**Acknowledgments**

I would like to thank Ross Palmer, my supervisor for providing me with a lot of reference materials and ideas.

**Project Abstract**

Pathfinding over the last few decades and resulted in a lot of different approaches while many solutions are more applicable depending on the computer application my particular deep dive into it will be finding the most efficient of these to use in a game and by extension compare a hybrid to some solutions individually

**Project Introduction and Research Question**

To investigate context-sensitive steering approaches and develop a hybrid AI which utilizes A\* and context-sensitive steering in unison and compare it to other steering solutions. The entity will swap between the two systems and use both to navigate to a point somewhere in the simulation.

**Literature Review**

**Pathfinding:**

Pathfinding is computing the shortest route between two points in a computer application using some kind of algorithm or software. There are several different approaches to pathfinding and there are multiple variants of those approaches.There are also situations where the program knows where it’s supposed to go and calculates the shortest path there. All pathfinding algorithms follow some fundamentals and may refer to them by a different name but will often serve the same purpose. To begin with an algorithm needs to have a starting point to search from some will use this starting point to compare the vertices(the cost associated with moving to that point)to the next point and others will simply use it as the starting point and calculate outwards checking every single point until they find what they’re looking for and/or calculate the entire graph/map. I mentioned graph in my last point. Not all algorithms will use a graph but the likes of A\* which I’ll talk about later would where as simpler algorithms such as “seek” or “flee” don’t require this as they simply need a point that they are seeking or fleeing from in order to calculate their velocity. Knowing where a program is supposed to go also makes it possible to compute bi-directionally(calculating the route from the end and starting point simultaneously and moving towards each other) as shown in Fig.1[2] This was conceptualized by a computer scientist Edsger W. Dijkstra [1] in 1959 and his algorithm appears unfocused and checks every vertex, working outwards until it finds its objective(Fig.2), which can be very time consuming.

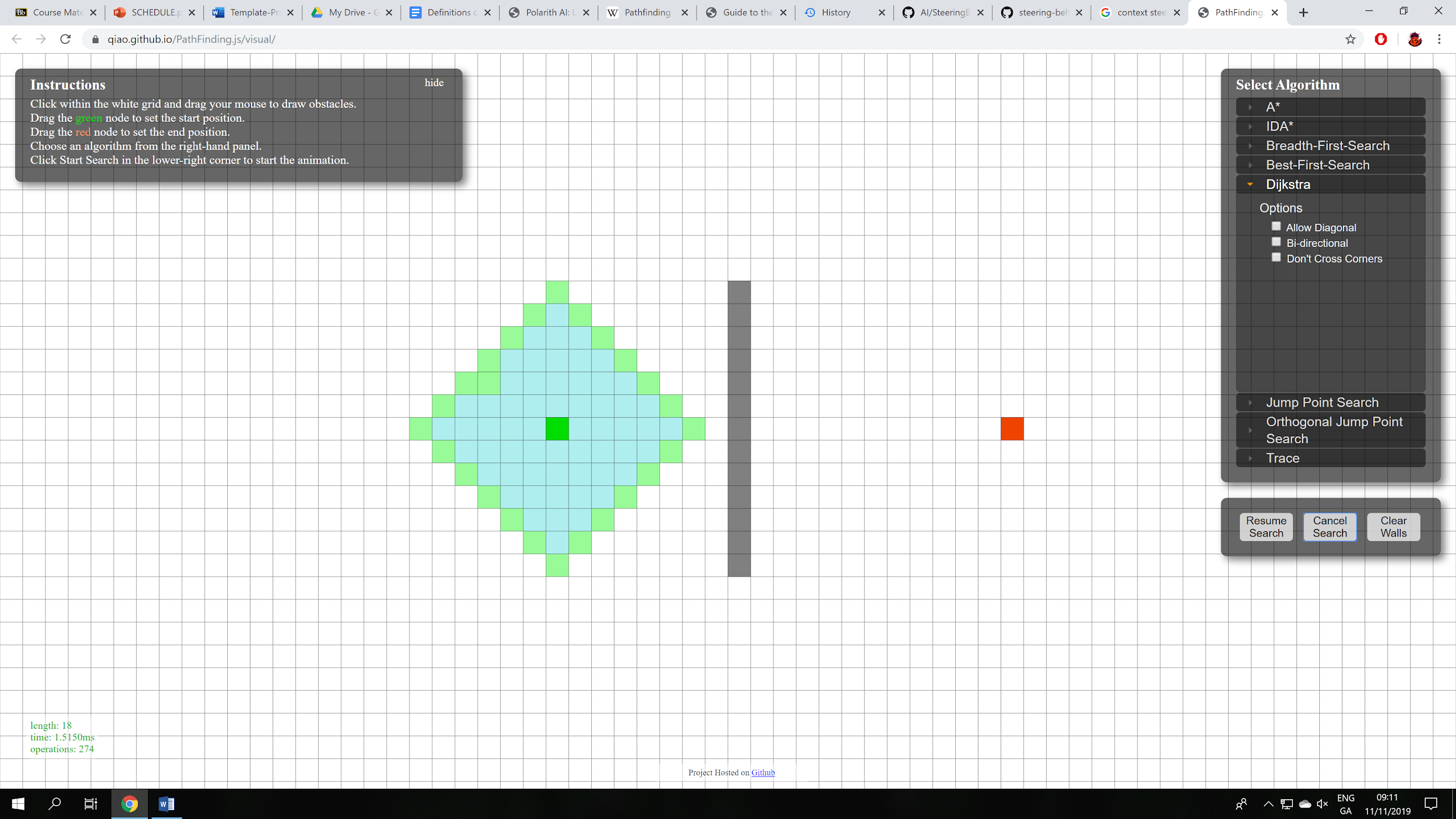
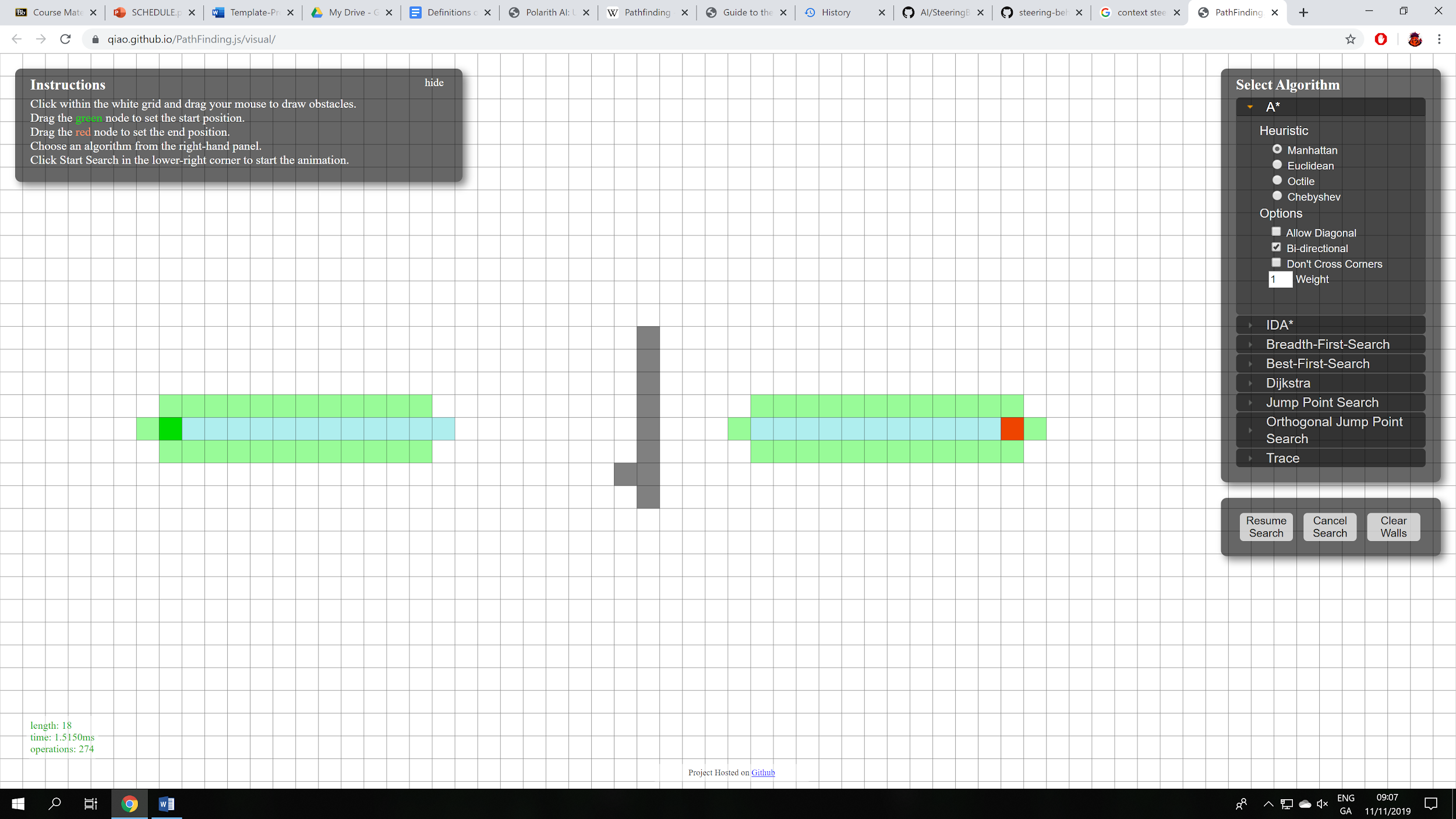


Fig.1 Fig.2

**A\*:**

A\* shown in Fig.1(example is bidirectional) is a graph traversal and path search algorithm. It works by storing nodes and calculating the routes between them all(heuristic), resulting in a “weighted-graph”. This “weighted graph” or heuristic “represents a minimum possible distance between that node and the end. This allows it to eliminate longer paths once an initial path is found.”[5] It can be seen as an extension of Dijkstra’s algorithm[3] shown in Fig.2[4]. “Dijkstra’s algorithm works by visiting vertices in the graph starting with the objects starting point. It then repeatedly examines the closest not-yet-examined vertex, adding its vertices to the set of vertices to be examined. It expands outwards from the starting point until it reaches its goal”[4]. Greedy best-search first improved upon this by weighing which nodes were closer to the objective(Fig.3)[4]. Both of these solutions only work when there are no obstacles and are only accurate “when the map has no obstacles, and the shortest path really is a straight line”[4]. If we take a look at Fig.4[4]and Fig.5[4] we see that this is how Djikstra’s and greedy best-search-first behave when objects are introduced.

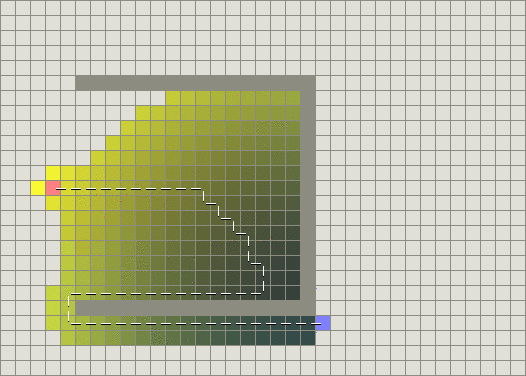
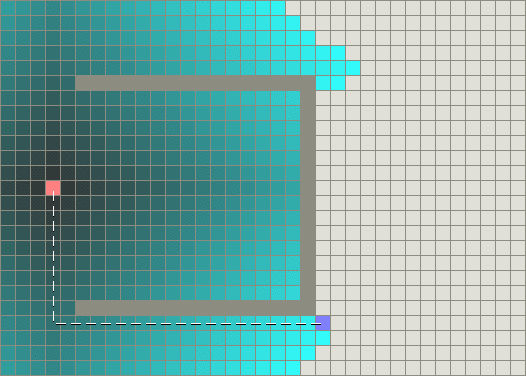
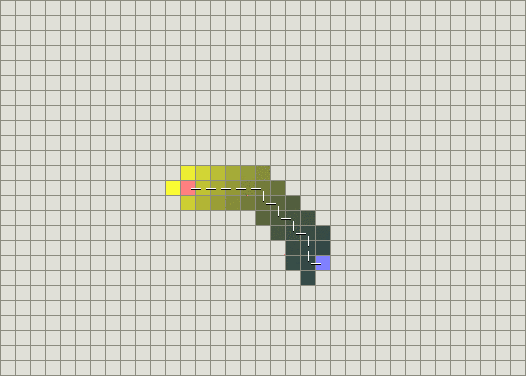


Fig.3 Fig.4 Fig.5

**Context-free steering:**

Context-free steering is specific to each game and by design does not take “context” into account when making steering decisions but instead performs calculations when a steering decision must be made. This can be very expensive the more objects you have in the game that needs to make these decisions and can provide results at the expense of smoothness and realism. Craig Reynolds[6] broke these decisions down into 3 parts: “Action Selection”, “Steering” and “Locomotion”. Action selection refers to the purpose or what needs to be done. An example of this would be seek which I mentioned earlier it’s “Action Selection” would be to seek and move towards its intended target.”Steering” is the “path determination” or pathfinding and “Locomotion” is the movement or adjustment itself and by extension in games the animation. Examples of AI context-free steering include the behaviours: seek(Fig.7), flee and flock. Seek and flee I’ve mentioned before,Flock works in a similar manner to seek except a group of objects will have one designated as the leader while the others will follow in whatever direction it goes with a bit of an offset. This is extremely useful for simulating bats or birds(Fig.8) or random enemies in a level.

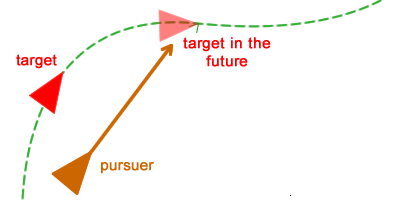
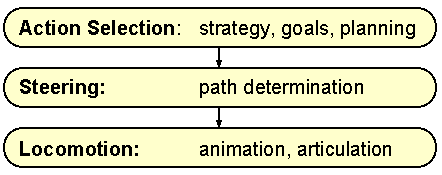


Fig.6[6] Fig.7[7] Fig.8[8]

**Context-sensitive steering:**

Context-sensitive steering tries to simplify the decision making process by representing the “context” of a decision numerically or “if the application requires a small number of entities that interact individually with the player, like a racing game”[8]. Andrew Fray explains how he did this on a game he worked on in his paper. The directions one can travel in, the desirable outcome and the danger are converted into a numerical value. This really simplifies the steering decision down into “go in the direction of the largest value” and really cuts down computational costs associated with steering algorithms. Direction can vary in complexity and will be specific to each game but will simply be a range of variables representing a given direction. The desirable outcome refers to what the autonomous body is trying to do such as chasing another entity or winning a race etc. Danger can be many things but in regards to steering it’s primarily obstacles.

In Fig.9 the circle represents the AI each different coloured arrow represents the different directions it can go in and are coloured simply for illustration purposes. The context map will be used for interest and danger with each direction represented numerically. In Fig.10 we can see an example of this in use. The largest direction on the danger map is the light blue arrow which we can see is directly perpendicular to the object. That would mean it’s numerical value would be greater than any other direction so if the range was 0-1, then the blue arrow might be estimated at 0.8 and the only other arrow in the danger map would have to be less than that. On the interest map we can also see that the direction of the shortest path to a target has the largest arrow on the interest “context map” and again this arrow would have the largest numerical value in the interest “context map”.

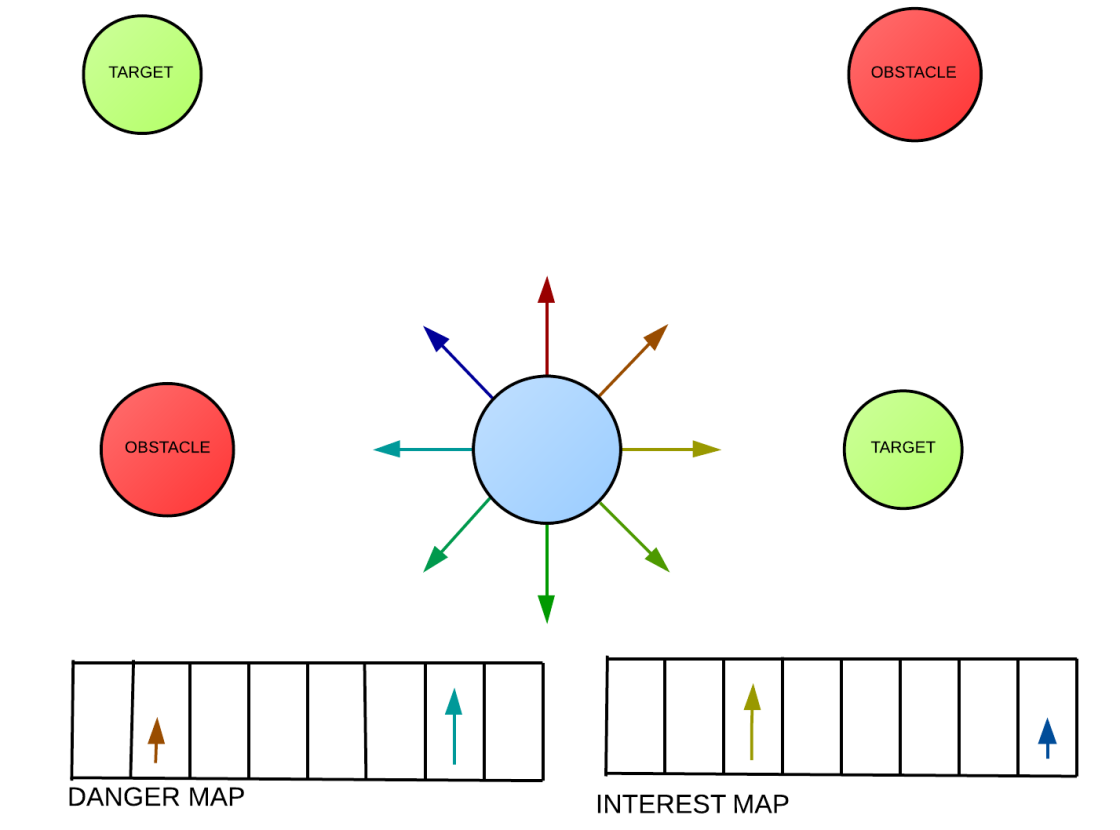
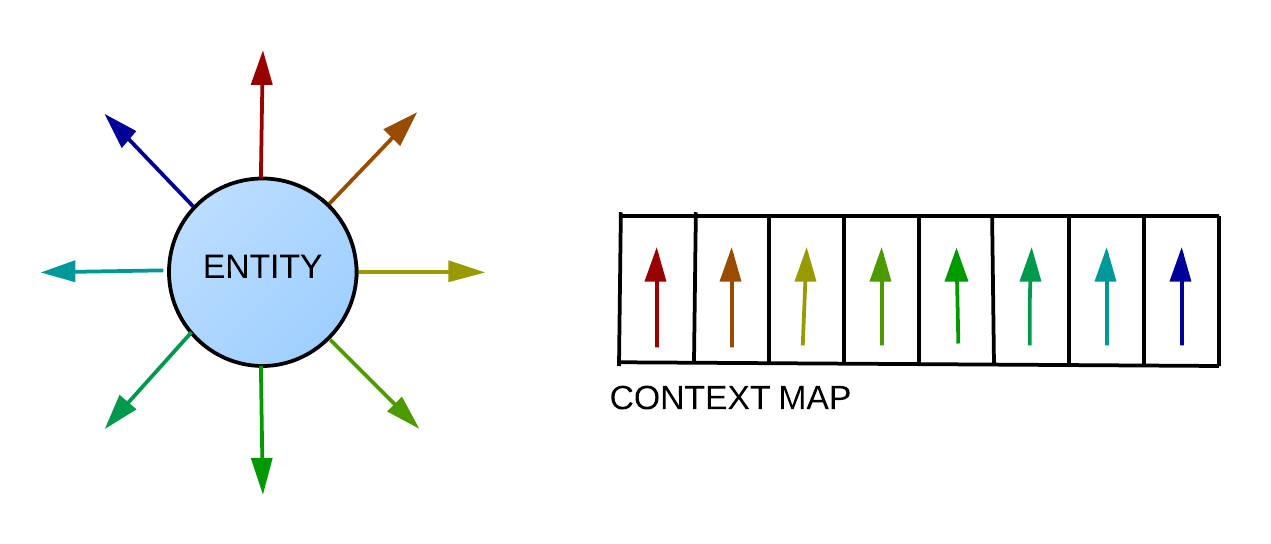


Fig.9[8] Fig.10[8]

**References:**

[1]In 1956, Edsger W. Dijkstraitte created a “shortest path first” algorithm and published it in 1959. Pathfinding, wikipedia accessed at: <https://en.wikipedia.org/wiki/Pathfinding>(accessed 21st Oct 2019).

[2] Pathfinding simulator accessed at: <https://qiao.github.io/PathFinding.js/visual/> (accessed on Nov 6th 2019).

[3] It was published by Peter Hart, Nils Nilson and Betram Raphael in 1965. Pathfinding, wikipedia accessed at: <https://en.wikipedia.org/wiki/Pathfinding>(accessed 21st Oct 2019).

[4] A\* comparison at theory stanford accessed at: <http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html> (accessed on 9th Nov 2019).

[5]Pathfinding, wikipedia accessed at: <https://en.wikipedia.org/wiki/Pathfinding>(accessed 21st Oct 2019).

[6] Craig Reynolds GDC paper 2019 accessed at: <https://www.red3d.com/cwr/steer/gdc99/> (accessed on 9th Nov 2019).

[7]Understanding steering behaviours at gamedevelopment.tutplus.com accessed at: <https://gamedevelopment.tutsplus.com/tutorials/understanding-steering-behaviors-pursuit-and-evade--gamedev-2946> (accessed on 9th Nov 2019)

[8] Context sensitive steering Andrew fray Accessed at: <https://andrewfray.wordpress.com/2013/03/26/context-behaviours-know-how-to-share/>

(accessed on 26th Oct 2019)