https://www.youtube.com/watch?v=lwLhG\_bQa\_w&list=PLX-uZVK\_0K\_6GjJ\_tgg1YXmO8lor7sRM7&index=2

A\* finds shortest path from A to B Uses F(x) = G(x) + H(x)

**G(x) :** how much does it cost to move from **START** to the current node throught the generated path

It Costs 10 to move Horizontal and vertical

It Costs 14 to move Diagonal **(NOTE: WILL NOT BE ABLE TO MOVE DIAGONALLY)**

**G** is calculated by adding 10 or 14 to the value of the **parent node**

|  |  |  |
| --- | --- | --- |
|  | G:10 |  |
| G:10 | **Start** | G:10 |
|  | G:10 |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Start ->** | G:10 -> | G:20 Finish |
|  |  |  |

**H(X) : H** is the estimated cost for moving from **any node to the goal** (called Heuristic value)

Unwalkable nodes are ignored

Diagonal moves are ignored

The cost of a move is 10

The cost can be calculated by taking the number of steps from start node to end node and multiplying by 10

**Steps x 10 = H**

|  |  |  |  |
| --- | --- | --- | --- |
| H:40 > | H:30 > | H:20 > | H:10 \/ |
| **Start /\ H:30 >** | H:20> | BLOCKED> | **Finish** |
| H:40 | H:30 | H:20 | H:10 |

**F(x) = G + H** this is the value we will use to find a path

|  |  |  |  |
| --- | --- | --- | --- |
| Start |  | Blocked | Finish |
|  | G:14  H:30  F:44 | G:24  H:20  F:44 | G:34  H:10F:44 |

Ex:

1st Add **start** to Open List

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | **Start** |  | Water | Goal |
|  |  |  |  |  |

2nd: Find all **Neighbors** unwalkable nodes are ignored

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | **Start** |  | Water | Goal |
|  |  |  |  |  |

3rd: the **Neighbors** are added to the open list

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | **Start** |  | Water | Goal |
|  |  |  |  |  |

4th: mark **Start** node as parent

Neighbors now point to start node cuz it’s the parent

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | **Start** |  | Water | Goal |
|  |  |  |  |  |

5th: move the **start** node from the **open list to the closed list**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | **Start** |  | Water | Goal |
|  |  |  |  |  |

6th: score the nodes as we learned earlier



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| G:14  H:50  F:64 | G:10:  H:40  F:50 | G:14  H:30  F:44 |  |  |
| G:10:  H:40  F:50 | **Start** | G:10:  H:20  F:30 | Water | Goal |
| G:14  H:50  F:64 | G:10:  H:40  F:50 | G:14  H:30  F:44 |  |  |

7th: the search continues by selecting the node with the **lowest F score** this is our **CURRENT** node



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| G:14  H:50  F:64 | G:10:  H:40  F:50 | G:14  H:30  F:44 |  |  |
| G:10:  H:40  F:50 | **Start** | G:10:  H:20  F:30 | Water | Goal |
| G:14  H:50  F:64 | G:10:  H:40  F:50 | G:14  H:30  F:44 |  |  |

8th: move the node from the **open list** to the **Closed List**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| G:14  H:50  F:64 | G:10:  H:40  F:50 | G:14  H:30  F:44 |  |  |
| G:10:  H:40  F:50 | **Start** | G:10:  H:20  F:30 | Water | Goal |
| G:14  H:50  F:64 | G:10:  H:40  F:50 | G:14  H:30  F:44 |  |  |

9.1 : Examine all Neighbors

**Ignore unwalkable nodes ignore nodes on the closed list**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| G:14  H:50  F:64 | G:10:  H:40  F:50 | G:14  H:30  F:44 |  |  |
| G:10:  H:40  F:50 | **Start** | G:10:  H:20  F:30 | Water | Goal |
| G:14  H:50  F:64 | G:10:  H:40  F:50 | G:14  H:30  F:44 |  |  |

9.2 Examine all Neighbors

If the **neighbor** is already on the **open list** check if the path through the **current** node is a better alternative

Is the G score lower if we use the **current** node as parent?

**If yes:** current node will be made new parent if No: don’t do anything

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| G:14  H:50  F:64 | G:10 **AG: 24**  H:40  F:50 | G:14 **AG:20**  H:30  F:44 |  |  |
| G:10:  H:40  F:50 | **Start** | G:10:  H:20  F:30 | Water | Goal |
| G:14  H:50  F:64 | G:10: **AG: 24**  H:40  F:50 | G:14 **AG: 20**  H:30  F:44 |  |  |

9.3: Examine all Neighbors

Add new Neighbors to the open list ignore nodes on closed list

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| G:14  H:50  F:64 | G:10 **AG: 24**  H:40  F:50 | G:14 **AG:20**  H:30  F:44 | New |  |
| G:10:  H:40  F:50 | **Start** | G:10:  H:20  F:30 | Water | Goal |
| G:14  H:50  F:64 | G:10: **AG: 24**  H:40  F:50 | G:14 **AG: 20**  H:30  F:44 | new |  |

9.4: Examine all Neighbors

Make the current node the parent of **the NEW neighbors**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| G:14  H:50  F:64 | G:10 **AG: 24**  H:40  F:50 | G:14 **AG:20**  H:30  F:44 |  |  |
| G:10:  H:40  F:50 | **Start** | G:10:  H:20  F:30 | Water | Goal |
| G:14  H:50  F:64 | G:10: **AG: 24**  H:40  F:50 | G:14 **AG: 20**  H:30  F:44 |  |  |

9.5: Examine all Neighbors

Calculate F(x) = G + h for the New Neighbors

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| G:14  H:50  F:64 | G:10 **AG: 24**  H:40  F:50 | G:14 **AG:20**  H:30  F:44 | G: 24  H: 20  F: 44 |  |
| G:10:  H:40  F:50 | **Start** | G:10:  H:20  F:30 | Water | Goal |
| G:14  H:50  F:64 | G:10: **AG: 24**  H:40  F:50 | G:14 **AG: 20**  H:30  F:44 | G:24  H:20  F:44 |  |

10: the search will continue by repeating steps 7 -9.5 until we hit the goal

11: Examine neigbors

If a neighbor is the **GOAL NODE** then stop the search

12: Reached Goal!

We can backtrack through the parents to make a complete PATH

**Psudo code**

List Open List Close

Int F Int G Int H

Tile clickedTile Tile rangeTile

Vector3int UnitStartPOS Vector3int ClickedTilePOS

Bool unitSelected = false Bool isInRange = false

//first click unit to select it

If(unitSelected == false)

{

//get units current location

//set current location to UnitStartPOS

// unitSelected = true

}

//second click destination tile

//if (unitSellected == true)

{

//check if clicked tile is in range

If(clickedTile == rangeTile)

{

//set clicked tile as goal node

//???????????

//run A\* algorithm (start, goal)

}

}

**A\* Algorithm(start, goal)**

{

//add start to open list

//find all horizontal/vertical neighbors

//FindNeighborsStartNode(start node)

//score nodes

//find G

//find H

//F = G + H

//Find node with lowest F score

//select node with lowest F score

//make current node

//move current node from open list to closed list

//examine all neighbors of current node

//FindNeighborsCurrentNode(Current node)

}

**FindNeighborsStartNode(Start node)**

{

//find all horizontal/vertical neighbors

//ignore unwalkable tiles

//ignore nodes on closed list

//add neighbors to open list

If(current node == start node)

{

//make start parent node

//move start node from open list to closed list

}

**FindNeighborsCurrentNode(Current node)**

{

//find all horizontal/vertical neighbors

//ignore unwalkable tiles

//ignore nodes on closed list

//if neigbor node is already in open list

//check if path through current node is a better alternative

}

}