VectorPath

0.1

Generated by Doxygen 1.11.0

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Namespace Documentation

4.1 FlowField Namespace Reference

Classes

- class **Chunk**

  A chunk is used to improve performance. It holds a number of NavNodes and adds some functionality over them. It can have a minimum size of 1 and a maximum size of the flowfield itself. Both extreme cases would not help performance, so you need to find a chunk size that is suitable for your application.
- class **Grid**

  Represents a grid structure composed of chunks, allowing retrieval and modification of navigation nodes within defined grid indices. The grid is initialized with specified dimensions and chunk sizes, and supports operations to get and set navigation nodes at specific indices. Additional functionality includes checking for out-of-bounds indices and calculating neighboring nodes optionally including diagonals.
- class **NavNode**

  *The **NavNode** (p. 17) is the atomic structure used to hold the data and functions for each point in the flow field. I used it like this: I overlaid a grid of the same size as the tilemap grid and added a **NavNode** (p. 17) to each entry in my grid. But you could scale this factor up or down. You could use more or less tiles for the **NavNode** (p. 17).*

4.2 VectorPath Namespace Reference

Classes

- class **ExtraGizmos**
- class **NavigationFlowField**

  Manages navigation flow field generation and visualization for AI pathfinding on a grid-based map. The flow field is computed based on walkable tilemaps and target positions within the defined grid dimensions. Features include generating and updating navigation costs, determining walkable neighbors, and calculating movement directions. Various debugging options are available to visualize grid boundaries, target positions, traversal costs, chunk updates, and flow directions.
- class **NavigationFlowFieldEditor**
- class **NavigationManager**

  Singleton manager responsible for coordinating navigation flow field operations and target management.
- class **Navigator**

  *Helper class for obtaining movement directions from the **NavigationFlowField** (p. 12) managed by the **NavigationManager** (p. 14).*

4.3 VectorPathExamples Namespace Reference

Classes

- class **RatMovement**

  Controls the movement and rotation of a rat in 2D space.
- class **RatSpawn**

  Spawns rats at two designated spawn points at regular intervals.
- class **TargetSpawn**

  Spawns target asset at designated positions and updates navigation towards them.

Chapter 5

Class Documentation

5.1 FlowField.Chunk Class Reference

A chunk is used to improve performance. It holds a number of NavNodes and adds some functionality over them. It can have a minimum size of 1 and a maximum size of the flowfield itself. Both extreme cases would not help performance, so you need to find a chunk size that is suitable for your application.

Public Member Functions

- **Chunk** (Vector2Int size, Vector2Int gridPosition)

  *Constructor for a **Chunk** (p. 9). A **Chunk** (p. 9) consists out of a size and a 2D-Array of NavNodes. Size will be set by the parameter, chunkWasModified to false and the Array will be initialized and filled with dummy NavNodes.*
- **NavNode GetNode** (Vector2Int index)

  *Returns the **NavNode** (p. 17) in the 2D-Array at the given index.*
- void **SetNode** (Vector2Int index, **NavNode** navNode)

  Sets the navNode in the 2D-Array at the given index. It will also mark the chunk as having been modified.
- Vector2Int **GetSize** ()

  Returns the size of the chunk.
- void **ResetModified** ()

  Resets chunkWasModified.

Properties

- bool **chunkWasModified** `[get]`

5.1.1 Detailed Description

A chunk is used to improve performance. It holds a number of NavNodes and adds some functionality over them. It can have a minimum size of 1 and a maximum size of the flowfield itself. Both extreme cases would not help performance, so you need to find a chunk size that is suitable for your application.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Chunk()

```
FlowField.Chunk.Chunk (
            Vector2Int size,
            Vector2Int gridPosition)  [inline]
```

Constructor for a **Chunk** (p. 9). A **Chunk** (p. 9) consists out of a size and a 2D-Array of NavNodes. Size will be set by the parameter, chunkWasModified to false and the Array will be initialized and filled with dummy NavNodes.

Parameters

size	Size of the **Chunk** (p. 9). It has to be greater then 0.
gridPosition	Position of the **Chunk** (p. 9). The position of the chunk is also the position of the bottom left **NavNode** (p. 17) in the chunk.

5.1.3 Member Function Documentation

5.1.3.1 GetNode()

```
NavNode FlowField.Chunk.GetNode (
            Vector2Int index)  [inline]
```

Returns the **NavNode** (p. 17) in the 2D-Array at the given index.

Parameters

index	Index in the array (0 is in the bottom left corner, 1st dimension in the x direction, 2nd dimension in the y direction)

Returns

NavNode (p. 17)

5.1.3.2 GetSize()

```
Vector2Int FlowField.Chunk.GetSize ()  [inline]
```

Returns the size of the chunk.

Returns

Size of the chunk.

5.1.3.3 SetNode()

```
void FlowField.Chunk.SetNode (
            Vector2Int index,
             NavNode navNode)  [inline]
```

Sets the navNode in the 2D-Array at the given index. It will also mark the chunk as having been modified.

Parameters

index	Index in the array (0 is in the bottom left corner, 1st dimension in the x direction, 2nd dimension in the y direction)
navNode	**NavNode** (p. 17) that you want to put in the array.

The documentation for this class was generated from the following file:

- Navigation/FlowField/Chunk.cs

5.2 FlowField.Grid Class Reference

Represents a grid structure composed of chunks, allowing retrieval and modification of navigation nodes within defined grid indices. The grid is initialized with specified dimensions and chunk sizes, and supports operations to get and set navigation nodes at specific indices. Additional functionality includes checking for out-of-bounds indices and calculating neighboring nodes optionally including diagonals.

Public Member Functions

- **Grid** (Vector2Int size, Vector2Int chunkSize)

  Initializes a grid structure with specified dimensions and chunk sizes.
- **NavNode GetNavNode** (Vector2Int index)

  Retrieves the navigation node at the specified grid index.
- void **SetNavNode** (**NavNode** navNode, Vector2Int index)

  Sets the navigation node at the specified grid index.

5.2.1 Detailed Description

Represents a grid structure composed of chunks, allowing retrieval and modification of navigation nodes within defined grid indices. The grid is initialized with specified dimensions and chunk sizes, and supports operations to get and set navigation nodes at specific indices. Additional functionality includes checking for out-of-bounds indices and calculating neighboring nodes optionally including diagonals.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 Grid()

```
FlowField.Grid.Grid (
            Vector2Int size,
            Vector2Int chunkSize)  [inline]
```

Initializes a grid structure with specified dimensions and chunk sizes.

Parameters

size	The overall size of the grid in grid units (tiles).
chunkSize	The size of each chunk in grid units.

5.2.3 Member Function Documentation

5.2.3.1 GetNavNode()

```
NavNode FlowField.Grid.GetNavNode (
            Vector2Int index)  [inline]
```

Retrieves the navigation node at the specified grid index.

Parameters

index	The grid index of the navigation node to retrieve.

Returns

The navigation node at the specified index.

5.2.3.2 SetNavNode()

```
void FlowField.Grid.SetNavNode (
            NavNode navNode,
          Vector2Int index)  [inline]
```

Sets the navigation node at the specified grid index.

Parameters

navNode	The navigation node to set.
index	The grid index where the navigation node should be set.

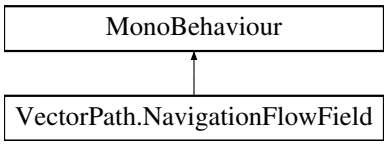
The documentation for this class was generated from the following file:

- Navigation/FlowField/Grid.cs

5.3 VectorPath.NavigationFlowField Class Reference

Manages navigation flow field generation and visualization for AI pathfinding on a grid-based map. The flow field is computed based on walkable tilemaps and target positions within the defined grid dimensions. Features include generating and updating navigation costs, determining walkable neighbors, and calculating movement directions. Various debugging options are available to visualize grid boundaries, target positions, traversal costs, chunk updates, and flow directions.

Inheritance diagram for VectorPath.NavigationFlowField:



Public Member Functions

- void **Instantiate** ()

  Initializes the Navigation Flow Field by validating and processing input parameters such as walkable tilemaps, grid size, chunk size, and target position. This method ensures that necessary components are set up, initializes data structures for navigation nodes, and checks the validity of the target position within the grid.
- void **CalculateNavMap** ()

  Calculates the navigation map (NavMap) including generating the cost map and calculating flow directions.
- Vector2 **GetDirection** (Vector2Int position)

  Retrieves the normalized direction vector at the specified grid position from the Navigation Flow Field.
- Vector2Int **GetPositionInNavMap** (Vector2 position)

  Converts a world position into a grid position in the navigation map (NavMap).
- bool **PositionIsInNavMap** (Vector2Int position)

  Checks if a grid position is within the bounds of the navigation map (NavMap).

Public Attributes

- Vector3 **targetPosition**
- **Chunk chunk**
- **NavNode**[,] **NavMap**
- bool **PreComputed** = false

5.3.1 Detailed Description

Manages navigation flow field generation and visualization for AI pathfinding on a grid-based map. The flow field is computed based on walkable tilemaps and target positions within the defined grid dimensions. Features include generating and updating navigation costs, determining walkable neighbors, and calculating movement directions. Various debugging options are available to visualize grid boundaries, target positions, traversal costs, chunk updates, and flow directions.

5.3.2 Member Function Documentation

5.3.2.1 GetDirection()

```
Vector2 VectorPath.NavigationFlowField.GetDirection (
            Vector2Int position)  [inline]
```

Retrieves the normalized direction vector at the specified grid position from the Navigation Flow Field.

Parameters

position	The grid position for which to retrieve the direction vector.

Returns

The normalized direction vector at the specified grid position.

This method ensures that the provided position is within the valid range of the Navigation Flow Field (NavMap). If the position is out of bounds, it logs an error and returns a zero vector.

5.3.2.2 GetPositionInNavMap()

```
Vector2Int VectorPath.NavigationFlowField.GetPositionInNavMap (
            Vector2 position)  [inline]
```

Converts a world position into a grid position in the navigation map (NavMap).

Parameters

position	The world position to convert.

Returns

The grid position in the NavMap corresponding to the world position.

5.3.2.3 PositionIsInNavMap()

```
bool VectorPath.NavigationFlowField.PositionIsInNavMap (
            Vector2Int position)  [inline]
```

Checks if a grid position is within the bounds of the navigation map (NavMap).

Parameters

position	The grid position to check.

Returns

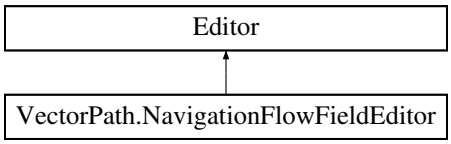
True if the position is within the bounds of the NavMap; otherwise, false.

The documentation for this class was generated from the following file:

- Navigation/NavigationFlowField.cs

5.4 VectorPath.NavigationFlowFieldEditor Class Reference

Inheritance diagram for VectorPath.NavigationFlowFieldEditor:



Public Member Functions

- override void **OnInspectorGUI** ()

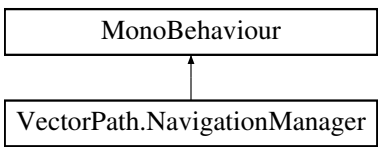
The documentation for this class was generated from the following file:

- Editor/NavigationFlowFieldEditor.cs

5.5 VectorPath.NavigationManager Class Reference

Singleton manager responsible for coordinating navigation flow field operations and target management.

Inheritance diagram for VectorPath.NavigationManager:



Public Member Functions

- void **ForceSetTarget** (Transform target)

  Sets the navigation target and updates the target position in the navigation flow field.
- void **ForceUpdateNavMap** ()

  Forces an update of the navigation map (NavMap) using the navigation flow field.

Public Attributes

- **NavigationFlowField navigationFlowField**

  The navigation flow field used for pathfinding calculations.
- Transform **Target**

  The current target for navigation purposes.

Properties

- static **NavigationManager Instance** `[get]`

  *Singleton instance of the **NavigationManager** (p. 14).*

5.5.1 Detailed Description

Singleton manager responsible for coordinating navigation flow field operations and target management.

5.5.2 Member Function Documentation

5.5.2.1 ForceSetTarget()

```
void VectorPath.NavigationManager.ForceSetTarget (
            Transform target)  [inline]
```

Sets the navigation target and updates the target position in the navigation flow field.

Parameters

target	The new target transform.

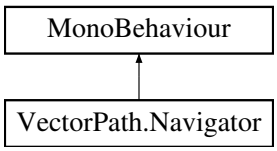
The documentation for this class was generated from the following file:

- Navigation/NavigationManager.cs

5.6 VectorPath.Navigator Class Reference

Helper class for obtaining movement directions from the **NavigationFlowField** (p. 12) managed by the **NavigationManager** (p. 14).

Inheritance diagram for VectorPath.Navigator:



Public Member Functions

- Vector2 **GetMoveDirection** (Vector2 position)

  Retrieves the movement direction based on the given position in the navigation grid.

5.6.1 Detailed Description

Helper class for obtaining movement directions from the **NavigationFlowField** (p. 12) managed by the **NavigationManager** (p. 14).

5.6.2 Member Function Documentation

5.6.2.1 GetMoveDirection()

```
Vector2 VectorPath.Navigator.GetMoveDirection (
            Vector2 position)  [inline]
```

Retrieves the movement direction based on the given position in the navigation grid.

Parameters

position	The position for which to retrieve the movement direction.

Returns

The movement direction vector.

The documentation for this class was generated from the following file:

- Navigation/Navigator.cs

5.7 FlowField.NavNode Class Reference

The **NavNode** (p. 17) is the atomic structure used to hold the data and functions for each point in the flow field. I used it like this: I overlaid a grid of the same size as the tilemap grid and added a **NavNode** (p. 17) to each entry in my grid. But you could scale this factor up or down. You could use more or less tiles for the **NavNode** (p. 17).

Public Member Functions

- **NavNode** (Vector2Int gridPosition, int costToTraverse=1)

  *Constructor for **NavNode** (p. 17). The costs will be set to Int32-MaxValue, which represents an obstacle. The direction is set to a zero-vector, since it should be changed once calculated.*
- bool **isObstacle** ()

  Getter Method to check if NavNode2D is an obstacle.
- int **GetCost** ()

  *Return the cost of the **NavNode** (p. 17).*
- void **SetCost** (int cost, bool useCostToTraverse=true)

  *Sets the cost of the **NavNode** (p. 17).*

Public Attributes

- int **CostToTraverse**

  Cost to traverse this Node. It could be used to make it less likely for agents to traverse ground types like mud. You would need to implement the behaviour of setting the travers cost.
- Vector2 **Direction**

  The direction has to be calculated by the kernal before you can use it. After calculation it will have the direction a agent has to take to get to the target location.
- readonly Vector2Int **GridPosition**

  *This is the position of the **NavNode** (p. 17) in the overall **FlowField** (p. 7).*

5.7.1 Detailed Description

The **NavNode** (p. 17) is the atomic structure used to hold the data and functions for each point in the flow field. I used it like this: I overlaid a grid of the same size as the tilemap grid and added a **NavNode** (p. 17) to each entry in my grid. But you could scale this factor up or down. You could use more or less tiles for the **NavNode** (p. 17).

5.7.2 Constructor & Destructor Documentation

5.7.2.1 NavNode()

```
FlowField.NavNode.NavNode (
            Vector2Int gridPosition,
            int costToTraverse = 1)  [inline]
```

Constructor for **NavNode** (p. 17). The costs will be set to Int32-MaxValue, which represents an obstacle. The direction is set to a zero-vector, since it should be changed once calculated.

Parameters

gridPosition	Position of the **NavNode** (p. 17) in the **FlowField** (p. 7) **Grid** (p. 11) (x, y Coordinats).
costToTraverse	Costs to traverse the **NavNode** (p. 17) (default set to 1). It has to be a positiv value. (0 is not positiv)

5.7.3 Member Function Documentation

5.7.3.1 GetCost()

```
int FlowField.NavNode.GetCost ()  [inline]
```

Return the cost of the **NavNode** (p. 17).

Returns

Cost of the **NavNode** (p. 17)

5.7.3.2 isObstacle()

```
bool FlowField.NavNode.isObstacle ()  [inline]
```

Getter Method to check if NavNode2D is an obstacle.

Returns

Returns true if NavNode2D is an obstacle.

5.7.3.3 SetCost()

```
void FlowField.NavNode.SetCost (
            int cost,
            bool useCostToTraverse = true)  [inline]
```

Sets the cost of the **NavNode** (p. 17).

Parameters

cost	Cost to set to
useCostToTraverse	

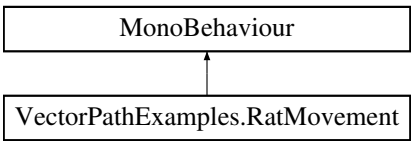
The documentation for this class was generated from the following file:

- Navigation/FlowField/NavNode.cs

5.8 VectorPathExamples.RatMovement Class Reference

Controls the movement and rotation of a rat in 2D space.

Inheritance diagram for VectorPathExamples.RatMovement:



Public Member Functions

- void **Move** (Vector2 newDirection)
  Moves the rat towards the specified direction.

5.8.1 Detailed Description

Controls the movement and rotation of a rat in 2D space.

5.8.2 Member Function Documentation

5.8.2.1 Move()

```
void VectorPathExamples.RatMovement.Move (
            Vector2 newDirection)  [inline]
```

Moves the rat towards the specified direction.

Parameters

newDirection	The direction vector in which the rat should move.

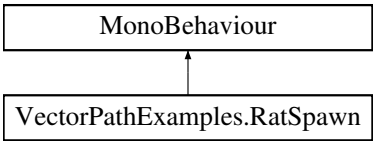
The documentation for this class was generated from the following file:

- ExampleScenes/Scripts/RatMovement.cs

5.9 VectorPathExamples.RatSpawn Class Reference

Spawns rats at two designated spawn points at regular intervals.

Inheritance diagram for VectorPathExamples.RatSpawn:



5.9.1 Detailed Description

Spawns rats at two designated spawn points at regular intervals.

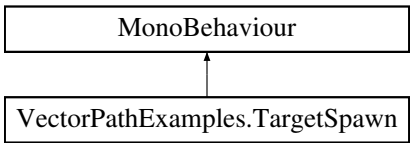
The documentation for this class was generated from the following file:

- ExampleScenes/Scripts/RatSpawn.cs

5.10 VectorPathExamples.TargetSpawn Class Reference

Spawns target asset at designated positions and updates navigation towards them.

Inheritance diagram for VectorPathExamples.TargetSpawn:



5.10.1 Detailed Description

Spawns target asset at designated positions and updates navigation towards them.

The documentation for this class was generated from the following file:

- ExampleScenes/Scripts/TargetSpawn.cs

Index