Flexible Pathfinding System for 3D Environments

User Setup Guide

Introduction

This document provides a comprehensive guide on how to use the Flexible Pathfinding System.
This pathfinding tool is composed of two separate systems: Link Generation and Pathfinding Logic.

- **Link Generation** automatically improves the navigation mesh with a thorough and complex web of links which connect all accessible points in the level.
- **Pathfinding Logic** applies a custom pathfinding system based on individual character parameters rather than navigation mesh settings. Physical statistics can be applied to each agent so that the system will choose an appropriate path for each of them.

It is recommended to use these systems together, as with the enhanced mesh and connections the agents have an increased range of options yet their movement is predictable and controlled by their physical statistics.
However the components can be used individually as well.

Before the setup it is important to note that this tool uses the **Unity version 2022.3.52f1** and a Unity AI navigation package **Unity.AI.Navigation**. Make sure the version is correct and that the project contains the appropriate package, otherwise additional adjustments may need to be made.

Below are the setup guides after the installation of this package for each of the systems (link generation and pathfinding logic)

Link Generation

Setup:

1. Add a Controller Object:
    - In the chosen Unity scene, create an empty GameObject to serve as the system's controller.
    - Attach the NavMeshLinksGenerator and NavLinkManager components to this object.
2. Auto-Assign Components:
    - Locate the NavLinkManager component in the Inspector window and click the **Auto Assign Components** button.

- If no errors appear in the console, the setup is successful. If errors do appear, they will indicate the missing elements. Ensure the following are properly configured:
    - The NavMeshLinksGenerator and NavLinkManager components are added to the controller object.
    - Two prefabs exist in the project:
        - LinkPrefab(Standard): For edge-to-edge connections.
        - LinkPrefab(Wide): For dropdown edge-to-floor connections.
    - A NavMeshSurface component exists somewhere in the scene.

3. Manual Adjustments (Optional):
    - In the Inspector window, locate the "Manager Object References and Prefabs" section under the NavLinkManager component.
    - Assign custom prefabs or specify a different NavMeshSurface here. Ensure the link prefabs have their starting points set to (0, 0, 0) relative to their origins.

4. Bake the Mesh:
    - Disable automatic link generation in the NavMeshSurface settings to let the system handle all link creation.
    - Ensure the NavMesh in the level is baked and finished.

5. Generate Links:
    - Use the **Create Links** button in the custom editor in the NavLinkManager component view to generate links automatically based on the system's configuration.

6. Manual Adjustments (Optional):
    - Adjust Links: After Adjusting links manually make sure the links are updated. This function is normally called with the Start() function of NavLinkManager but can also be called by pressing the Update **Links** button.
    - Adding New Links: All manually added links can be added to the system through pressing the button **Update Links.** Note that all manually added links will not be deleted when pressing the **Delete Links** button.

Modifying Link Generation Settings (from NavMeshLinksGenerator component view):

1. Link Raycast Settings
   These settings control the raycasts made when checking for link connections.
    - **enableJumpArcRayCasts**: Enable this to perform box casts that account for potential collisions on jump paths. The size of the box cast depends on the agent's dimensions and the jump arc height. (When disabled the system will perform simple ray casts when checking for collisions on the path of the jump)

- **maxjumpArcHeight**: Adjust the stylized height of the jump arc. The value should represent the distance from the character head to the highest point of the arc. (Used in box casts when enabled)

Note: The raycast box is calculated using the agent's width and height. The height of the box automatically decreases for steeper jumps to ensure collision checks match the character's trajectory.

2. Edge Detection Settings:
   These settings allow for grouping and averaging smaller edges from the baked NavMeshSurface.
   - **minEdgeLength**: Edges shorter than this value will be ignored unless grouped. Set to 0 to disable filtering by length.
   - **maxGroupSize**: Maximum number of small edges that can be grouped into one edge.
   - **minGroupSize**: Minimum number of small edges required to form a group.
3. Basic Link Generation Settings:
   These settings control the primary functionality of link generation and include parameters for the maximum distance and type of links generated:
   - **maxEdgeLinkDistance**: The maximum distance for creating edge-to-edge links. Links won't be generated between edges farther apart than this value. Set to 0 to disable this restriction.
   - **shortLinkDistance**: Short links are generated with fewer restrictions on their angles. Use this to allow more permissive connections for closer edges.
   - **maxDropDownLinkDistance**: The maximum distance to search for dropdown links (edge-to-floor connections). These connections use the LinkPrefab(Wide) prefab and are generated when edges drop directly downwards.
4. Advanced Link Generation Settings:
   These parameters refine the link generation process, focusing on the angle restrictions for both dropdown and edge-to-edge links:
   - Dropdown Link Settings:
     - **dropDownSteepnessModifier**: Controls the steepness (Y component) of the raycast used to detect dropdown connections. Higher values mean considering dropdowns at steeper angles.
     - **dropDownLinkAngles**: Specifies additional angles (rotations along the Y-axis) at which dropdown links will be searched. default values (0f, -30f, 30f mean) the system will look directly below the edge as well as 30 degrees to the sides. Still only one dropdown link will be created per edge.

- Standard Edge-to-Edge Link Angle Restrictions:
    - **standardAngleRestrictionForward**: Limits the angle relative to the forward direction (the direction the edge faces, projected onto the XZ plane).
    - **standardAngleRestrictionUpward**: Limits the angle relative to the upward direction (normal to the surface the edge is part of).

  Note: Only links exceeding both of these restrictions will not be generated.
- Permissive Edge-to-Edge Link Angle Restrictions:
    - **permissiveAngleRestrictionForward**: Similar to standardAngleRestrictionForward but applies to shorter links (<= shortLinkDistance).
    - **permissiveAngleRestrictionUpward**: Similar to standardAngleRestrictionUpward but applies to shorter links (<= shortLinkDistance).

Pathfinding Logic

Setup:

1. Verify Existing Components:
    - Ensure that the current scene has a manager object with **NavLinkManager** and **NavLinkGenerator** present, and that the desired navigation mesh is baked and contains proper NavMeshLinks.
    - Use the **Update Links** button on the NavLinkManager component menu to make sure all links from the current scene are indexed and visible.
2. Create a Navigation Agent:
    - Add a navigation agent to the current scene.
    - Assign its type to match the **AgentType** set in the **NavMesh** settings.
    - Customize its speed and other Unity Navigation parameters as desired.
3. Attach the PlayerController Script:
    - Add the **PlayerController** script to the agent.
    - Assign the **Main Camera** of the current scene to the PlayerController script for raycasting mouse clicks onto the geometry.
    - Select the mouse button from the dropdown menu that will be used to control the agent's movement.
4. Set Up Physical Stats:
    - Attach the **PhysicalStatsLogic** script to the agent.
    - In the **PhysicalStatsLogic** component, configure the following physical parameters:

        - **maxJumpHeight**: Maximum height the agent can jump.
        - **maxJumpDistance**: Maximum distance the agent can jump.
        - **maxDropDistance**: Maximum distance the agent can drop down safely.
    - These values will override the Unity **AgentType** properties for link traversal.

5. Test the System:
    - Start the game. The agent will now navigate the scene using paths that adhere to its physical limitations. It will ignore links that exceed the **PhysicalStatsLogic** parameters.

Customization

- **Modifying Path Request Behavior**:
    - The **PlayerController** script is designed to be simple and should be further customized in the code to fit a given project.
    - Pathfinding requests for the agent are handled by the **NavLinkManager** using public method:

      RequestPath(PhysicalStatsLogic character, Vector3 destination, Action<bool> onPathCalculated = null)

        - **character**: The agent's **PhysicalStatsLogic** component.
        - **destination**: The target position for the agent to navigate to.
        - **onPathCalculated**: An optional callback to handle when the path is successfully calculated.

- **Optimizing Performance**:
    - If a following behavior must be implemented for the character try to avoid frequent path requests. Limit the number of requests per second to prevent overloading the system.