**Range Sight Algorithm**   
**by** [**Amit Patel**](http://www-cs-students.stanford.edu/%7Eamitp)

Someone asked me about an algorithm that, given a location of a unit, would tell you what locations can be reached in the next turn. I'm assuming you have a MovementCost function that tells you how hard it is to walk from one space to another. You can use Dijkstra's algorithm to find this area.

Let Costs be an array (as big as your map) of integers, init. value = -1

Function Find(UnitLocation)

Let Open be a priority queue of locations /\* sorted by Costs[x] \*/

Let Closed be a set of locations

Put Location in Open

While Open isn't empty:

Remove a location X from Open

Add X to Closed

If Costs[X] is less than the Movement Limit:

For each Y that is a neighbor of X:

NewCost = Costs[X] + MovementCost(from X to Y)

If Costs[Y] is -1 or NewCost is less than Costs[Y]:

Set Costs[Y] to NewCost

Add Y to Open if it's not already in Open

Let Results be a list of locations

For each X in Closed:

If Costs[X] is less than the Movement Limit:

Add X to Results

Set Costs[X] to -1

(It's important to keep Costs outside the Find function; otherwise the initialization time for that big array would slow down the whole algorithm. The algorithm restores Costs to -1 at the end so that you can use it the next time you want to Find.)

This algorithm should be pretty fast. In practice the number of things that go into Open should be the area (which are spaces you can reach) + the perimeter (which are spaces just out of reach), which is not too high as long as the unit's movement is pretty limited.