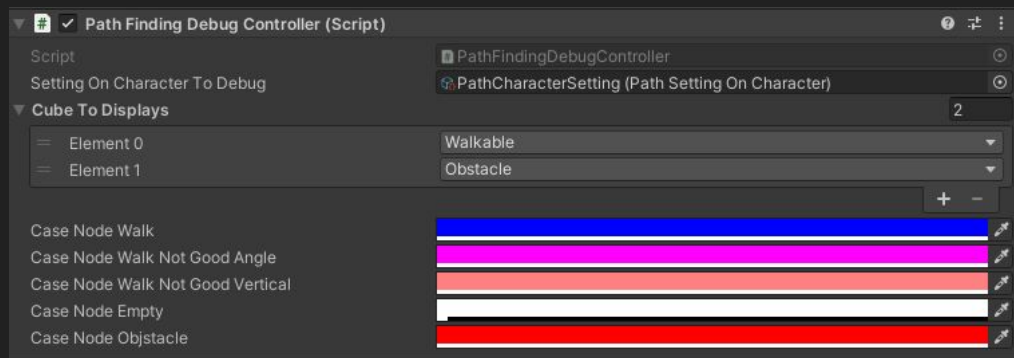
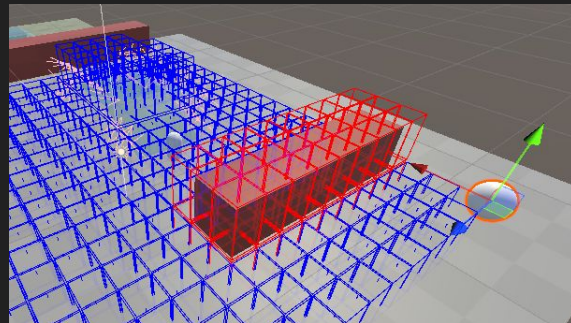


How to Debug (3/3)

On the end, there is another elements, there is the prefab “**PathFindingDebugger**” that you can use. Drag and drop it on the scene (no need to play), and it will show you what your character can walk or not.

This is a “sphere” that you can move around, to see what looks like the nodes on your maps.

On his inspector there is a script “**PathFindingDebuggerController**”. Some settings can be made :



You have to put a reference of your “Character settings” scriptable.

The rest is to define what type of node you want to see, and the color you want to display depending of the status of the node. (the line you see above, display what your character can’t auto climb)