Imane Chtaini , Souleiman Oulmaati

CSC 4301: Introduction to Artificial Intelligence

Dr. Rachidi Tajjeeddine

June 12th; 2022

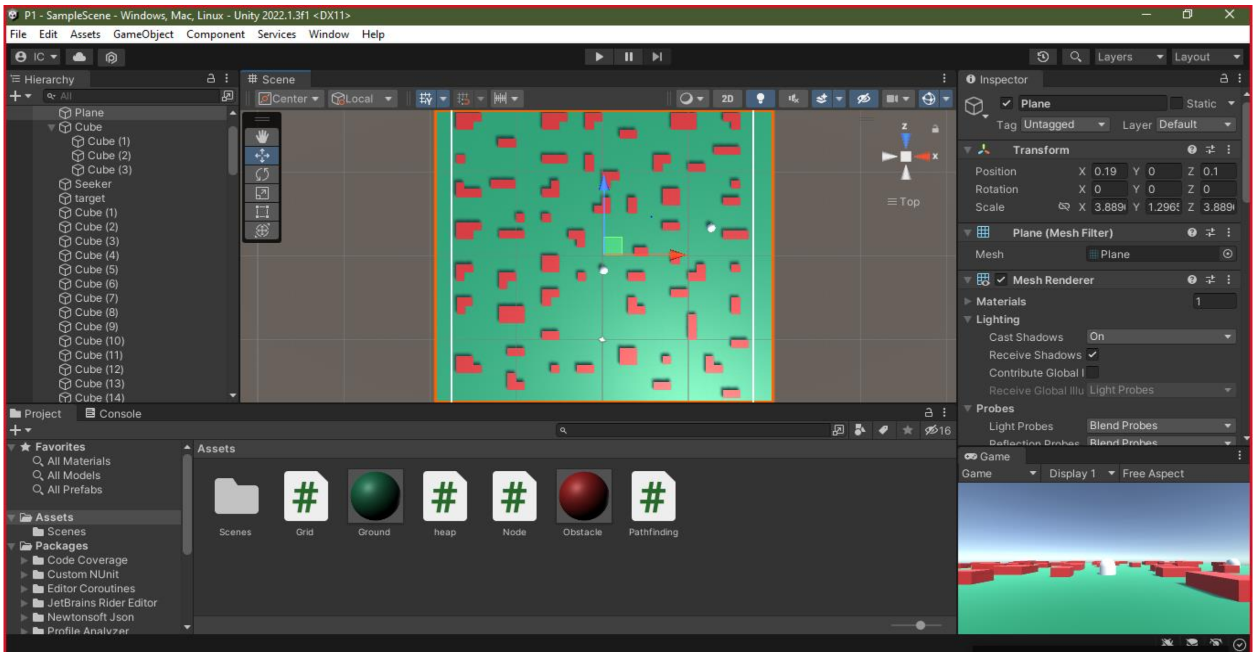
Introduction:

In class, we discussed different searching algorithms, so our instructor Dr. Rachidi Tajjeeddine gave us this project which consists of moving from theory to practice by implementing those algorithms in an environment using unity. we first started by watching few videos that were given as a requirement for this project to understand more the concept and how unity works. Indeed, we did the work thanks to the help of Sebastian Lague who shared his knowledge for free in youtube. Hence, the source code to implement the basics of this project was already given. The source code was in C# , so we updated it and appended it on what we were asked for in this project. Our programming background of Object oriented programming that we have studied in java before helped us to understand well how C# works, so we did not have any problem regarding that. That project was a good initiative that helped us to learn more about searching algorithms, since we were able to implement them in the real world.

Steps done:

- We first started by creating our own environment in order to start finding different search paths.
- The design of our environment was not based on any specific idea but mainly to make it as complex as possible in order to view or see the difference between different search paths.
- We then created a seeker and a target which basically will serve as a starting node and a goal node.
- After implementing the code that was already given in github.
- We started to adjust it in order for it to fit different search methods such as DFS which start by exploring the deepest node by using an abstract data type: a LIFO stack.
- Then BFS which explore all nodes around the shallowest solution. It uses as an abstract data type: a FIFO queue.
- For UCS, we expand the cheapest node first so it needs to know the cost of each path. It uses a priority queue as an abstract data type.
- For A*, it is a combination of both backward cost(distance from the goal state) and an estimate of forward cost(heuristic from the current node to the goal test).

Now let’s see what our environment looks like and what did we use on it.


P1 - SampleScene - Windows, Mac, Linux - Unity 2022.1.3f1 <DX11>

File Edit Assets GameObject Component Services Window Help

Hierarchy
- Plane
- Cube
  - Cube (1)
  - Cube (2)
  - Cube (3)
- Seeker
- target
- Cube (1)
- Cube (2)
- Cube (3)
- Cube (4)
- Cube (5)
- Cube (6)
- Cube (7)
- Cube (8)
- Cube (9)
- Cube (10)
- Cube (11)
- Cube (12)
- Cube (13)
- Cube (14)

Inspector

Plane — Static

Tag Untagged — Layer Default

Transform
- Position X 0.19 Y 0 Z 0.1
- Rotation X 0 Y 0 Z 0
- Scale X 3.889 Y 1.2965 Z 3.889

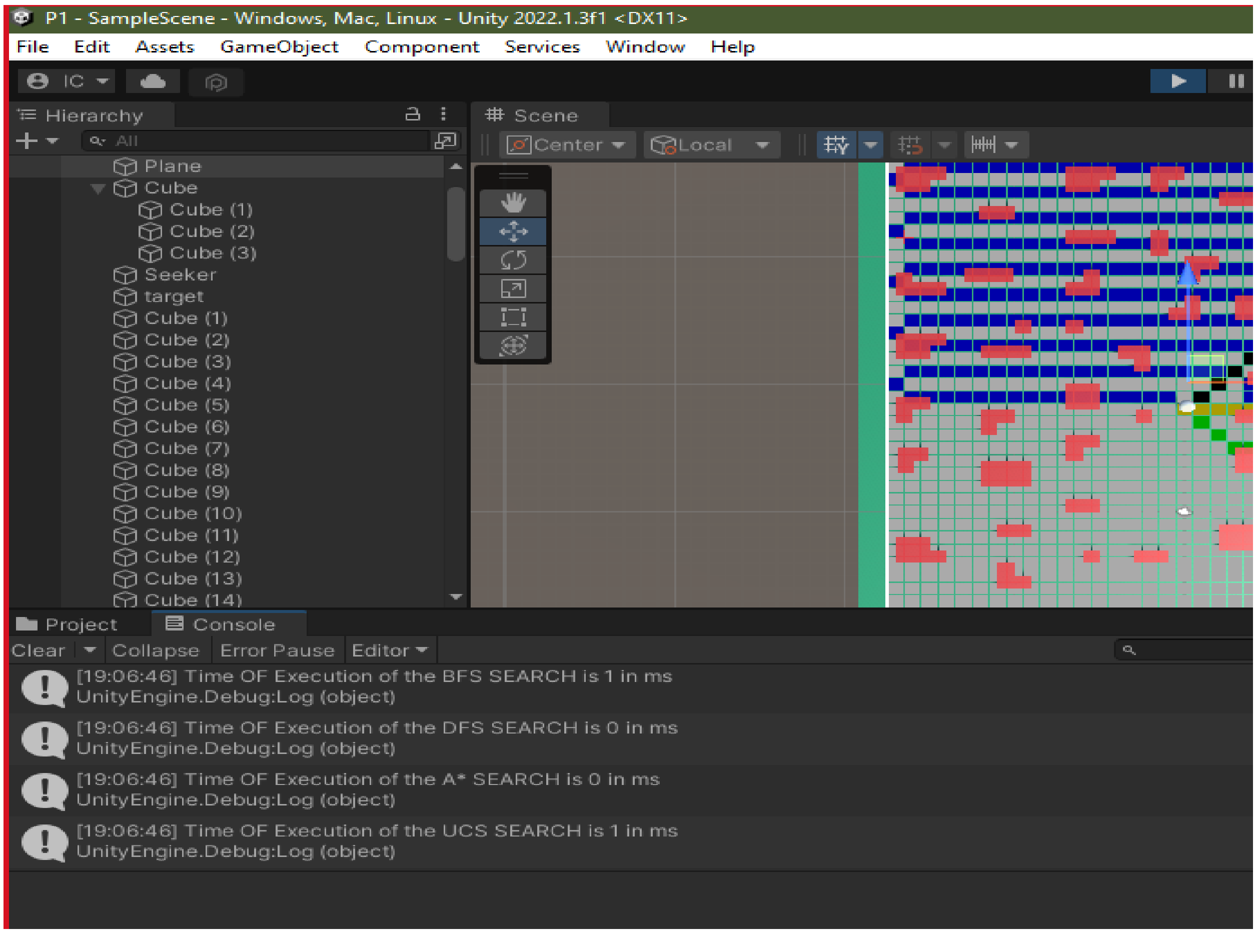
Plane (Mesh Filter)
- Mesh Plane

Mesh Renderer
- Materials 1
- Lighting
  - Cast Shadows On
  - Receive Shadows ✓
  - Contribute Global I
  - Receive Global Illu Light Probes
- Probes
  - Light Probes Blend Probes
  - Reflection Probes Blend Probes

Project — Assets
- Favorites
  - All Materials
  - All Models
  - All Prefabs
- Assets
  - Scenes
- Packages
  - Code Coverage
  - Custom NUnit
  - Editor Coroutines
  - JetBrains Rider Editor
  - Newtonsoft Json
  - Profile Analyzer

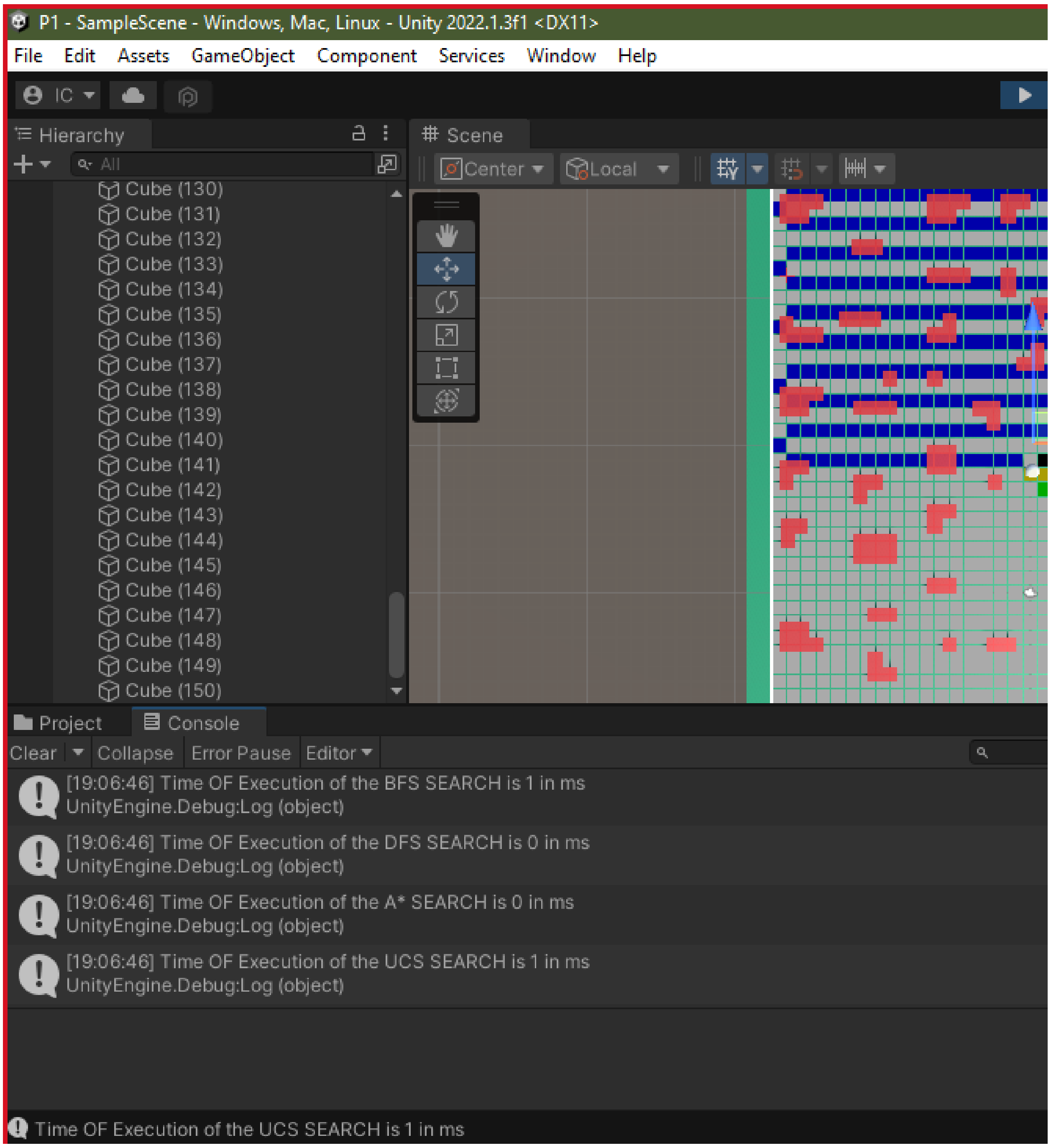
Assets: Scenes, Grid, Ground, heap, Node, Obstacle, Pathfinding

Here above we have the whole environment with the agent, target and obstacles. The obstacles are in red, and the ground is in green.



```
[19:06:46] Time OF Execution of the BFS SEARCH is 1 in ms
UnityEngine.Debug:Log (object)
[19:06:46] Time OF Execution of the DFS SEARCH is 0 in ms
UnityEngine.Debug:Log (object)
[19:06:46] Time OF Execution of the A* SEARCH is 0 in ms
UnityEngine.Debug:Log (object)
[19:06:46] Time OF Execution of the UCS SEARCH is 1 in ms
UnityEngine.Debug:Log (object)
```

we used a seeker which is the start state and a target which is the goal test.

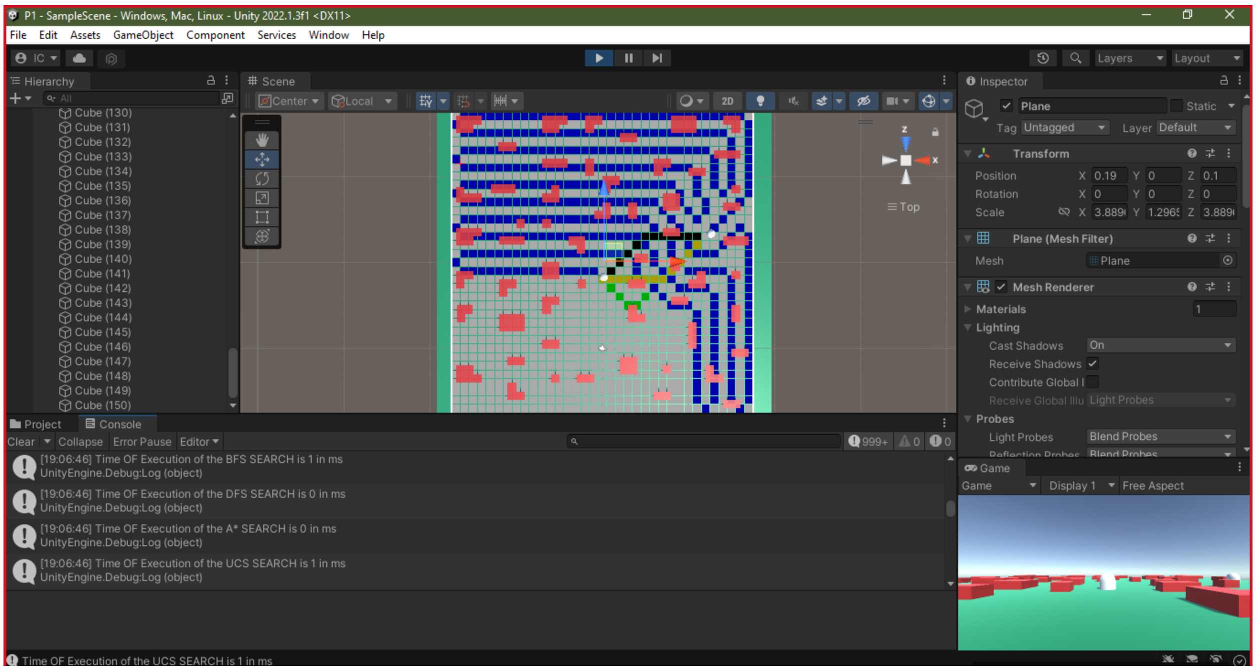

P1 - SampleScene - Windows, Mac, Linux - Unity 2022.1.3f1 <DX11>
File Edit Assets GameObject Component Services Window Help
Hierarchy
Scene
Center
Local
Project
Console
Clear
Collapse
Error Pause
Editor

Cube (130)
Cube (131)
Cube (132)
Cube (133)
Cube (134)
Cube (135)
Cube (136)
Cube (137)
Cube (138)
Cube (139)
Cube (140)
Cube (141)
Cube (142)
Cube (143)
Cube (144)
Cube (145)
Cube (146)
Cube (147)
Cube (148)
Cube (149)
Cube (150)

```
[19:06:46] Time OF Execution of the BFS SEARCH is 1 in ms
UnityEngine.Debug:Log (object)
[19:06:46] Time OF Execution of the DFS SEARCH is 0 in ms
UnityEngine.Debug:Log (object)
[19:06:46] Time OF Execution of the A* SEARCH is 0 in ms
UnityEngine.Debug:Log (object)
[19:06:46] Time OF Execution of the UCS SEARCH is 1 in ms
UnityEngine.Debug:Log (object)

Time OF Execution of the UCS SEARCH is 1 in ms
```

We used 150 cubes



```
[19:06:46] Time OF Execution of the BFS SEARCH is 1 in ms
UnityEngine.Debug:Log (object)
[19:06:46] Time OF Execution of the DFS SEARCH is 0 in ms
UnityEngine.Debug:Log (object)
[19:06:46] Time OF Execution of the A* SEARCH is 0 in ms
UnityEngine.Debug:Log (object)
[19:06:46] Time OF Execution of the UCS SEARCH is 1 in ms
UnityEngine.Debug:Log (object)
```

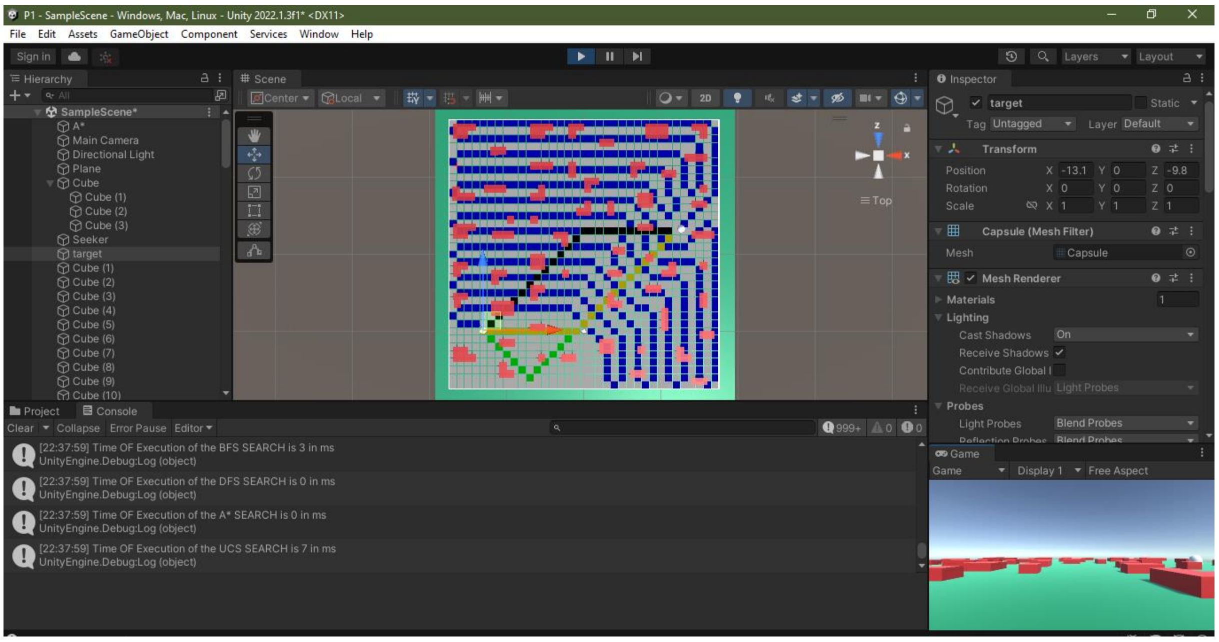
Here different paths are displayed in addition to the timing that corresponds to each one.

Green path is the BFS path

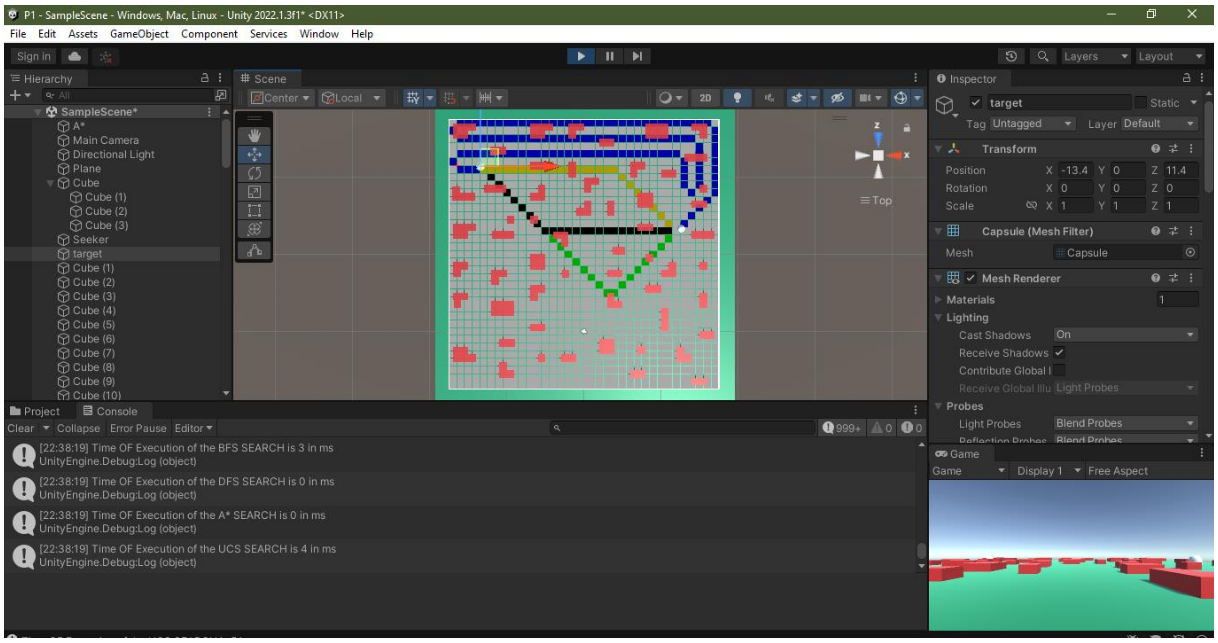
Blue path is the DFS

Black path is the UCS

Yellow path is A*

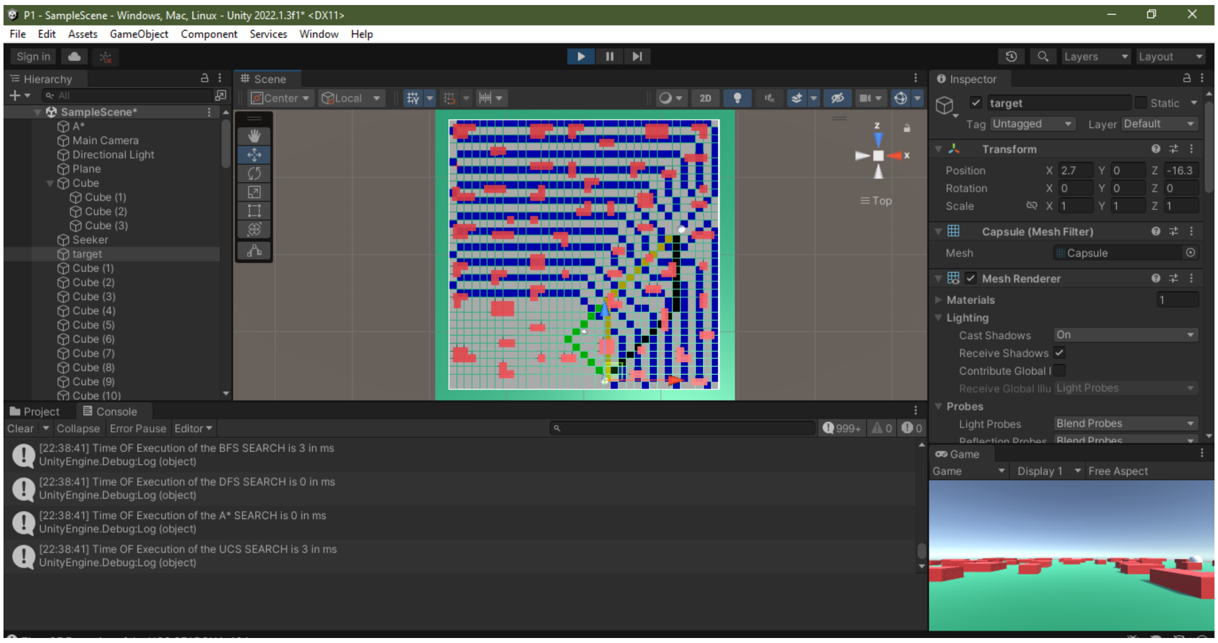


```
[22:37:59] Time OF Execution of the BFS SEARCH is 3 in ms
UnityEngine.Debug:Log (object)
[22:37:59] Time OF Execution of the DFS SEARCH is 0 in ms
UnityEngine.Debug:Log (object)
[22:37:59] Time OF Execution of the A* SEARCH is 0 in ms
UnityEngine.Debug:Log (object)
[22:37:59] Time OF Execution of the UCS SEARCH is 7 in ms
UnityEngine.Debug:Log (object)
```



```
[22:38:19] Time OF Execution of the BFS SEARCH is 3 in ms
UnityEngine.Debug:Log (object)
[22:38:19] Time OF Execution of the DFS SEARCH is 0 in ms
UnityEngine.Debug:Log (object)
[22:38:19] Time OF Execution of the A* SEARCH is 0 in ms
UnityEngine.Debug:Log (object)
[22:38:19] Time OF Execution of the UCS SEARCH is 4 in ms
UnityEngine.Debug:Log (object)
```

target — Position X -13.4 Y 0 Z 11.4; Rotation X 0 Y 0 Z 0; Scale X 1 Y 1 Z 1

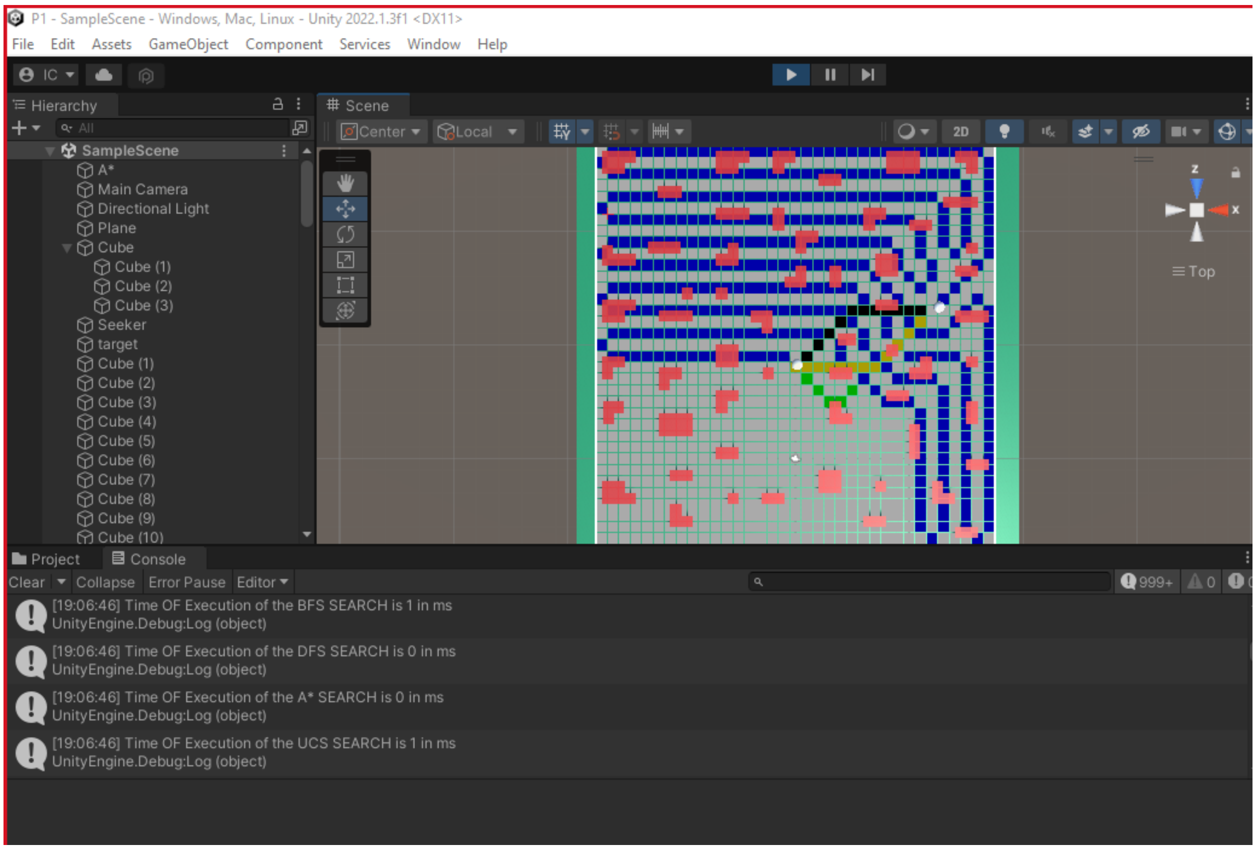


```
[22:38:41] Time OF Execution of the BFS SEARCH is 3 in ms
UnityEngine.Debug:Log (object)
[22:38:41] Time OF Execution of the DFS SEARCH is 0 in ms
UnityEngine.Debug:Log (object)
[22:38:41] Time OF Execution of the A* SEARCH is 0 in ms
UnityEngine.Debug:Log (object)
[22:38:41] Time OF Execution of the UCS SEARCH is 3 in ms
UnityEngine.Debug:Log (object)
```

target — Position X 2.7 Y 0 Z -16.3; Rotation X 0 Y 0 Z 0; Scale X 1 Y 1 Z 1

<u>Analysis:</u>

In the three previous screenshots, we notice that the path changes depending on our choice of the start node. Also, for BFS and UCS, the time of execution differs when we change the position of the agent. For A* and DFS, the time remains the same(0ms) so those 2 algorithms are very optimal in our environment no matter where we put the agent.



```
[19:06:46] Time OF Execution of the BFS SEARCH is 1 in ms
UnityEngine.Debug:Log (object)
[19:06:46] Time OF Execution of the DFS SEARCH is 0 in ms
UnityEngine.Debug:Log (object)
[19:06:46] Time OF Execution of the A* SEARCH is 0 in ms
UnityEngine.Debug:Log (object)
[19:06:46] Time OF Execution of the UCS SEARCH is 1 in ms
UnityEngine.Debug:Log (object)
```

As a result of our testing:

- BFS took 1ms to 8ms
- DFS took 0ms
- A* took 0ms
- For UCS took 1ms to 7ms

Conclusion:

As a conclusion we can see that BFS and UCS took more time than DFS and A* which means that DFS and A* are less optimal in the environment that we created. A* and DFS are the most efficient solutions in our environment. Mostly but not always, A* is quicker than UCS. UCS expands the least cost node in every direction because it was not provided with any information about the target. On the other hand, A* has a clear information about the test goal through an admissible heuristic. The result we got makes sense. Here in our specific case, DFS was quicker than BFS, but it does happen