**IMPLEMENTATION DETAILS:**

For the A\* tie breaker we decided to use the formula fCost = gCost + (1+p) \* hCost, where p<=MOVE\_STRAIGHT\_COST / (MOVE\_DIAGONAL\_COST \* 10).

**TEST CASE 1:**

In section *Anexes* you can find screenshots of the paths computed by each algorithm.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | TRADIOTIONAL A\* (NO TIE BREAKING) | TRADIOTIONAL A\* (WITH TIE BREAKING) | | NODE ARRAY A\* | A\* WITH GOAL BOUNDING |
| Fill | 16% | 13% | 16% | | 14% |
| Nodes visited | 847 | 708 | 848 | | 751 |
| Nodes processed | 752 | 608 | 757 | | 686 |
| Max open list size | 97 | 104 | 97 | | 77 |
| Total processing time | 1.146s | 0.838s | 1.097s | | 1.001s |
| Processing time per node | 0.0015s | 0.0013s | 0.0014s | | 0.0015s |
| GetBestAndRemove | 15 calls | 15 calls | 756 calls | | 667 calls |
| AddToOpen | 35 calls | 15 calls | 847 calls | | 749 calls |
| SearchInOpen | 109 calls | 102 calls | 0 calls | | 0 calls |
| RemoveFromOpen | 15 calls | 15 calls | 756 calls | | 667 calls |
| Replace | 3 calls | 1 call | 0 calls | | 0 calls |
| AddToClosed | 15 calls | 15 calls | 756 calls | | 667 calls |
| SearchInClosed | 109 calls | 102 calls | 0 calls | | 0 calls |
| RemoveFromClosed | 0 calls | 0 calls | 0 calls | | 0 calls |

**TEST CASE 2:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | TRADIOTIONAL A\* (NO TIE BREAKING) | TRADIOTIONAL A\* (WITH TIE BREAKING) | | NODE ARRAY A\* | A\* WITH GOAL BOUNDING |
| Fill | 12% | 10% | 12% | | 9% |
| Nodes visited | 640 | 523 | 641 | | 481 |
| Nodes processed | 553 | 436 | 568 | | 445 |
| Max open list size | 89 | 91 | 88 | | 52 |
| Total processing time | 0.813s | 0.607s | 0.830s | | 0.659s |
| Processing time per node | 0.0015s | 0.0014s | 0.0015s | | 0.0015s |
| GetBestAndRemove | 15 calls | 15 calls | 561 calls | | 441 calls |
| AddToOpen | 16 calls | 16 calls | 634 calls | | 480 calls |
| SearchInOpen | 111 calls | 110 calls | 0 calls | | 0 calls |
| RemoveFromOpen | 15 calls | 15 calls | 561 calls | | 441 calls |
| Replace | 8 calls | 10 calls | 0 calls | | 0 calls |
| AddToClosed | 15 calls | 15 calls | 561 calls | | 441 calls |
| SearchInClosed | 111 calls | 110 calls | 0 calls | | 0 calls |
| RemoveFromClosed | 0 calls | 0 calls | 0 calls | |  |

**TEST CASE 3:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | TRADIOTIONAL A\* (NO TIE BREAKING) | TRADIOTIONAL A\* (WITH TIE BREAKING) | | NODE ARRAY A\* | A\* WITH GOAL BOUNDING |
| Fill | 17% | 14% | 17% | | 13% |
| Nodes visited | 898 | 734 | 899 | | 717 |
| Nodes processed | 815 | 644 | 820 | | 674 |
| Max open list size | 88 | 92 | 89 | | 84 |
| Total processing time | 1.212s | 0.902s | 1.220s | | 0.982s |
| Processing time per node | 0.0015s | 0.0014s | 0.0015s | | 0.0015s |
| GetBestAndRemove | 15 calls | 15 calls | 819 calls | | 672 calls |
| AddToOpen | 13 calls | 14 calls | 895 calls | | 717 calls |
| SearchInOpen | 108 calls | 116 calls | 0 calls | | 0 calls |
| RemoveFromOpen | 15 calls | 15 calls | 819 calls | | 672 calls |
| Replace | 7 calls | 14 calls | 0 calls | | 0 calls |
| AddToClosed | 15 calls | 15 calls | 819 calls | | 672 calls |
| SearchInClosed | 108 calls | 116 calls | 0 calls | | 0 calls |
| RemoveFromClosed | 0 calls | 0 calls | 0 calls | | 0 calls |

**TEST CASE 4:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | TRADIOTIONAL A\* (NO TIE BREAKING) | TRADIOTIONAL A\* (WITH TIE BREAKING) | | NODE ARRAY A\* | A\* WITH GOAL BOUNDING |
| Fill | 26% | 20% | 26% | | 12% |
| Nodes visited | 1365 | 1080 | 1366 | | 622 |
| Nodes processed | 1226 | 905 | 1235 | | 560 |
| Max open list size | 156 | 177 | 156 | | 72 |
| Total processing time | 1.855s | 1.301s | 1.815s | | 0.825s |
| Processing time per node | 0.0015s | 0.0014s | 0.0015s | | 0.0015s |
| GetBestAndRemove | 15 calls | 15 calls | 1233 calls | | 559 calls |
| AddToOpen | 19 calls | 27 calls | 1366 calls | | 622 calls |
| SearchInOpen | 110 calls | 119 calls | 0 calls | | 0 calls |
| RemoveFromOpen | 15 calls | 15 calls | 1233 calls | | 559 calls |
| Replace | 2 calls | 9 calls | 0 calls | | 0 calls |
| AddToClosed | 15 calls | 15 calls | 1233 calls | | 559 calls |
| SearchInClosed | 110 calls | 119 calls | 0 calls | | 0 calls |
| RemoveFromClosed | 0 calls | 0 calls | 0 calls | | 0 calls |

**TEST CASE 5:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | TRADIOTIONAL A\* (NO TIE BREAKING) | TRADIOTIONAL A\* (WITH TIE BREAKING) | | NODE ARRAY A\* | A\* WITH GOAL BOUNDING |
| Fill | 25% | 17% | 25% | | 22% |
| Nodes visited | 1357 | 899 | 1358 | | 1191 |
| Nodes processed | 1187 | 768 | 1199 | | 1055 |
| Max open list size | 172 | 133 | 169 | | 144 |
| Total processing time | 1.818 | 1.169s | 1.739s | | 1.573s |
| Processing time per node | 0.0015s | 0.0015s | 0.0015s | | 0.0015s |
| GetBestAndRemove | 15 calls | 15 calls | 1195 calls | | 1040 calls |
| AddToOpen | 13 calls | 14 calls | 1357 calls | | 1186 calls |
| SearchInOpen | 103 calls | 97 calls | 0 calls | | 0 calls |
| RemoveFromOpen | 15 calls | 15 calls | 1195 calls | | 1040 calls |
| Replace | 4 calls | 2 calls | 0 calls | | 0 calls |
| AddToClosed | 15 calls | 15 calls | 1195 calls | | 1040 calls |
| SearchInClosed | 103 calls | 97 calls | 0 calls | | 0 calls |
| RemoveFromClosed | 0 calls | 0 calls | 0 calls | | 0 calls |

**TEST CASE 6:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | TRADIOTIONAL A\* (NO TIE BREAKING) | TRADIOTIONAL A\* (WITH TIE BREAKING) | | NODE ARRAY A\* | A\* WITH GOAL BOUNDING |
| Fill | 45% | 37% | 45% | | 38% |
| Nodes visited | 2382 | 1981 | 2383 | | 2057 |
| Nodes processed | 2231 | 1801 | 2242 | | 1943 |
| Max open list size | 156 | 182 | 157 | | 130 |
| Total processing time | 3.285 | 2.758 | 2.961s | | 2.548s |
| Processing time per node | 0.0015s | 0.0015s | 0.0013s | | 0.0013s |
| GetBestAndRemove | 15 calls | 15 calls | 2233 calls | | 1939 calls |
| AddToOpen | 14 calls | 15 calls | 2380 calls | | 2057 calls |
| SearchInOpen | 110 calls | 107 calls | 0 calls | | 0 calls |
| RemoveFromOpen | 15 calls | 15 calls | 2233 calls | | 1939 calls |
| Replace | 6 calls | 9 calls | 0 calls | | 0 calls |
| AddToClosed | 15 calls | 15 calls | 2233 calls | | 1939 calls |
| SearchInClosed | 110 calls | 107 calls | 0 calls | | 0 calls |
| RemoveFromClosed | 0 calls | 0 calls | 0 calls | | 0 calls |

**ALGORITHM ANALYSIS AND COMPARISON:**

**Fill Percentage:**

The algorithms with the smallest fill are A\* with tie breaker (3 out of 6 test cases with smallest fill) and A\* with Goal Bounding (also 3 out of 6 test cases with smallest fill). However, the A\* with tie breaker will always show a lower percentage of fill when compared to the A\* with no tie breaker. This doesn’t hold true when comparing the A\* with Goal Bounding and the Node Array A\*. In the latter case, if the bounding box is too big (for example, englobing the whole grid), the A\* with Goal Bounding algorithm will display the same result as the Node Array A\*.

**Function Calls per Frame:**

Both Node Array A\* and A\* with Goal Bounding have the biggest amount of function calls per frame. This is due to the use of the *NodeRecordArray* and the *NodePriorityHeap*, that allow for greater speed. The use of a single list with all the nodes prevents the algorithm from wasting time getting the nodes from the grid and moving them between the closed and open list (they are simply marked as *closed* or *open*).

**Total Processing Time:**

Once again, A\* with tie breaker and A\* with Goal Bounding are tied when it comes to total processing time.

The A\* with Goal Bounding takes a lot of time pre-processing the map, metric that wasn’t included in the test cases. It’s especially heavy (memory wise and time consuming) when the grid is at least as big as the giantGrid. Us, as the user, considered it was too long to wait for a performance that appeared to be virtually the same as the one displayed by Node Array A\*. After the initial pre-process, if the bounding box was small enough the performance was significantly better. With this in mind, we attempted to always choose the smallest bounding box that contained the goal node. However, due to how the bounding boxes are represented, some of them ended up containing the target but no direct path to it within the box. That idea was, of course, discarded.

The worst overall processing time belongs to the A\* with no tie breaker. The Euclidean Distance heuristic “forced” the algorithm to often search a whole area before moving on to another, which is not very adequate for maps with rooms.

Overall, the algorithms with the best average performance were the A\* with tie breaker and the A\* with Goal Bounding. Mixing the two together would certainly improve the metrics. However, the pre-processing of the map for bigger grids makes it expensive to run on less powerful computers. Since the differences between the Goal Bounding and the Node Array A\* were practically imperceptible to the naked eye, it would make sense to use that one in combination with a tie breaker.

**ANEXES:**

Uma imagem com texto, palavras cruzadas

Descrição gerada automaticamente

Figure 1 - Test case 1 for A\*, no tie breaker



Figure - Test case 1 for Node Array A\*

Uma imagem com texto, palavras cruzadas, quadro de resultados

Descrição gerada automaticamente

Figure - Test case 1 for Goal Bound



Figure - Test case 1 for A\* with tie breaker

Uma imagem com texto, palavras cruzadas, quadro de resultados

Descrição gerada automaticamente

Figure - Test case 2 for A\*, no tie breaker



Figure - Test case 2 for Node Array A\*

Uma imagem com texto, palavras cruzadas, quadro de resultados

Descrição gerada automaticamente

Figure - Test case 2 for Goal Bound

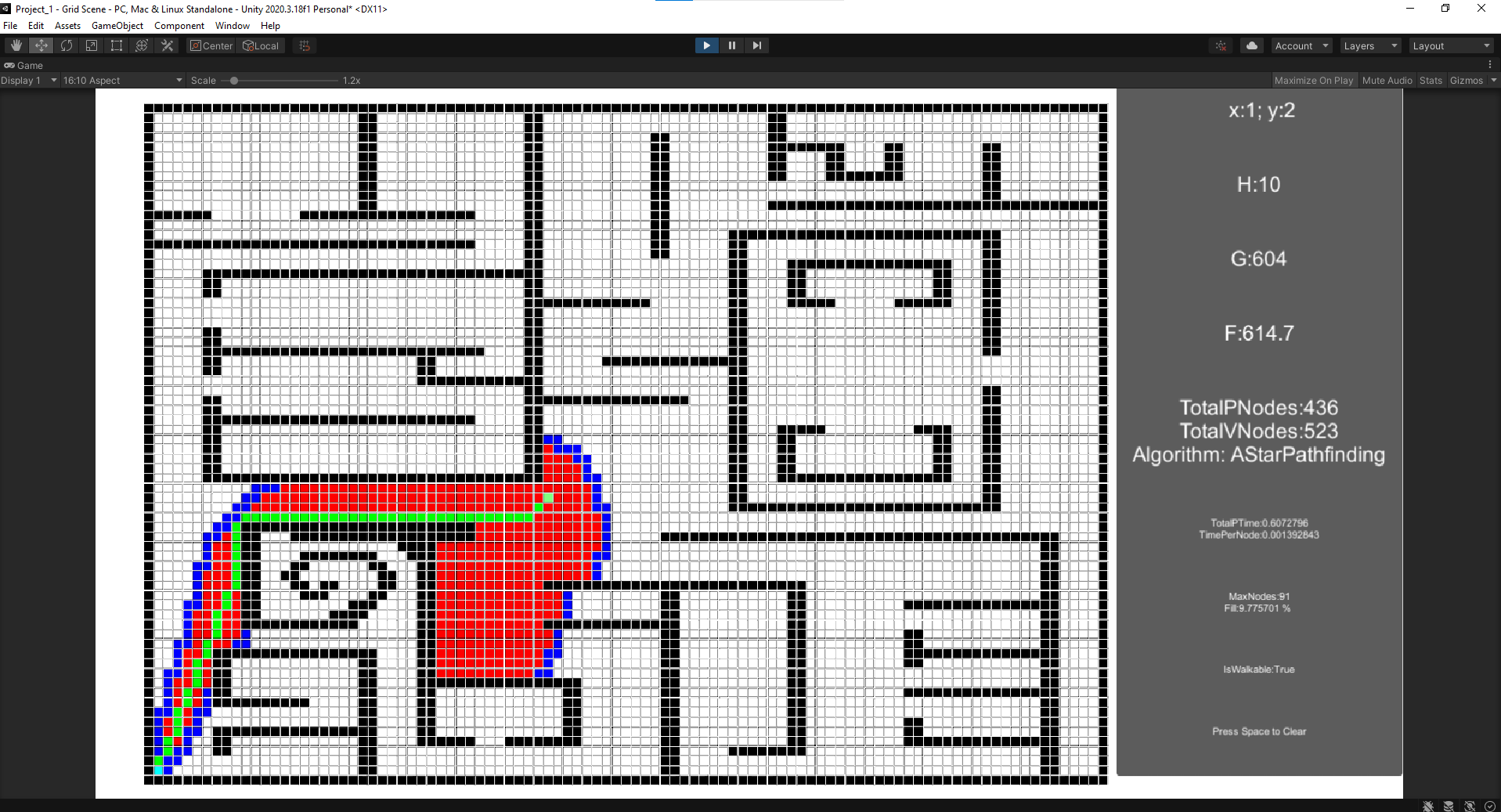


Figure - Test case 2 for A\* with tie breaker

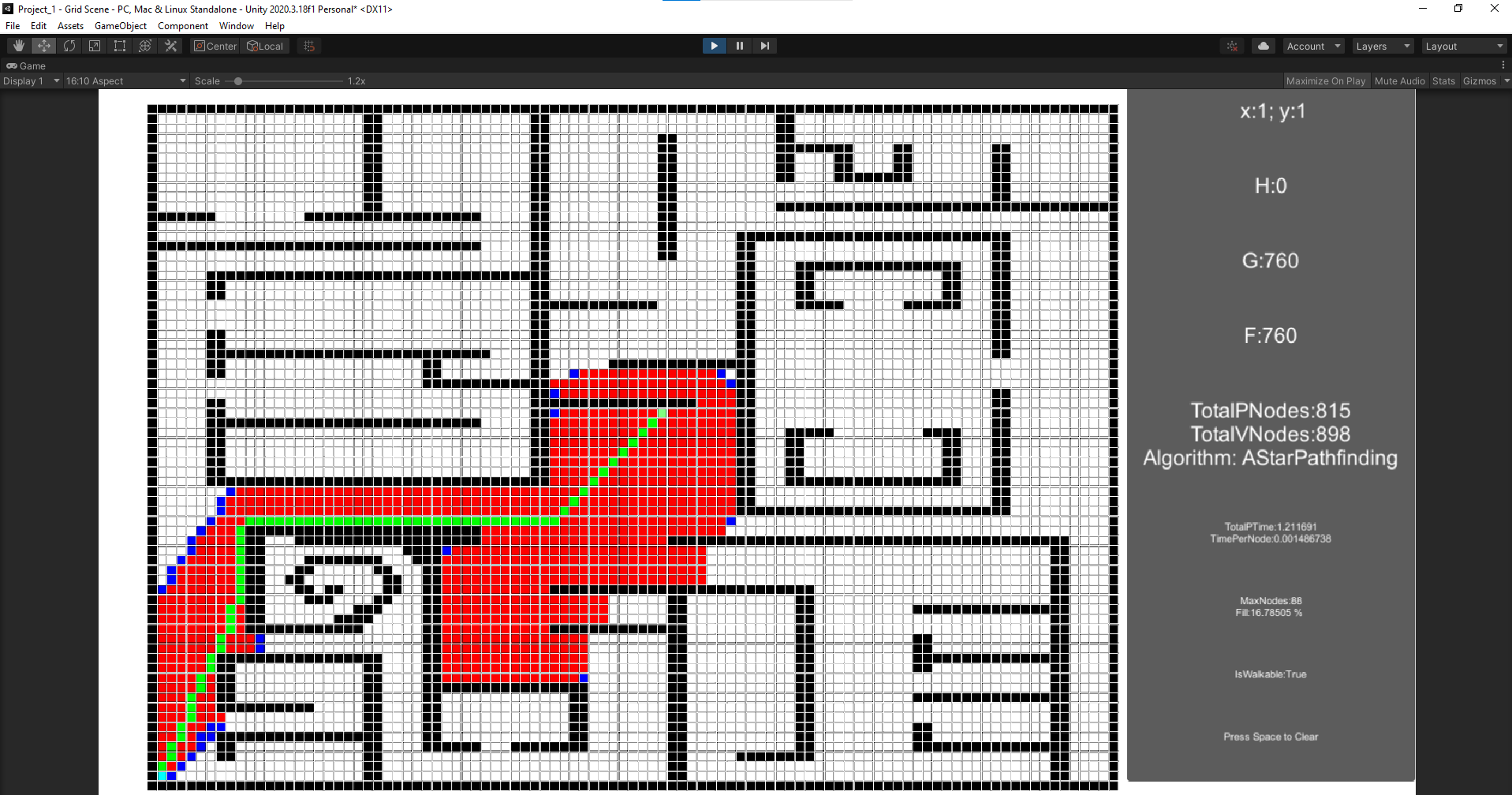


Figure - Test case 3 for A\*, no tie breaker

Uma imagem com texto, palavras cruzadas

Descrição gerada automaticamente

Figure - Test case 3 for Node Array A\*

Uma imagem com texto, palavras cruzadas

Descrição gerada automaticamente

Figure - Test case 3 for Goal Bound

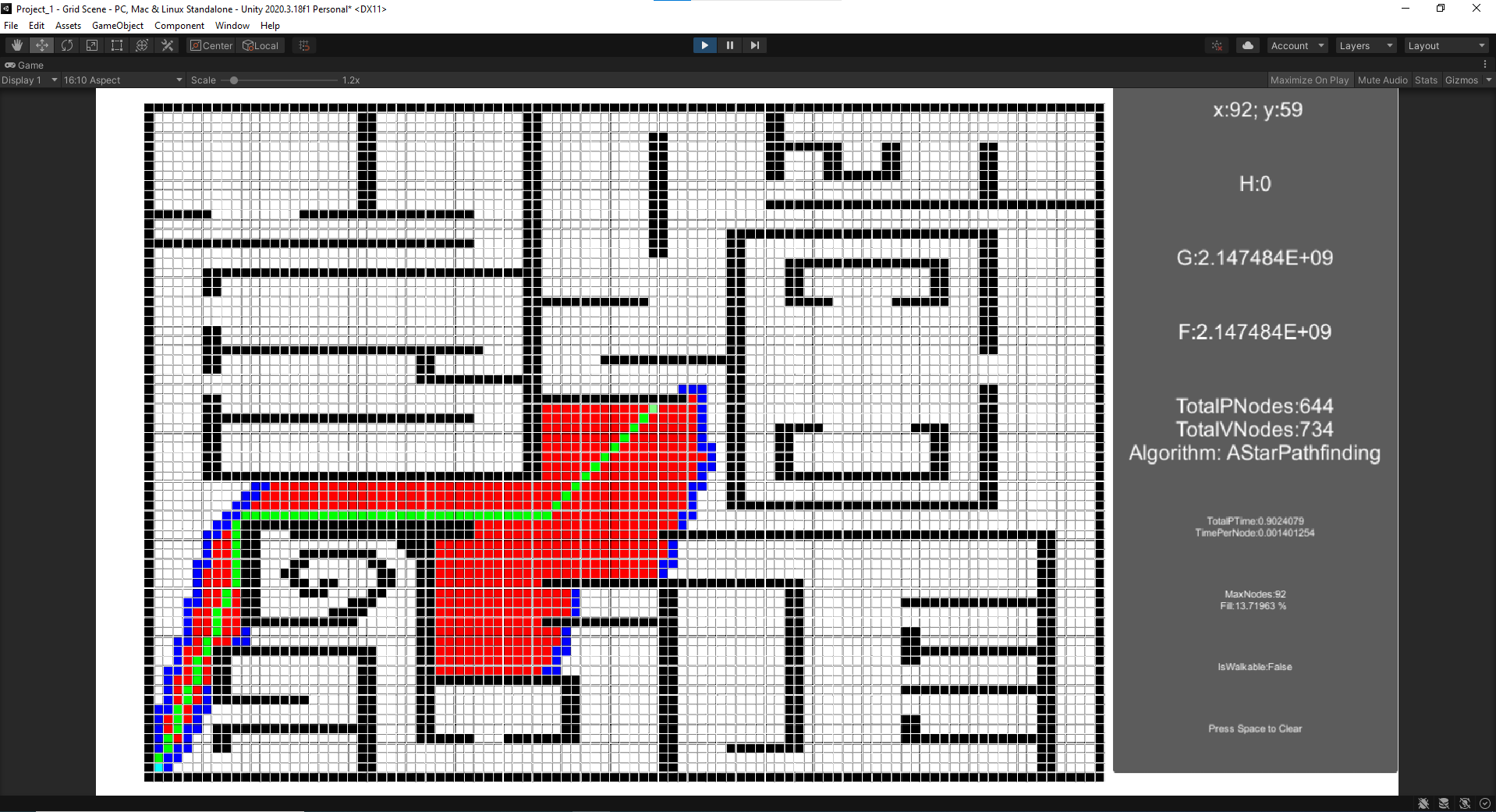


Figure - Test case 3 for A\* with tie breaker

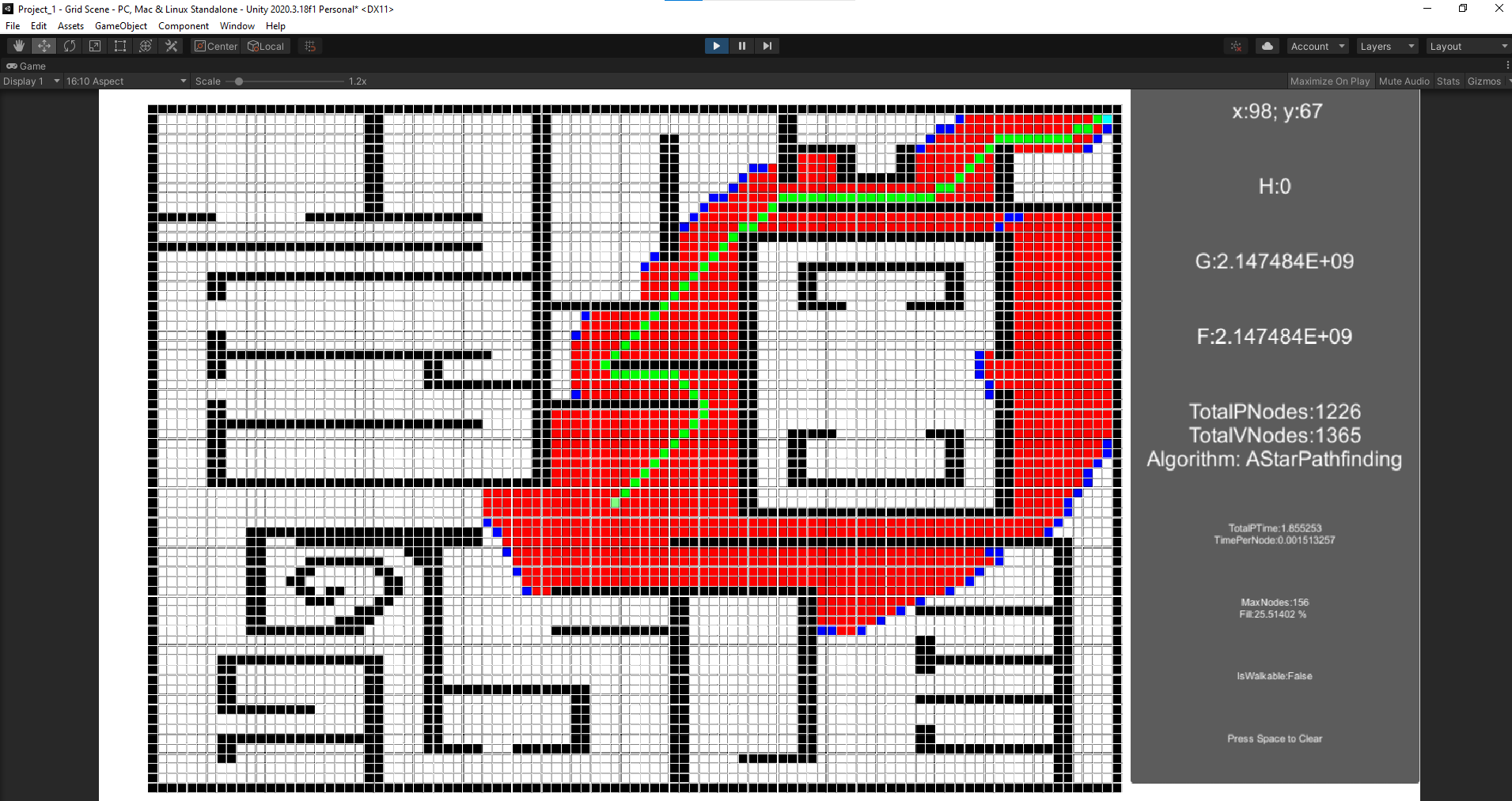


Figure - Test case 4 for A\*, no tie breaker

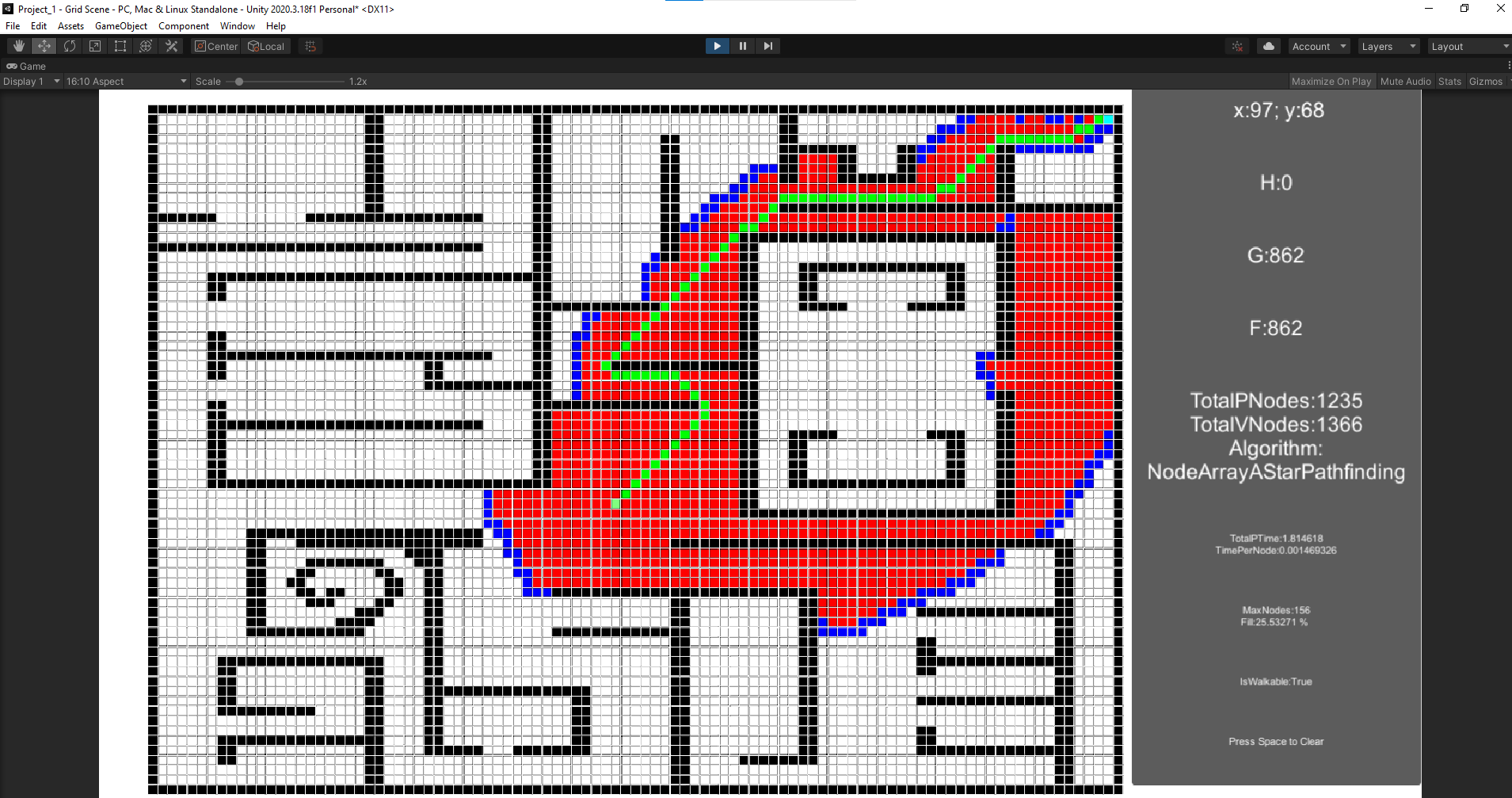


Figure - Test case 4 for Node Array A\*

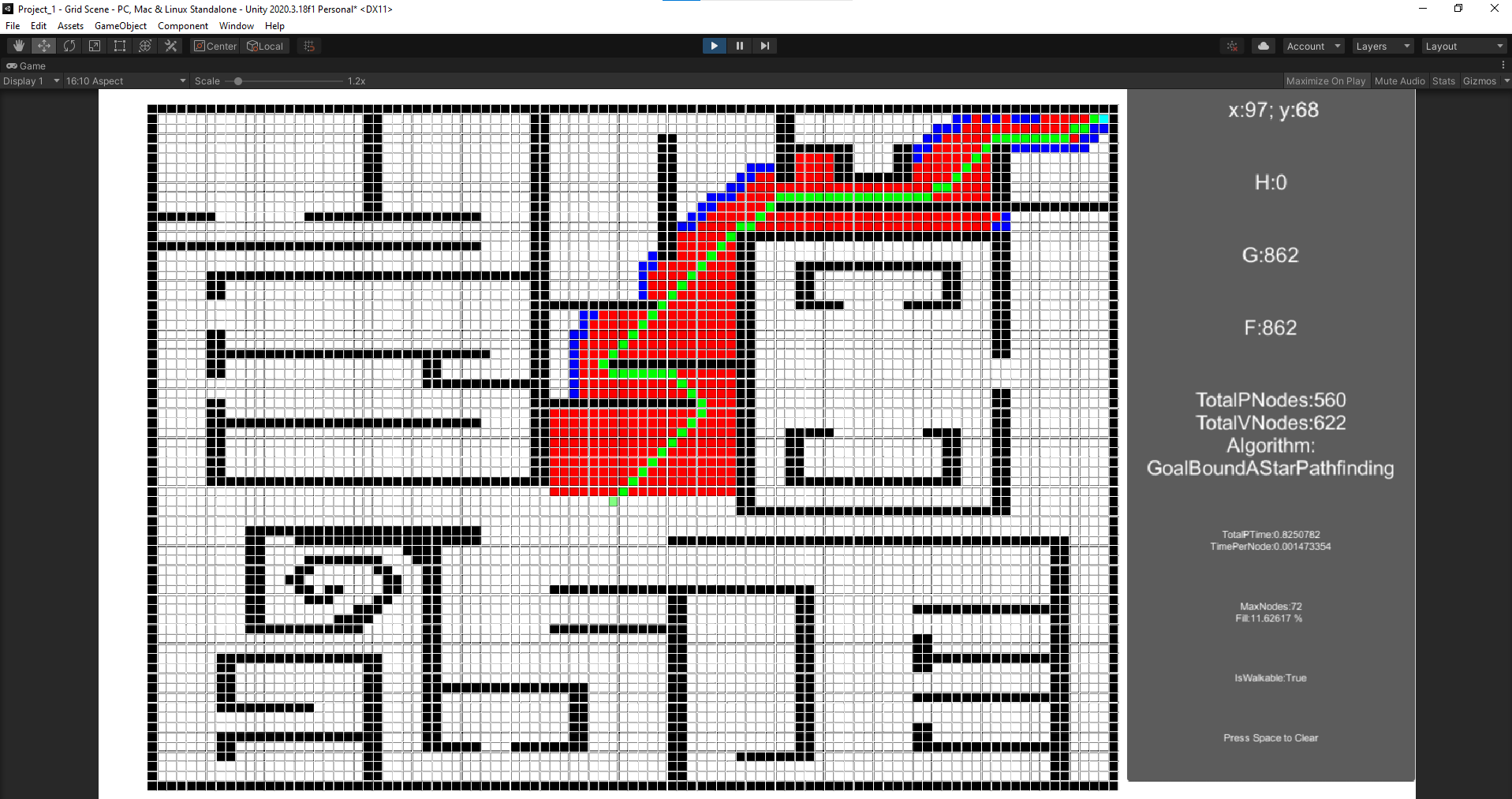


Figure - Test case 4 for Goal Bound

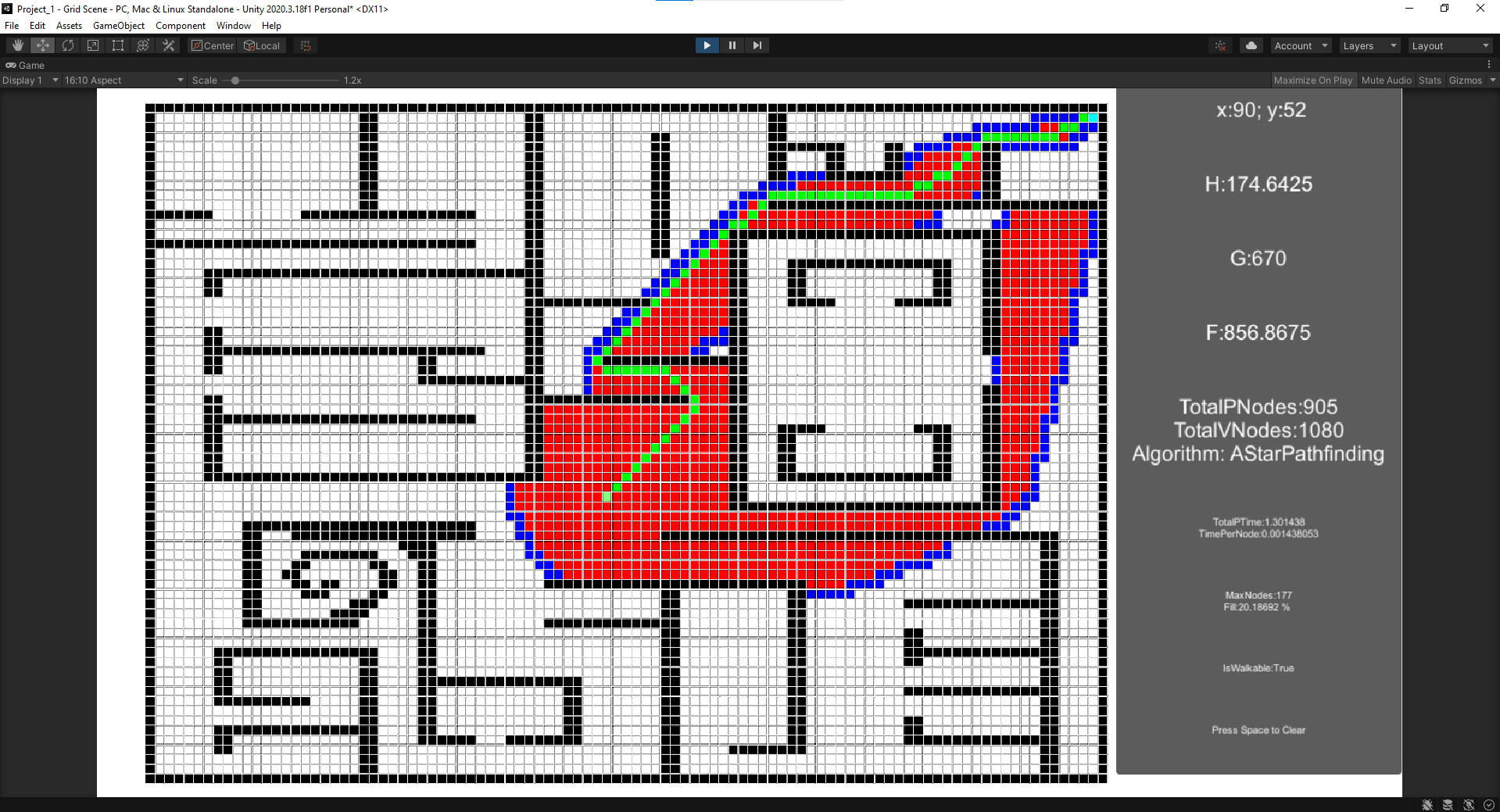


Figure - Test case 4 for A\* with tie breaker

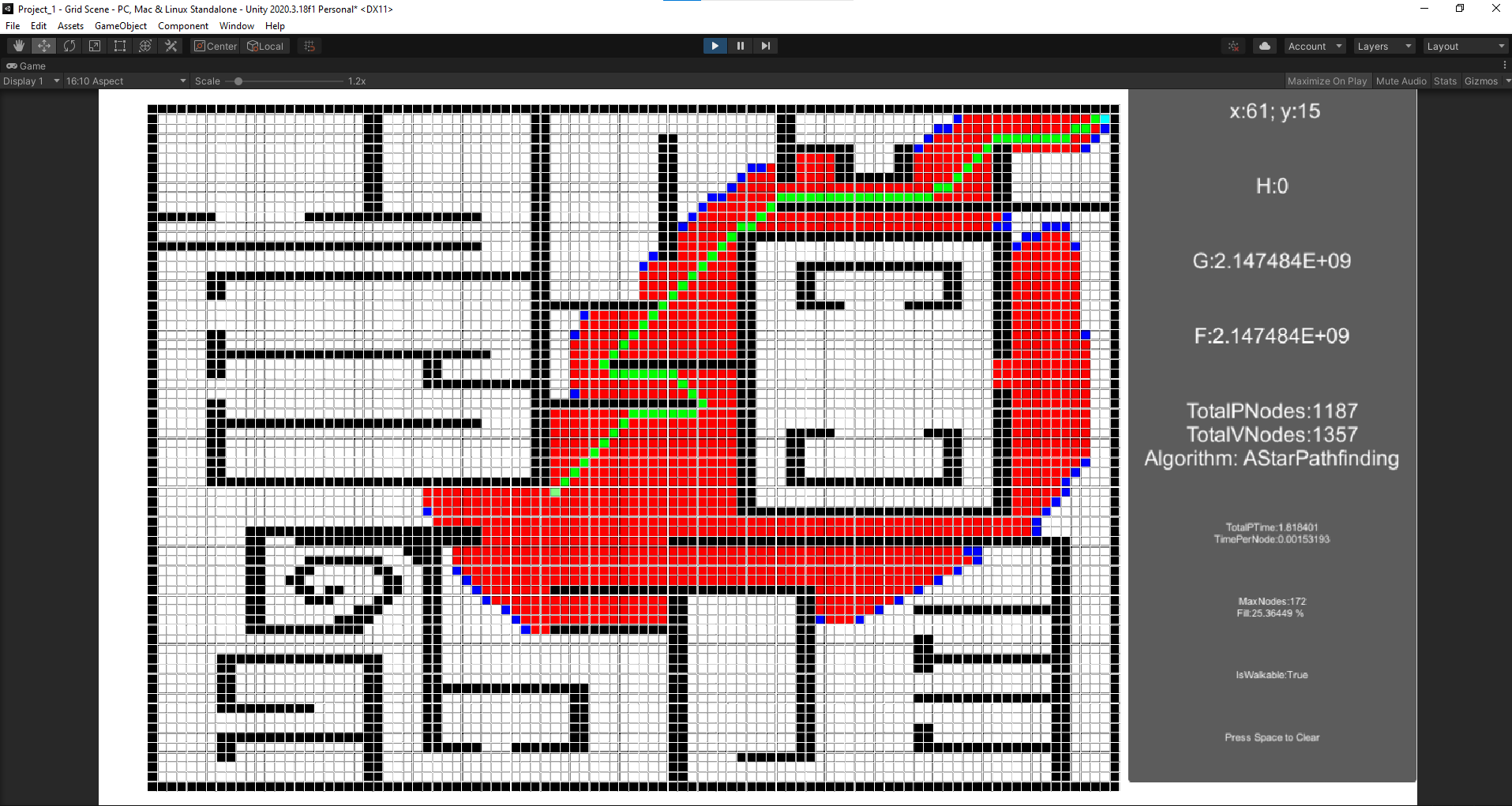


Figure - Test case 5 for A\*, no tie breaker

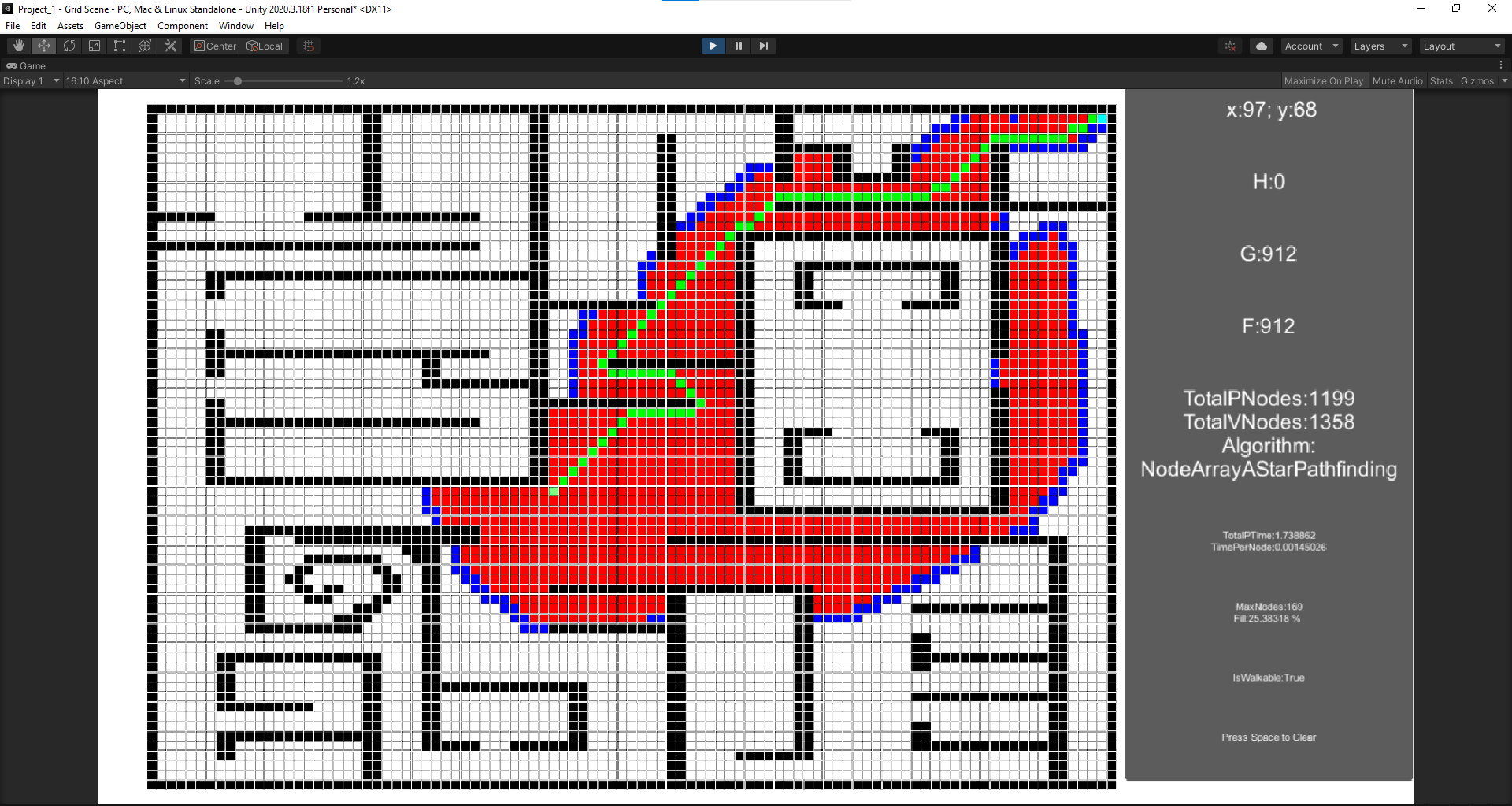


Figure - Test case 5 for Node Array A\*

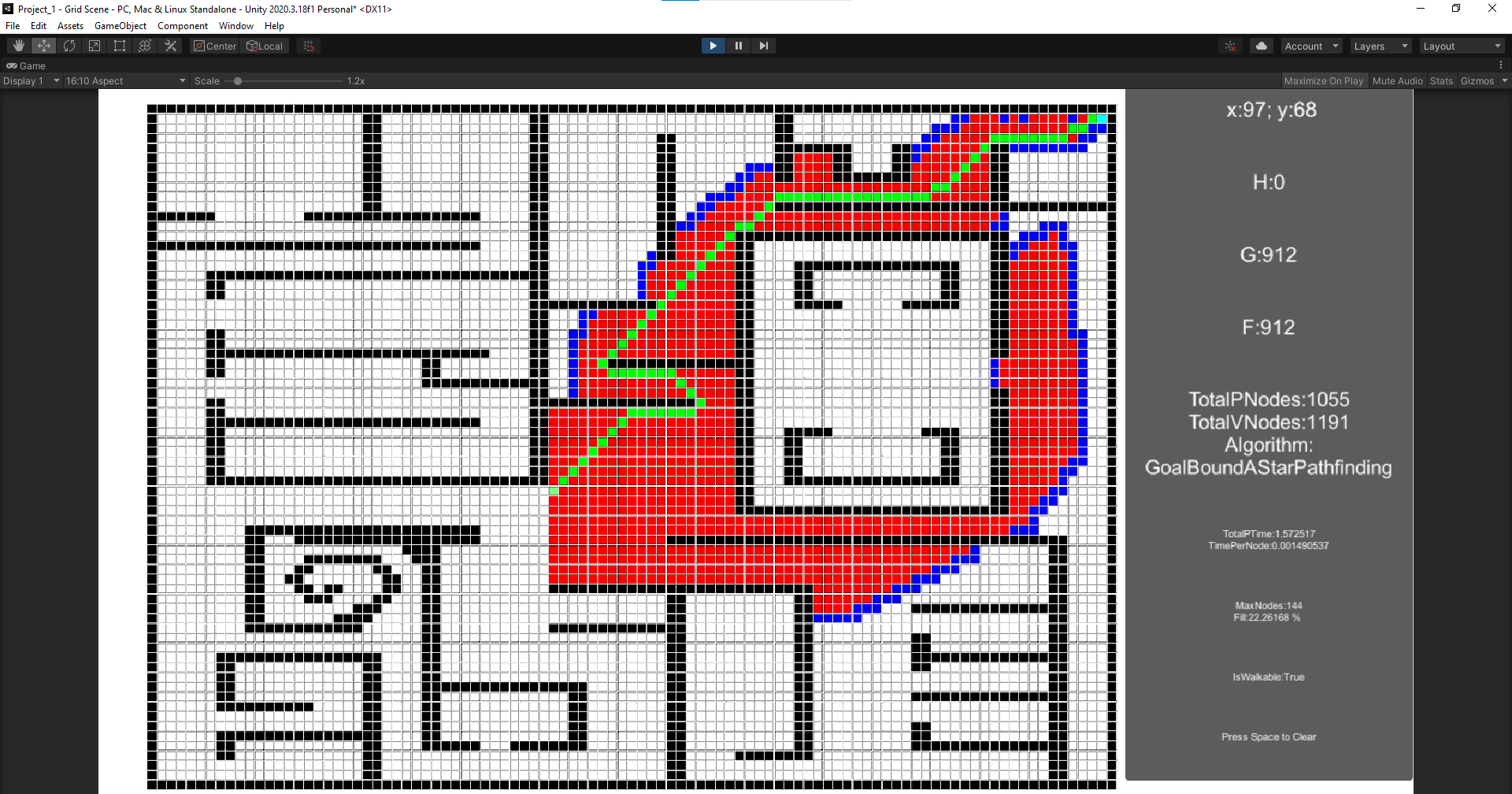


Figure - Test case 5 for Goal Bound

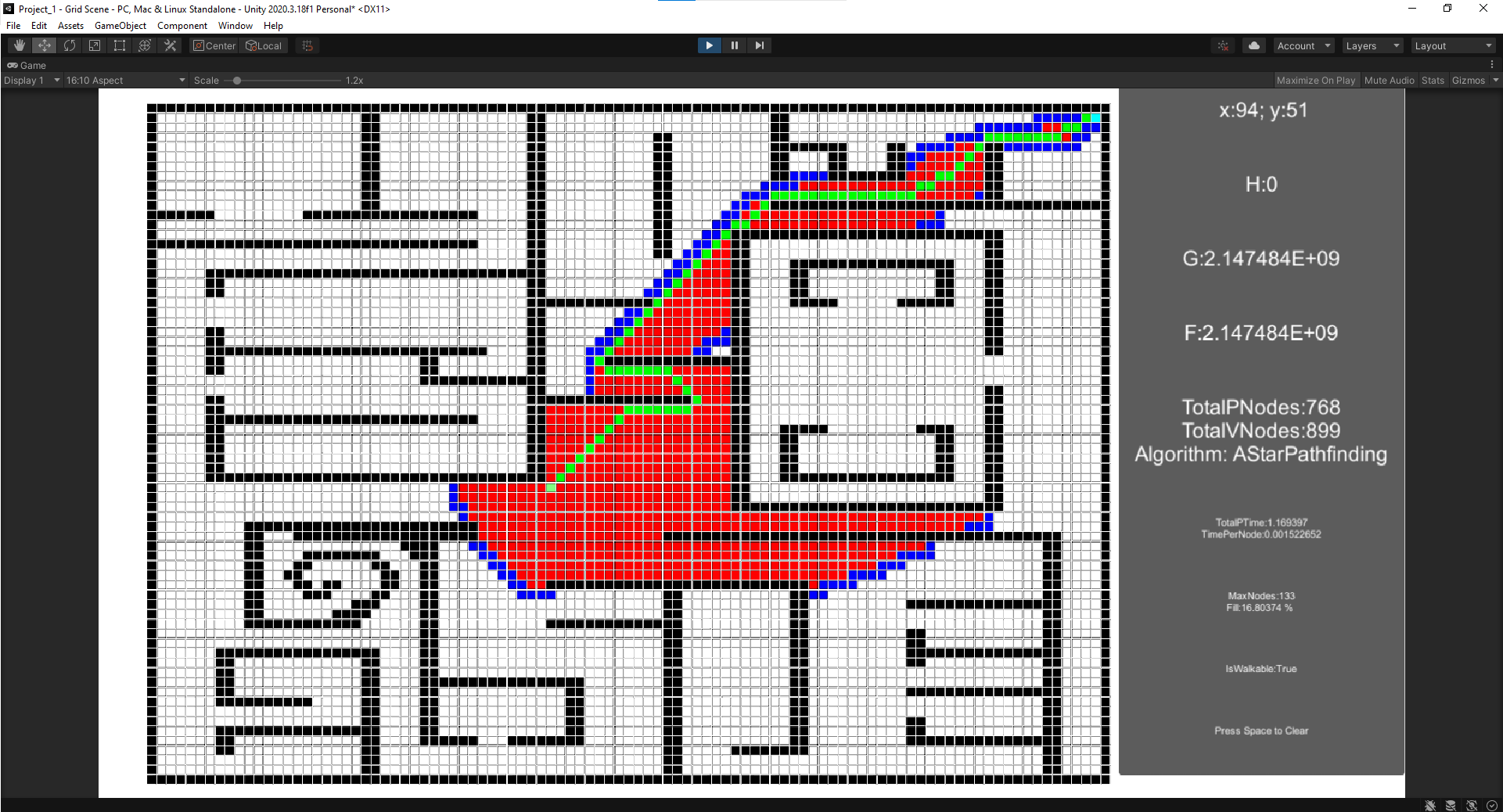


Figure - Test case 5 for A\* with tie breaker



Figure - Test case 6 for A\*, no tie breaker

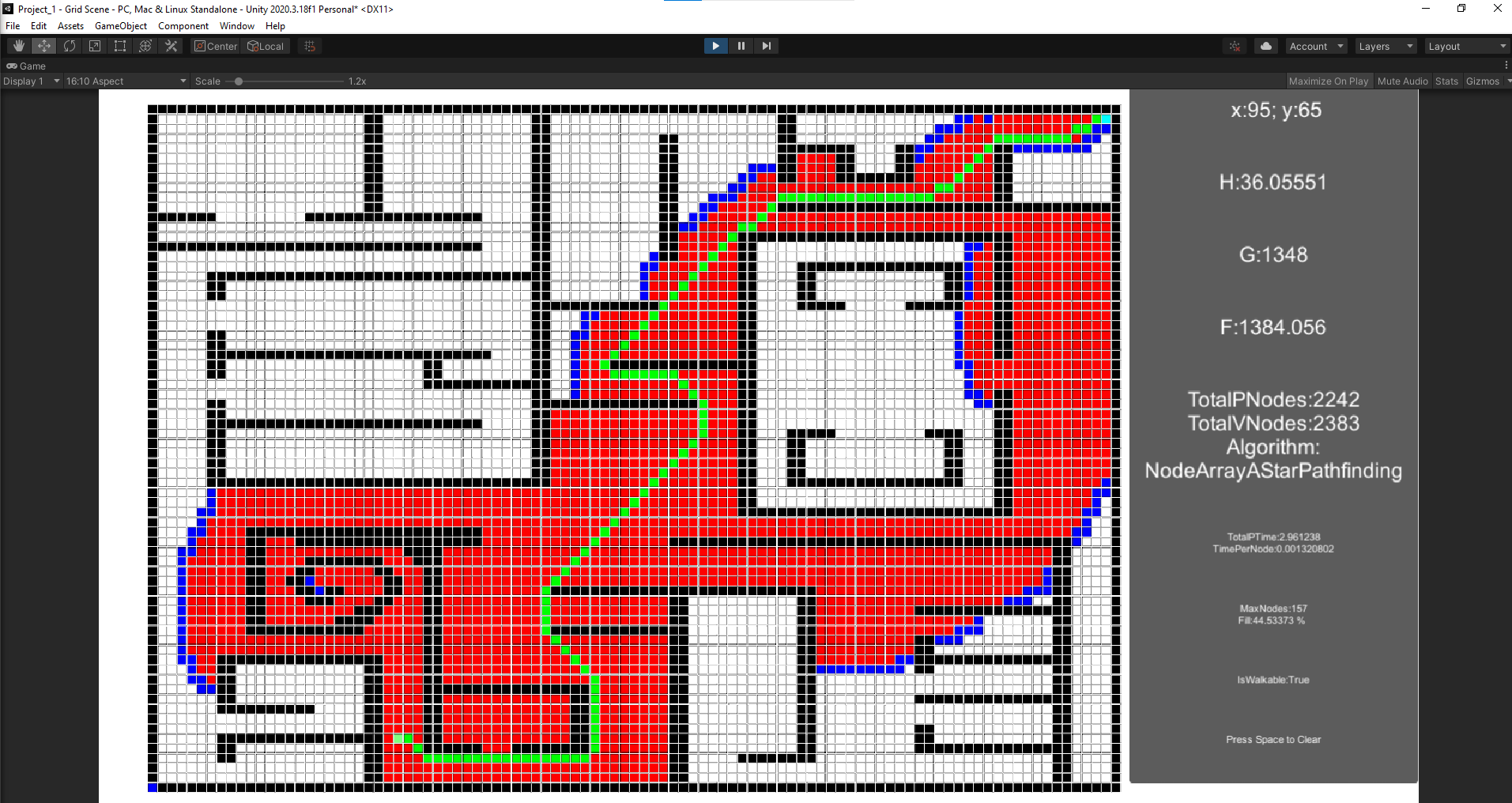


Figure - Test case 6 for Node Array A\*

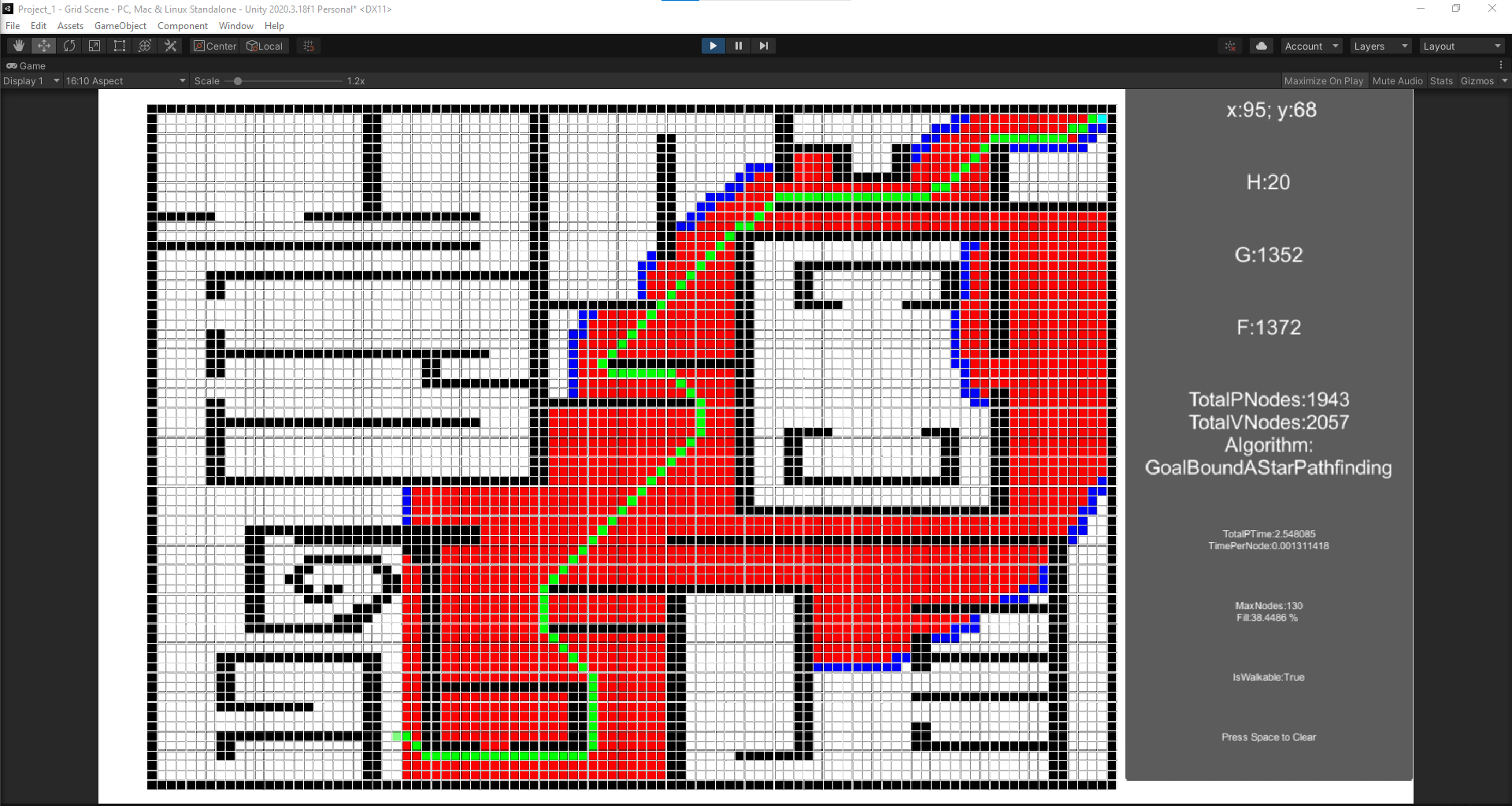


Figure - Test case 6 for Goal Bound

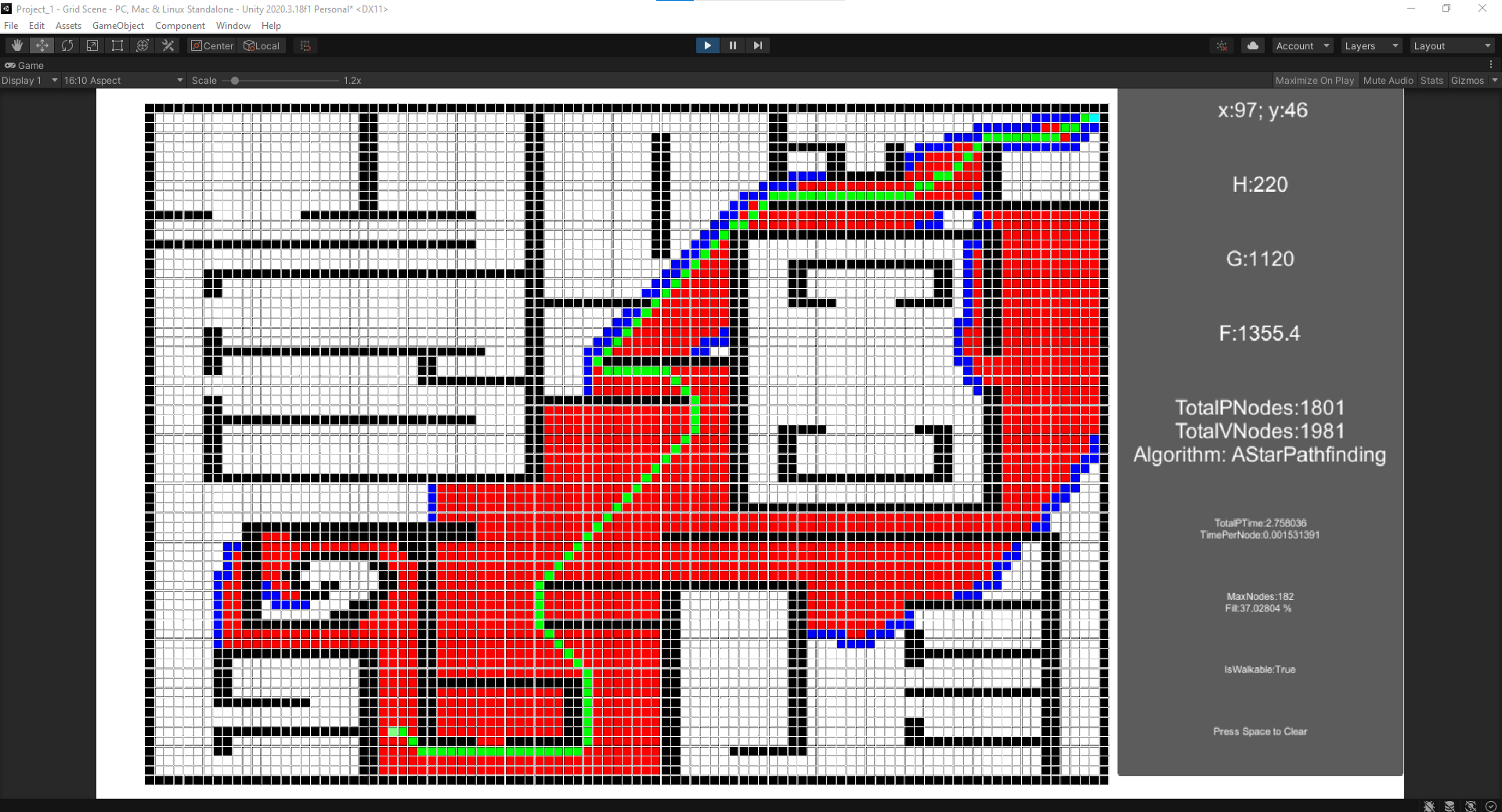


Figure - Test case 6 for A\* with tie breaker