Documentation for AStarPathfinding: 2
Introduction 2
Methods 3
GeneratePath 3
With walkableMap 3
With costMap 4
GeneratePathSync 5
With walkableMap 5
With costMap 6

Documentation for AStarPathfinding:

Introduction

The **AStarPathfinding class provides functionality for generating paths** using A* algorithm for **two types of maps: walkable maps and cost maps**.

The class provides **two public methods**, **GeneratePath** and **GeneratePathSync**, **for generating paths asynchronously and synchronously**, respectively.

If you find this AStarPathfinding asset useful in your Unity projects, I would **greatly appreciate** it if you could take a moment to **leave a review**.

AStar 2D Grid Pathfinding | Behavior AI | Unity Asset Store

Your feedback is invaluable to me and allows me to make more informed decisions about how to improve this asset.

Methods

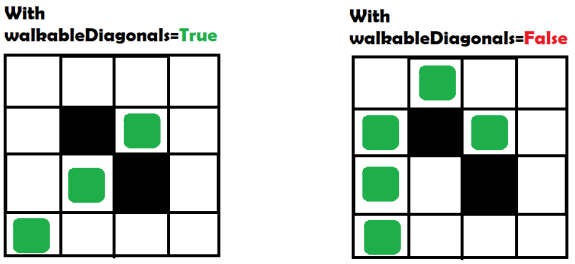
GeneratePath

With walkableMap

```
public static async Task<(int, int)[]> GeneratePath(int startX,
int startY, int goalX, int goalY, bool[,] walkableMap,
bool manhattanHeuristic = true, bool walkableDiagonals = false)
```

This method asynchronously generates a path from the start coordinates (startX, startY) to the goal coordinates (goalX, goalY) on the boolean walkableMap 2D array.

- `**startX**` (int) - the x coordinate of the starting point
- `**startY**` (int) - the y coordinate of the starting point
- `**goalX**` (int) - the x coordinate of the goal point
- `**goalY**` (int) - the y coordinate of the goal point
- `**walkableMap**` (bool[,]) - a 2D array indicating whether a tile is traversable or not. The array is ordered as `[rows, columns]` i.e `[y, x]`
- `**manhattanHeuristic**` (bool) - if `true`, the Manhattan distance heuristic is used, otherwise the Euclidean distance heuristic is used. Default value is `true`
- `**walkableDiagonals**` (bool) - if true diagonal movement is allowed even if it is not reachable via a horizontal and a vertical move



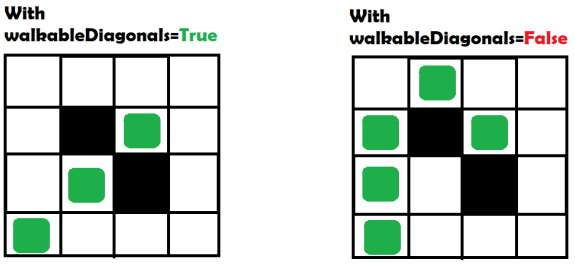
Asynchronously returns a tuple of `(int, int)[]` representing path coordinates traveling from start to goal. The coordinates are ordered as `(x, y)`. If no path is found, an empty array is returned.

With costMap

```
public static async Task<(int, int)[]> GeneratePath(int startX,
int startY, int goalX, int goalY, float[,] costMap, bool
manhattanHeuristic = true, bool walkableDiagonals = false)
```

This method asynchronously generates a path from the start coordinates (startX, startY) to the goal coordinates (goalX, goalY) on the boolean walkableMap 2D array.

- `**startX**` (int) - the x coordinate of the starting point
- `**startY**` (int) - the y coordinate of the starting point
- `**goalX**` (int) - the x coordinate of the goal point
- `**goalY**` (int) - the y coordinate of the goal point
- `**costMap**` (float[,]) - a 2D array indicating the cost of traveling through tiles (-1f if the tile is not walkable). The array is ordered as `[rows, columns]` i.e `[y, x]`
- `**manhattanHeuristic**` (bool) - if `true`, the Manhattan distance heuristic is used, otherwise the Euclidean distance heuristic is used. Default value is `true`
- `**walkableDiagonals**` (bool) - if true diagonal movement is allowed even if it is not reachable via a horizontal and a vertical move



Synchronously returns a tuple of `(int, int)[]` representing path coordinates traveling from start to goal. The coordinates are ordered as `(x, y)`. If no path is found, an empty array is returned.

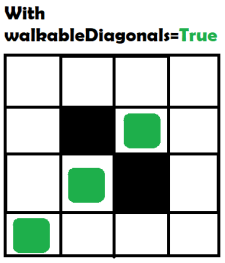
GeneratePathSync

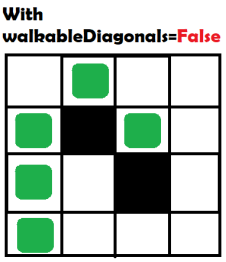
With walkableMap

public static (int, int)[] GeneratePathSync(int startX, int startY, int goalX, int goalY, bool[,] walkableMap, bool manhattanHeuristic = true, bool walkableDiagonals = false)

This method asynchronously generates a path from the start coordinates (startX, startY) to the goal coordinates (goalX, goalY) on the boolean walkableMap 2D array.

- `**startX**` (int) - the x coordinate of the starting point
- `**startY**` (int) - the y coordinate of the starting point
- `**goalX**` (int) - the x coordinate of the goal point
- `**goalY**` (int) - the y coordinate of the goal point
- `**walkableMap**` (bool[,]) - a 2D array indicating whether a tile is traversable or not. The array is ordered as `[rows, columns]` i.e `[y, x]`
- `**manhattanHeuristic**` (bool) - if `true`, the Manhattan distance heuristic is used, otherwise the Euclidean distance heuristic is used. Default value is `true`
- `**walkableDiagonals**` (bool) - if true diagonal movement is allowed even if it is not reachable via a horizontal and a vertical move





Synchronously returns a tuple of `(int, int)[]` representing path coordinates traveling from start to goal. The coordinates are ordered as `(x, y)`. If no path is found, an empty array is returned.

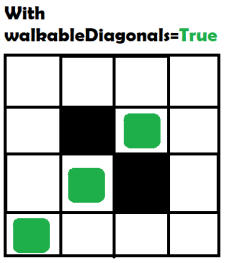
Warning: This method runs synchronously on Unity's thread and may cause momentary freezing if a hard path is calculated. If this occurs, use the asynchronous variant `GeneratePath`.

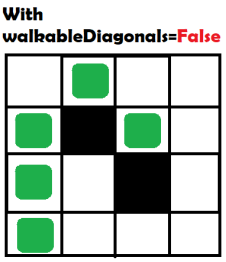
With costMap

`public static (int, int)[] GeneratePathSync(int startX, int startY, int goalX, int goalY, float[,] costMap, bool manhattanHeuristic = true, bool walkableDiagonals = false)`

This method asynchronously generates a path from the start coordinates (startX, startY) to the goal coordinates (goalX, goalY) on the boolean walkableMap 2D array.

- `**startX**` (int) - the x coordinate of the starting point
- `**startY**` (int) - the y coordinate of the starting point
- `**goalX**` (int) - the x coordinate of the goal point
- `**goalY**` (int) - the y coordinate of the goal point
- `**costMap**` (float[,]) - a 2D array indicating the cost of traveling through tiles (-1f if the tile is not walkable). The array is ordered as `[rows, columns]` i.e `[y, x]`
- `**manhattanHeuristic**` (bool) - if `true`, the Manhattan distance heuristic is used, otherwise the Euclidean distance heuristic is used. Default value is `true`
- `**walkableDiagonals**` (bool) - if true diagonal movement is allowed even if it is not reachable via a horizontal and a vertical move





Synchronously returns a tuple of `(int, int)[]` representing path coordinates traveling from start to goal. The coordinates are ordered as `(x, y)`. If no path is found, an empty array is returned.

Warning: This method runs synchronously on Unity's thread and may cause momentary freezing if a hard path is calculated. If this occurs, use the asynchronous variant `GeneratePath`.