COMP 3200 Artificial Intelligence

Lecture 6

Grid Space Representations

Vector Fields

Notes

- In this lecture I will mention games as an example because that's what I work in
- Any time I say game, this could also apply to some real-world problems, robotics problems, simulations, etc
- Remember: everything is a game!

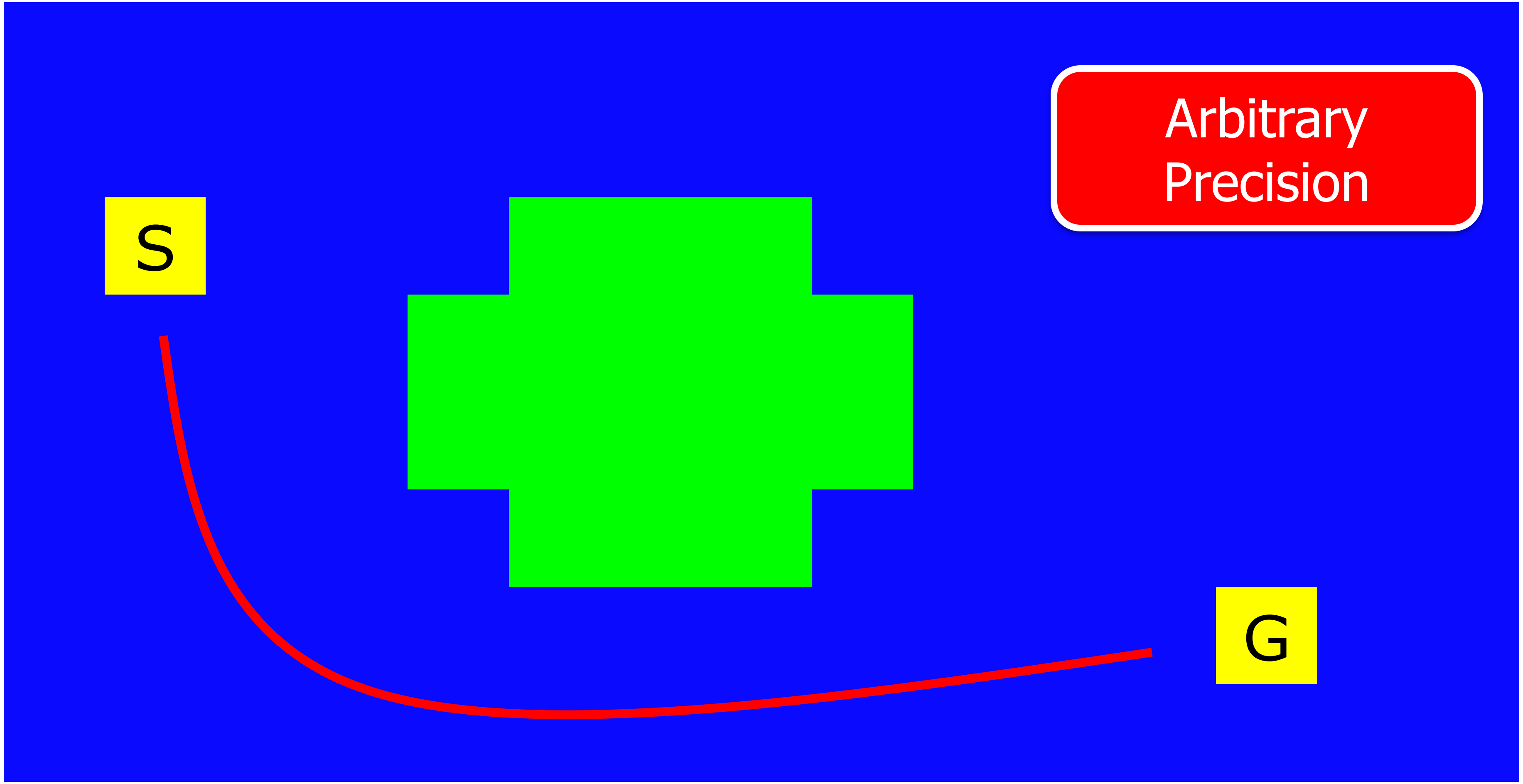
Space Representation

- How do we choose to represent the game/real world space for problems like path-finding
- Space and action representations are important decisions, ideas are linked
- Which representation to choose is based on many factors from the problem description
- The choice of space representation in the pathfinding algorithm will determine which movement actions can be performed

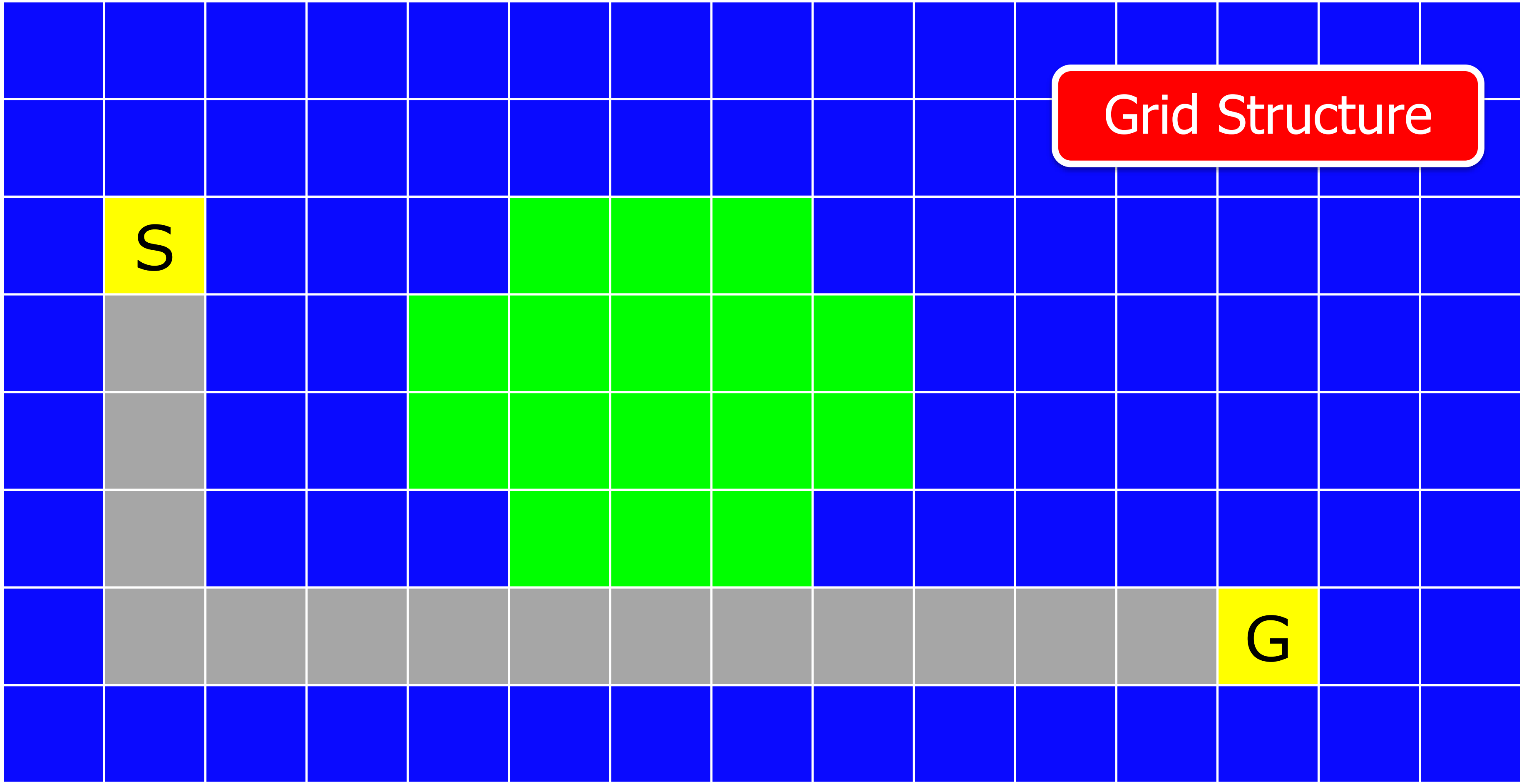
Representation Considerations

- Localization
  - How do you find where you are in the representation?
- Generation
  - How is the representation generated?
- Dynamic Changes
  - How do you handle dynamic changes?
- Planning / Pathfinding
  - How are actions computed / represented?
- Memory / Speed
  - How much memory / time required to compute paths?

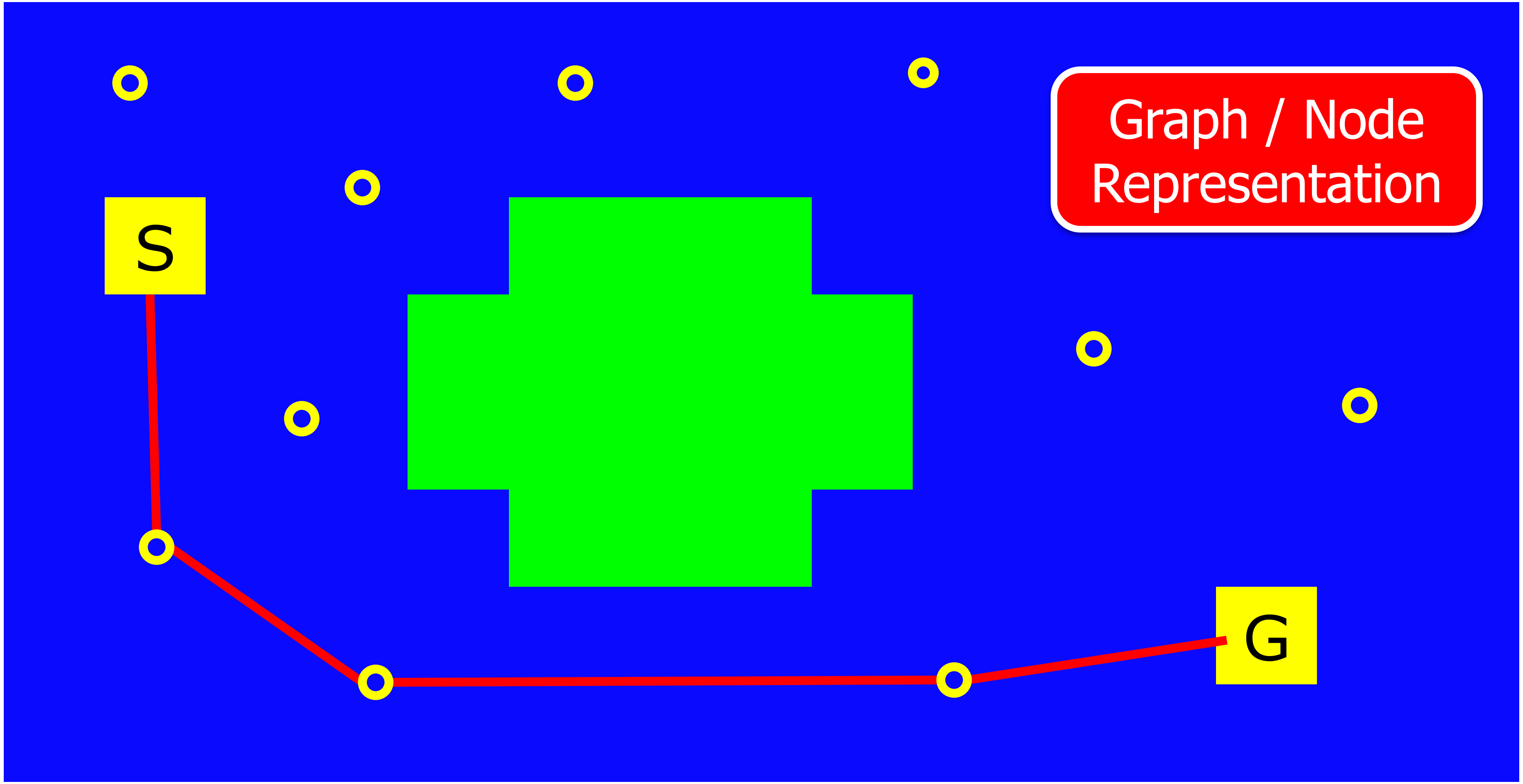
Space Representations


Arbitrary Precision
S
G

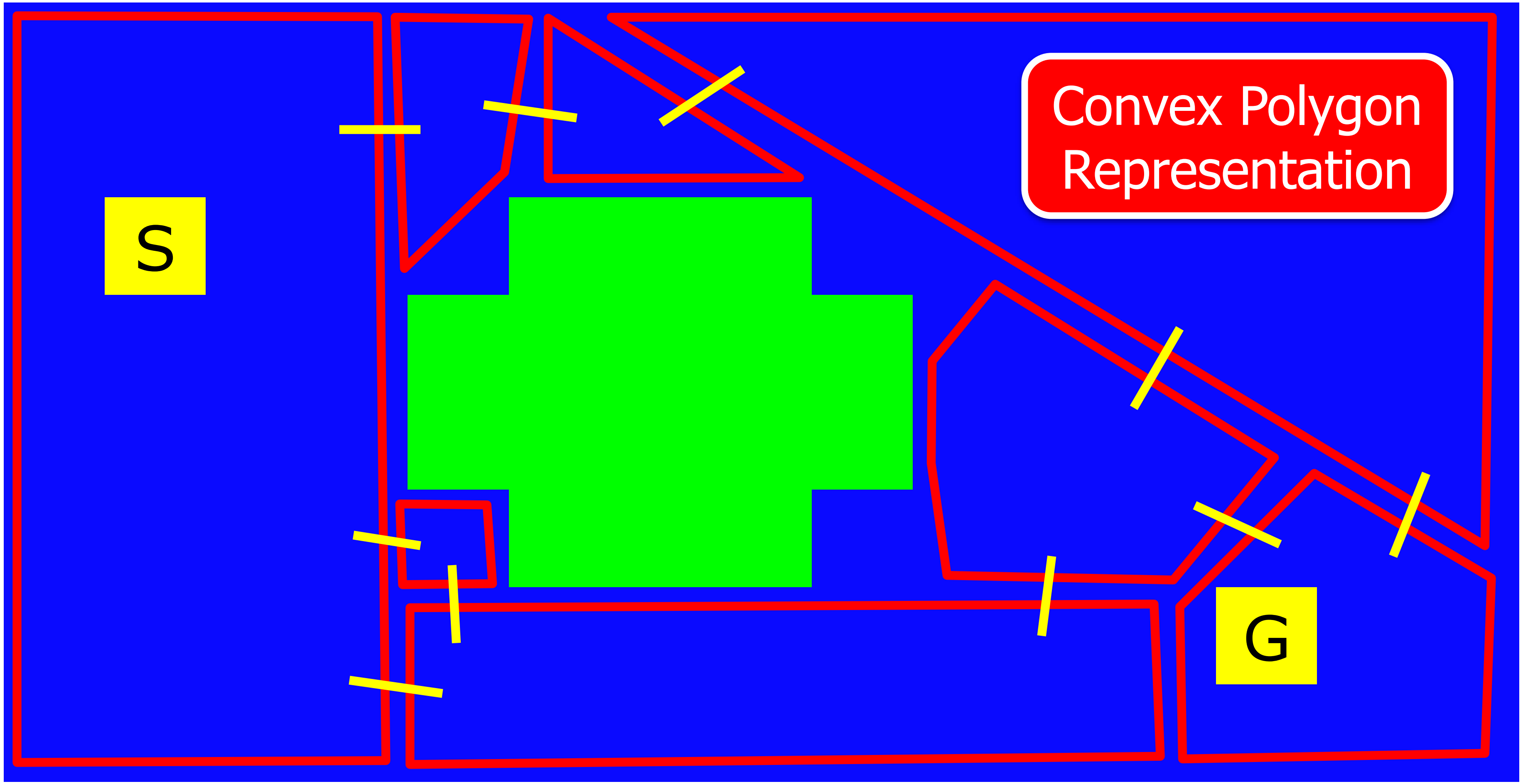
Grid Structure



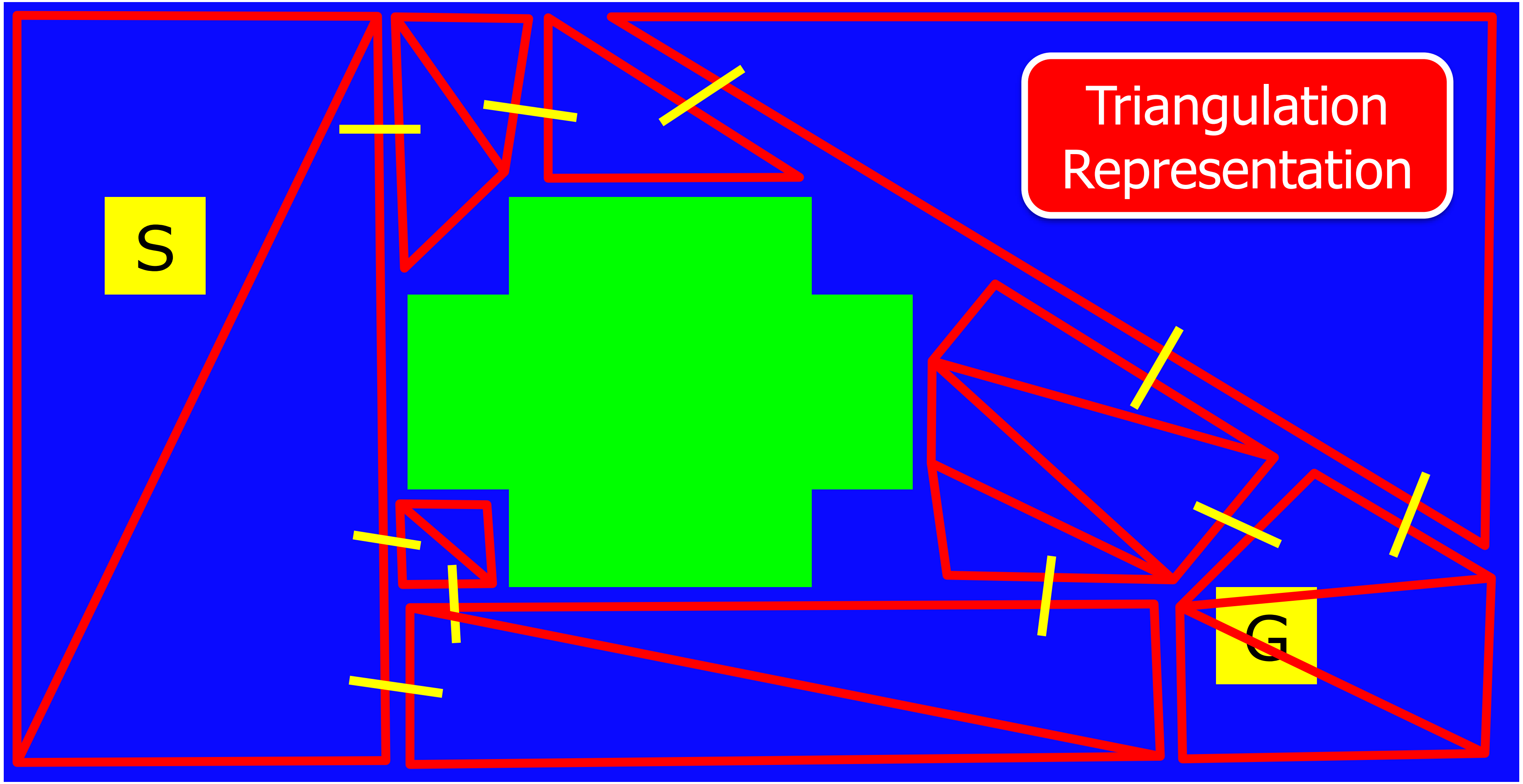
Graph / Node Representation

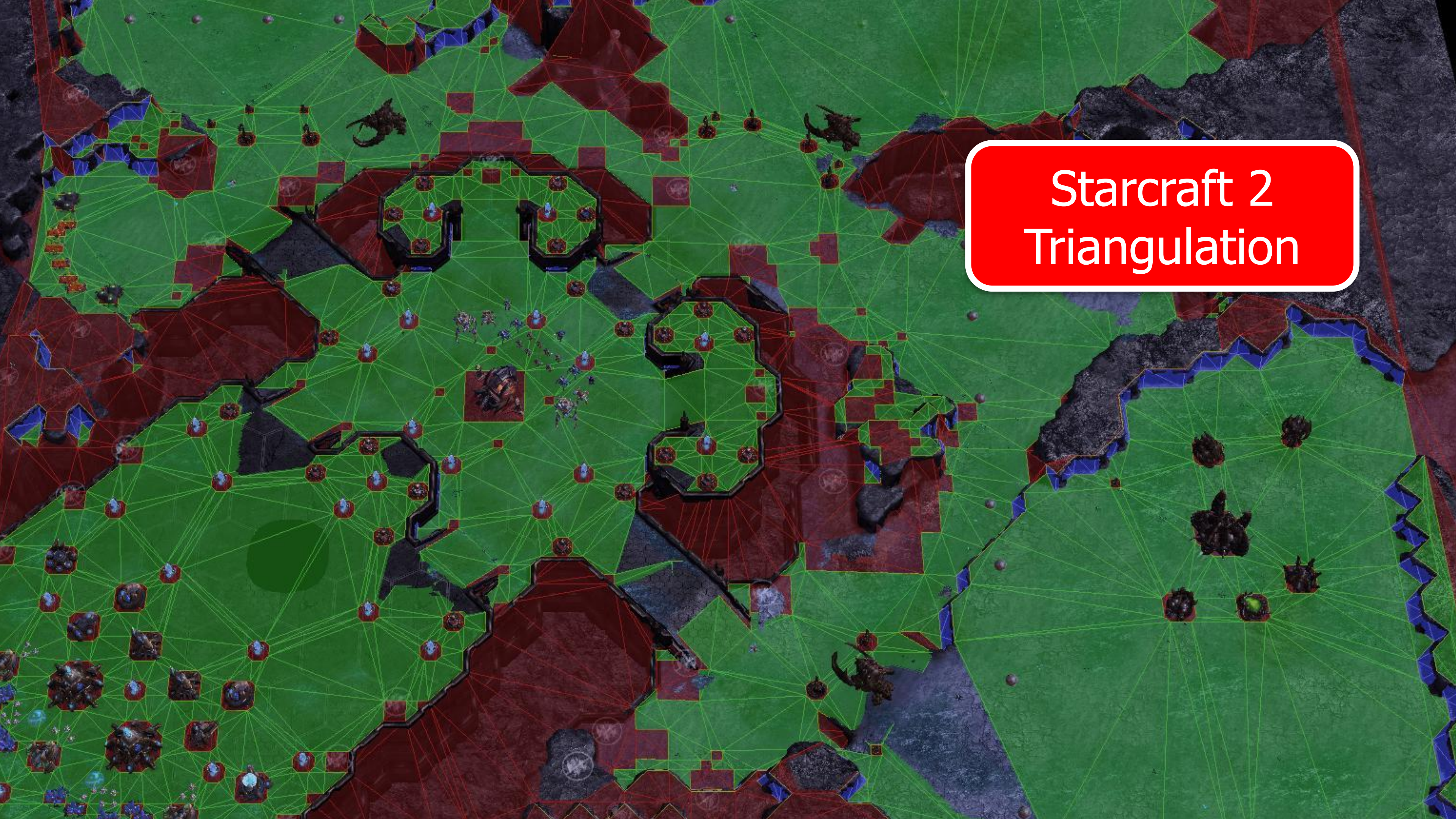


Convex Polygon Representation

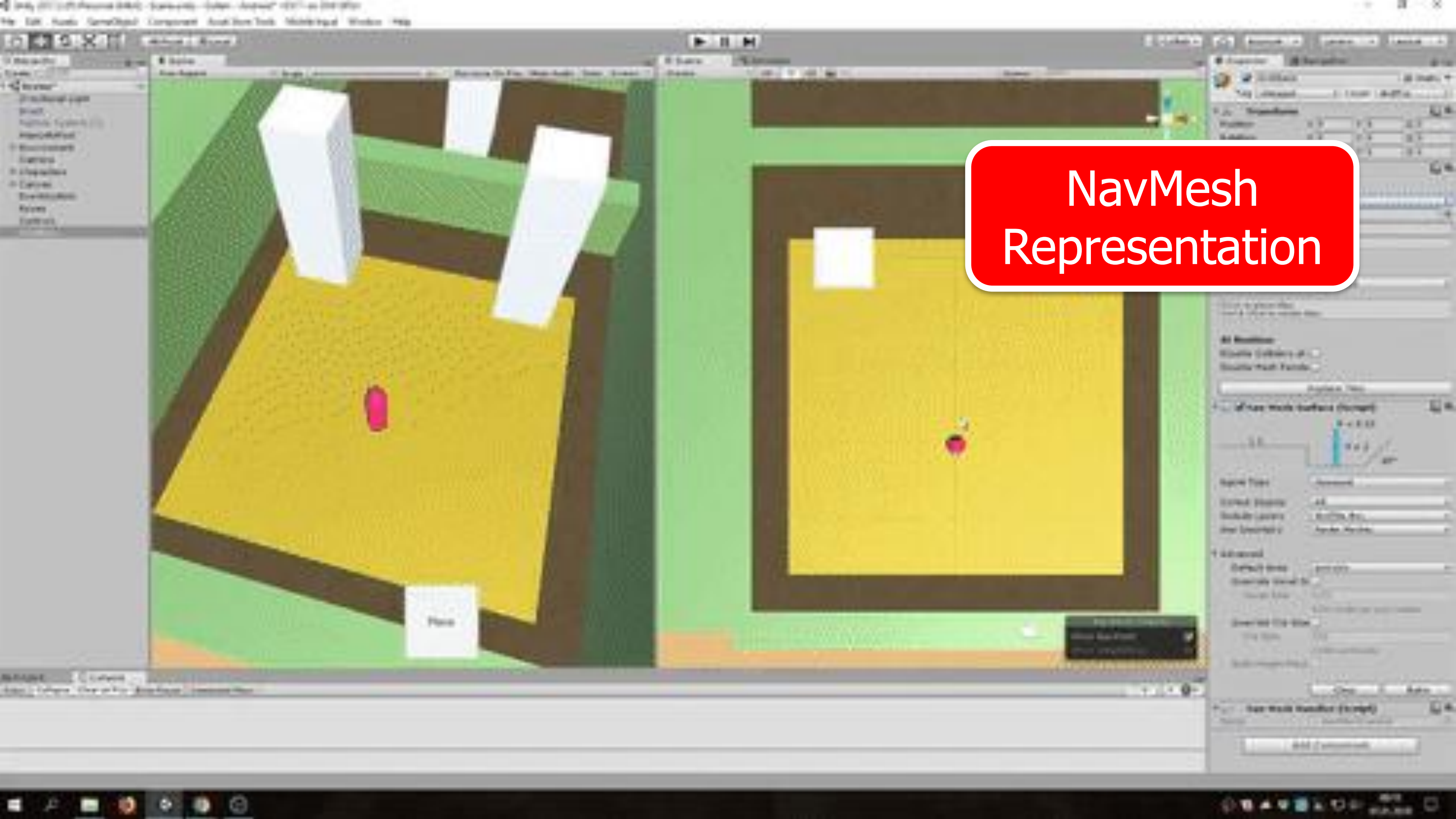


Triangulation Representation





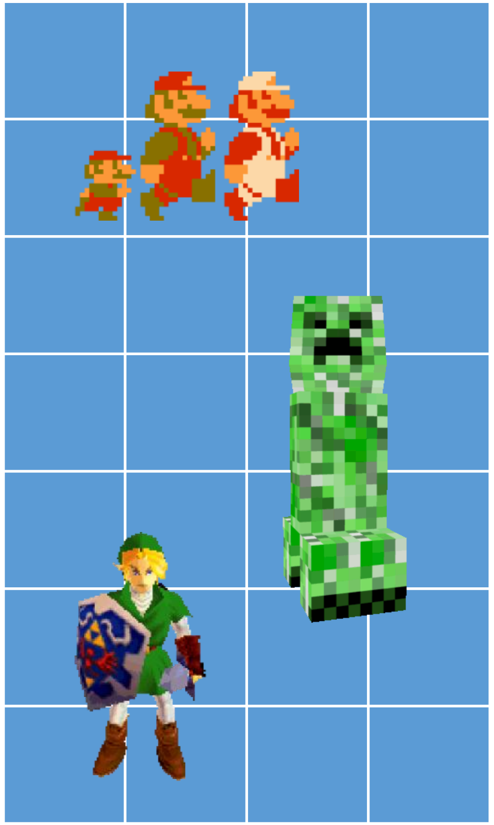
Starcraft 2 Triangulation



NavMesh Representation

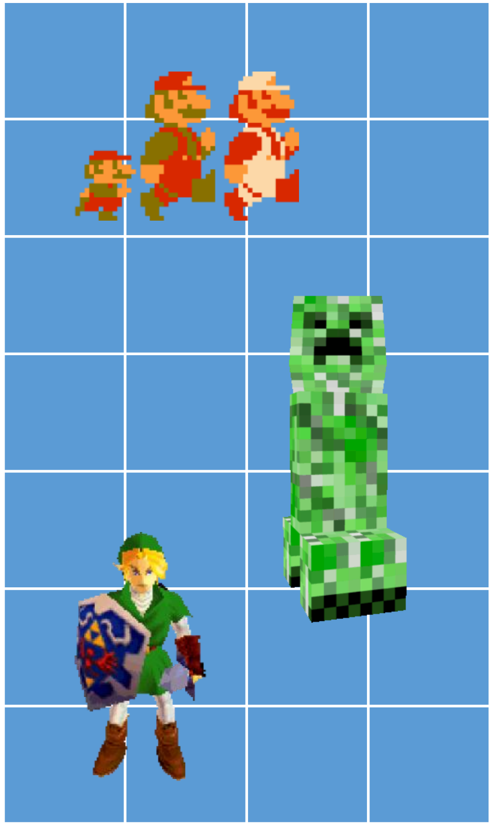
Grid Representations

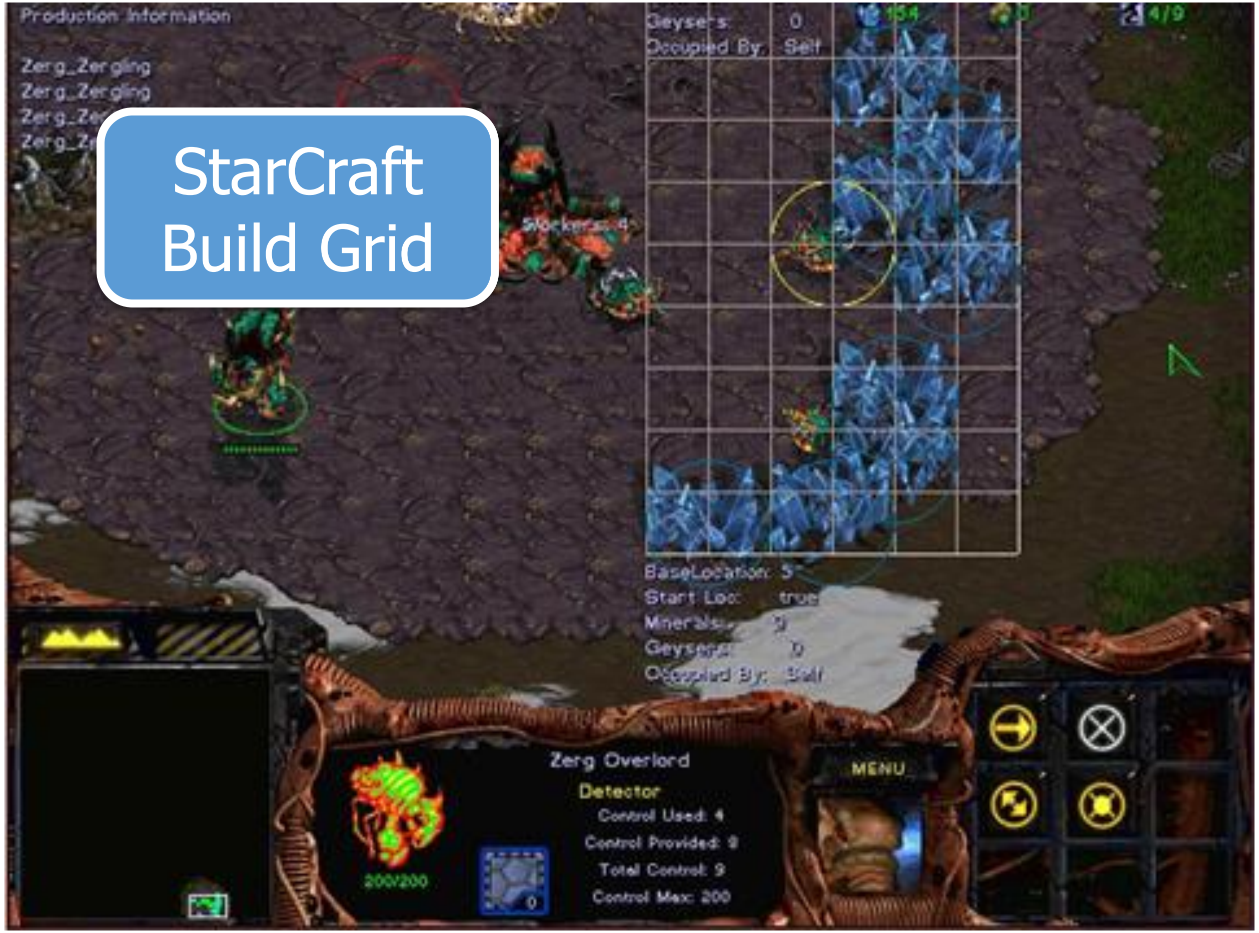
- 2D structure that divides the world into equally sized cells
- Simplest method to implement
- Used to represent 2D surfaces
- Used in many games
  - Warcraft / StarCraft
  - Dragon Age

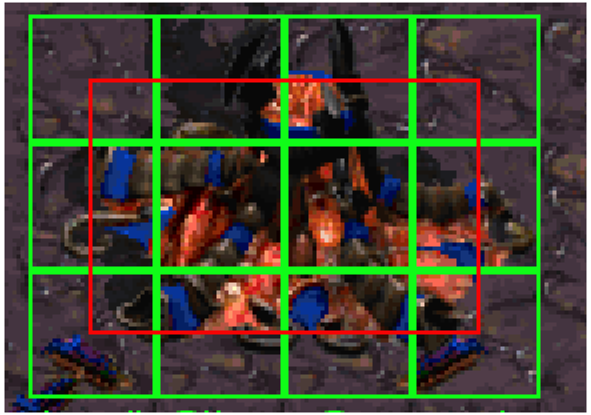


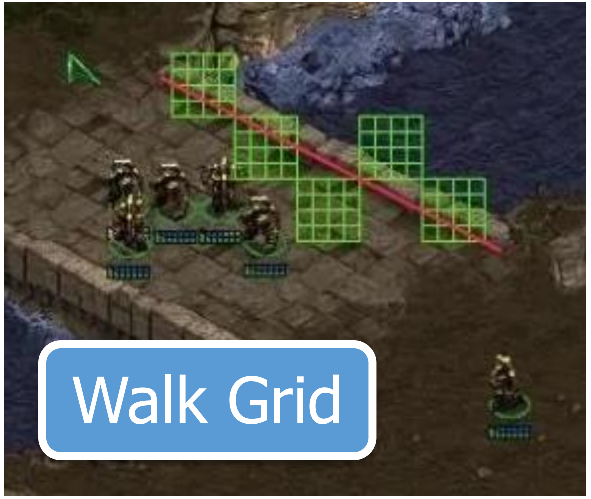
Grid Representations

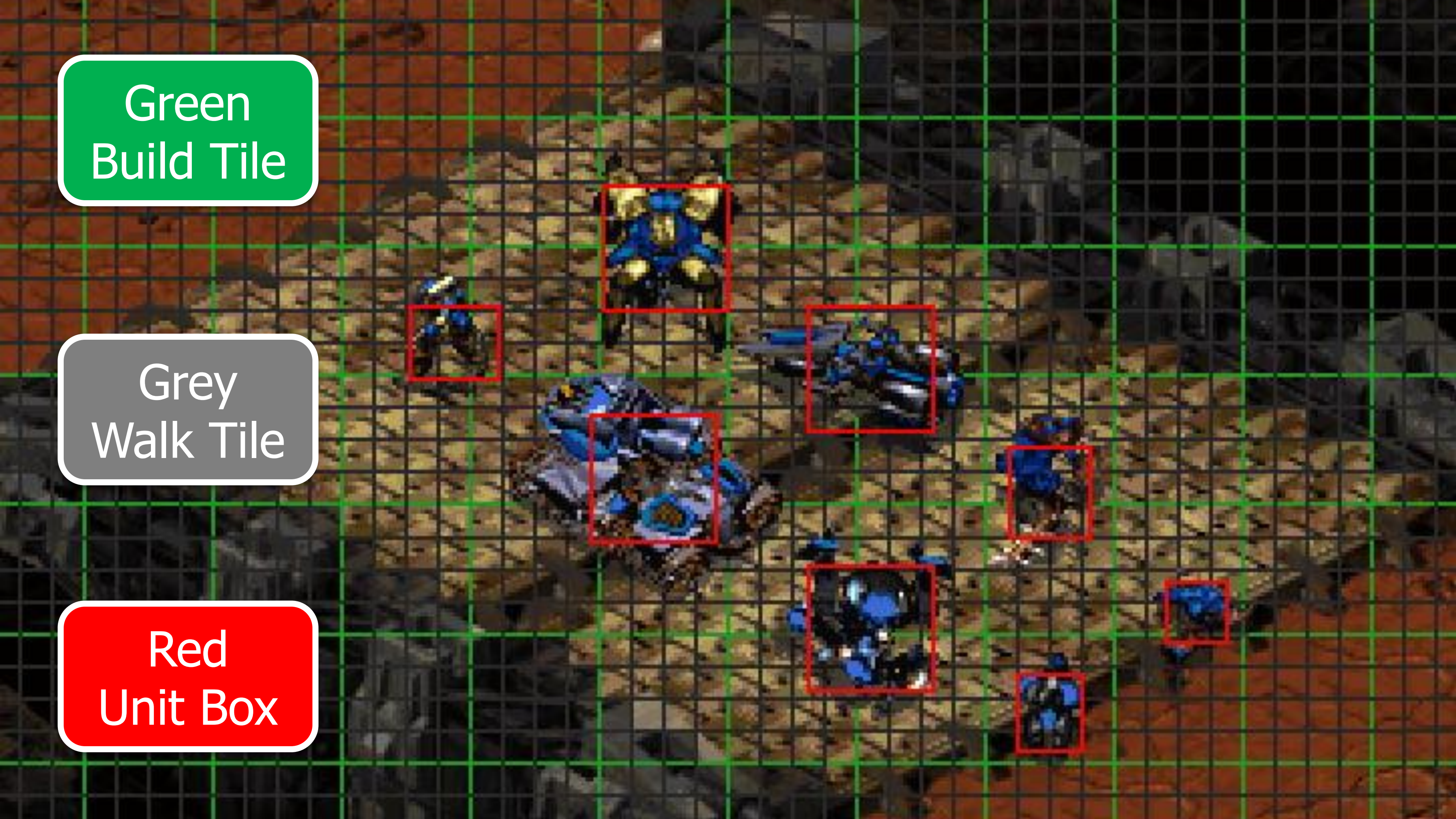
- Uses grids to abstract space so certain tasks are simpler
- Idea: Algorithms run on grid cells rather than game coords
- Can be implemented by 2D array / vector internally
- Used for many types of tasks




StarCraft Build Grid
Production Information
Zerg_Zergling
Zerg_Zergling
Geysers: 0
Occupied By: Self
BaseLocation: 5
Start Loc: true
Minerals: 9
Geysers: 0
Occupied By: Self
Zerg Overlord
Detector
Control Used: 4
Control Provided: 8
Total Control: 9
Control Max: 200
200/200
MENU

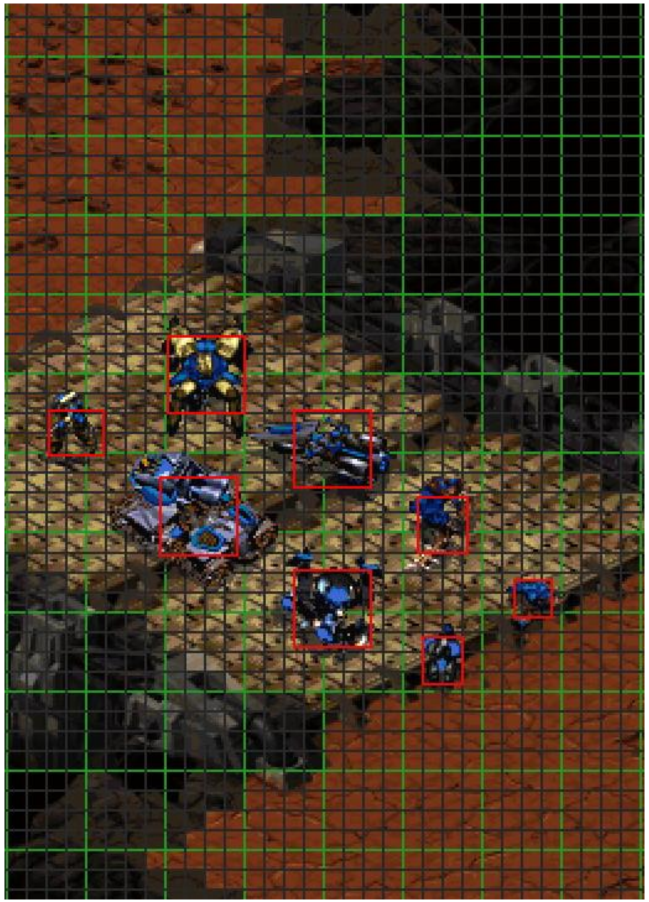


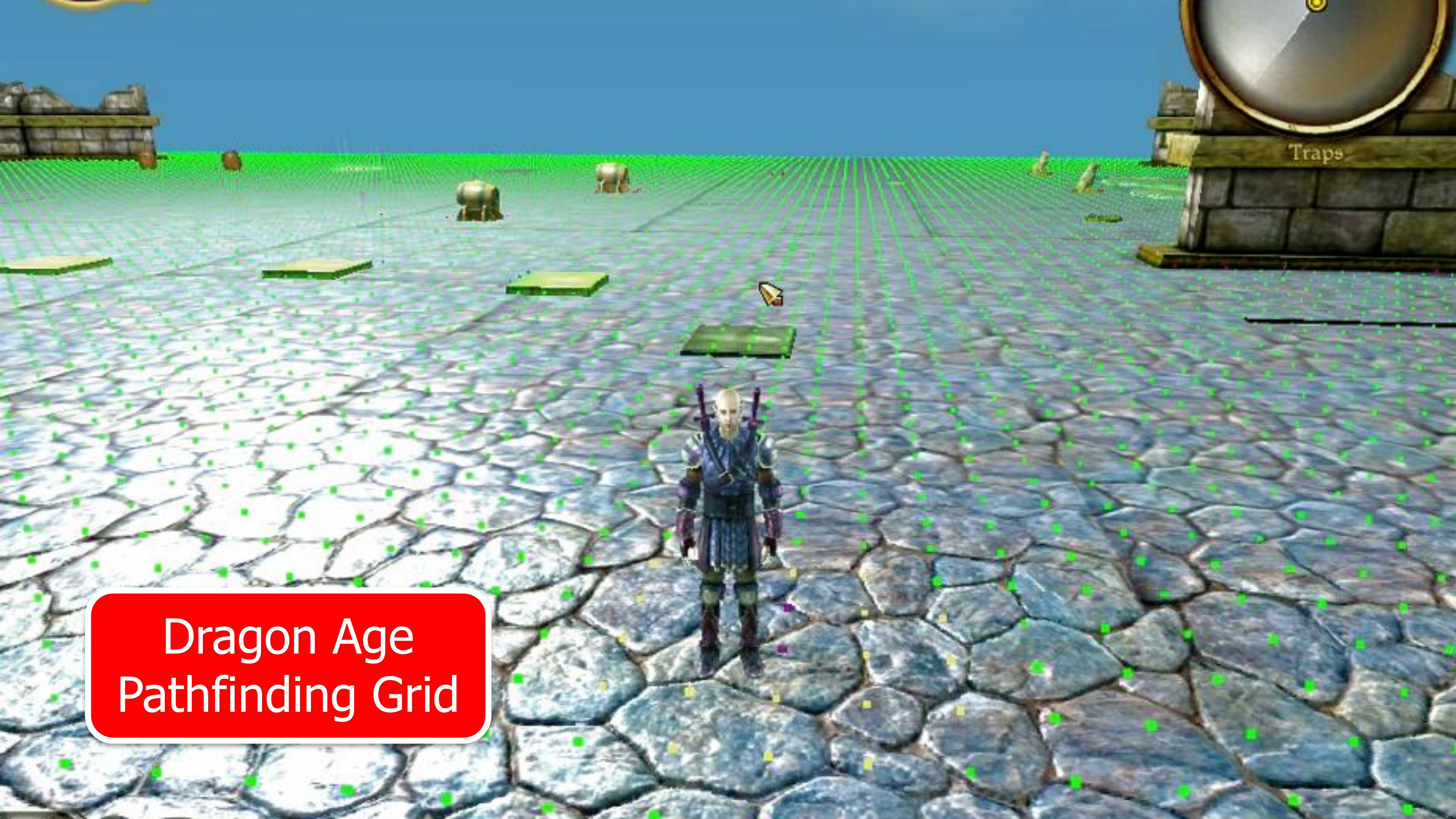

Walk Grid


Green Build Tile
Grey Walk Tile
Red Unit Box

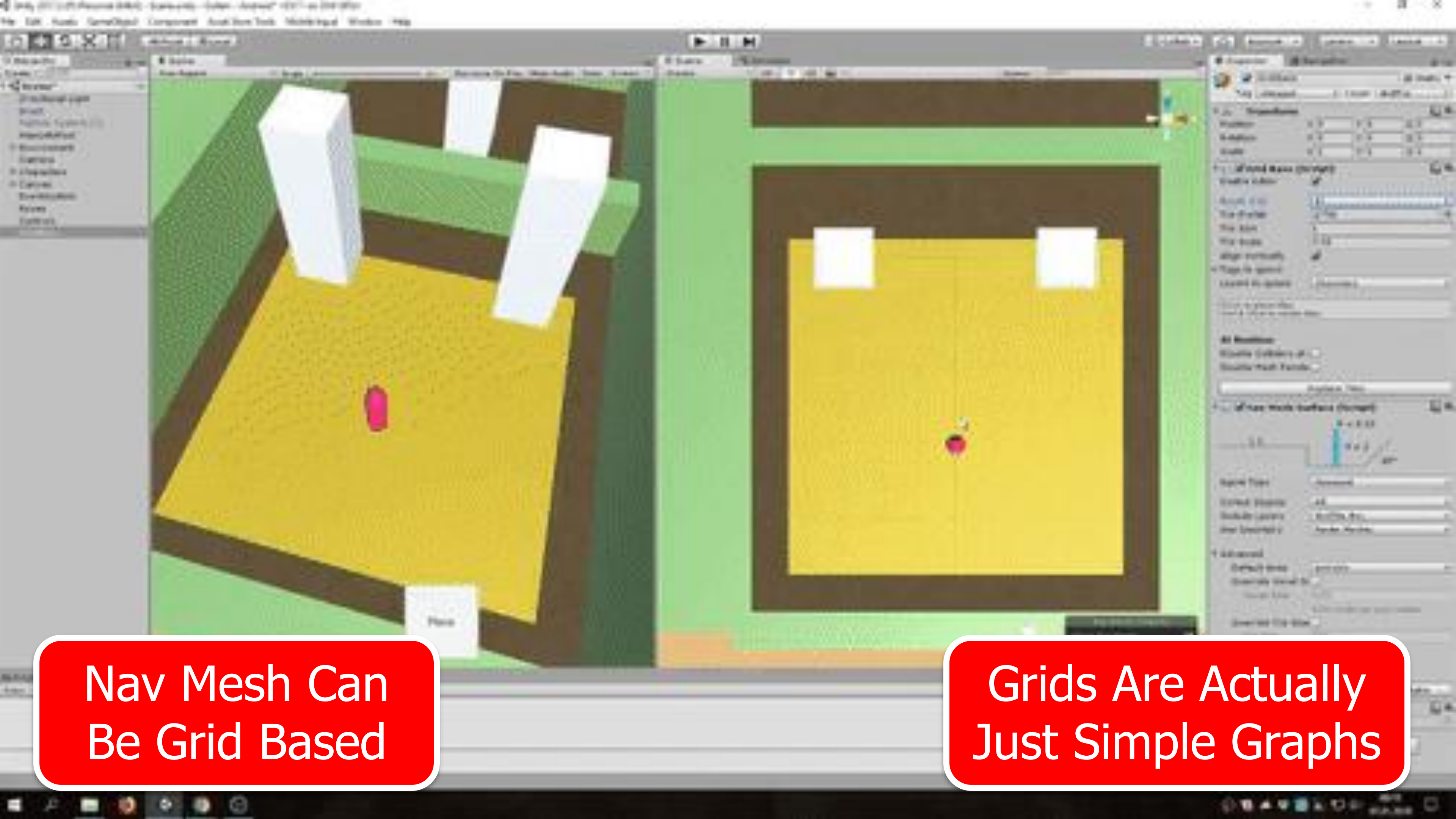
StarCraft Grid

- 3 grids of different precision
- Pixel Level (1x1 pixel)
  - Units move in pixel increments
- Walk Tile (8x8 pixels)
  - Map 'walkability' Boolean grid
  - Units can't overlap 'false' tiles
- Build Tile (32x32 pixels)
  - Building placed on w*h rectangle
  - Can't place on unwalkable tile





Dragon Age Pathfinding Grid

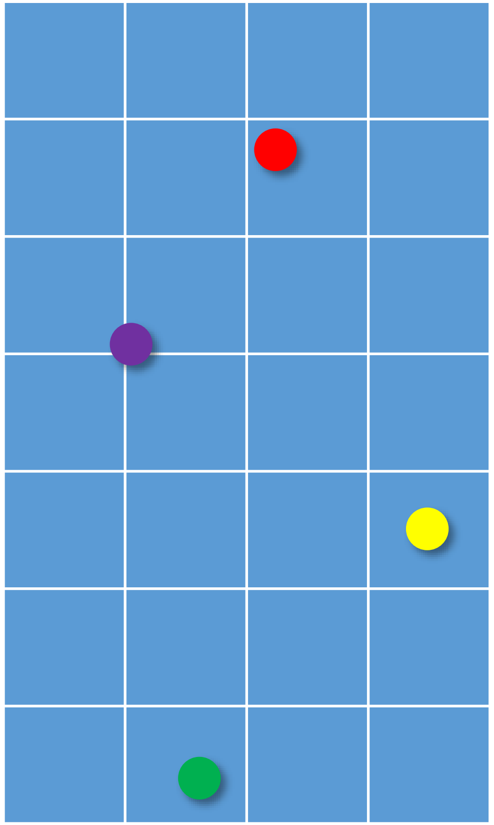


Nav Mesh Can Be Grid Based

Grids Are Actually Just Simple Graphs

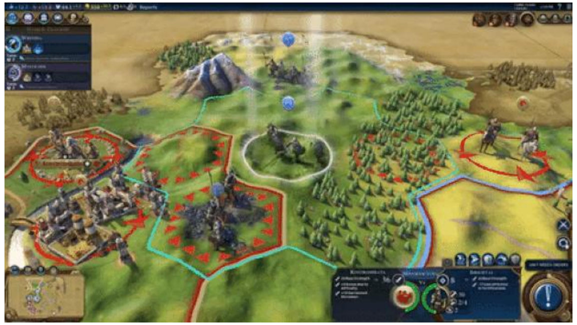
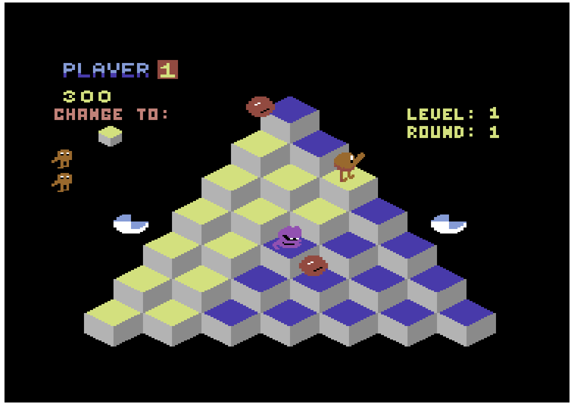
Grids: Localization

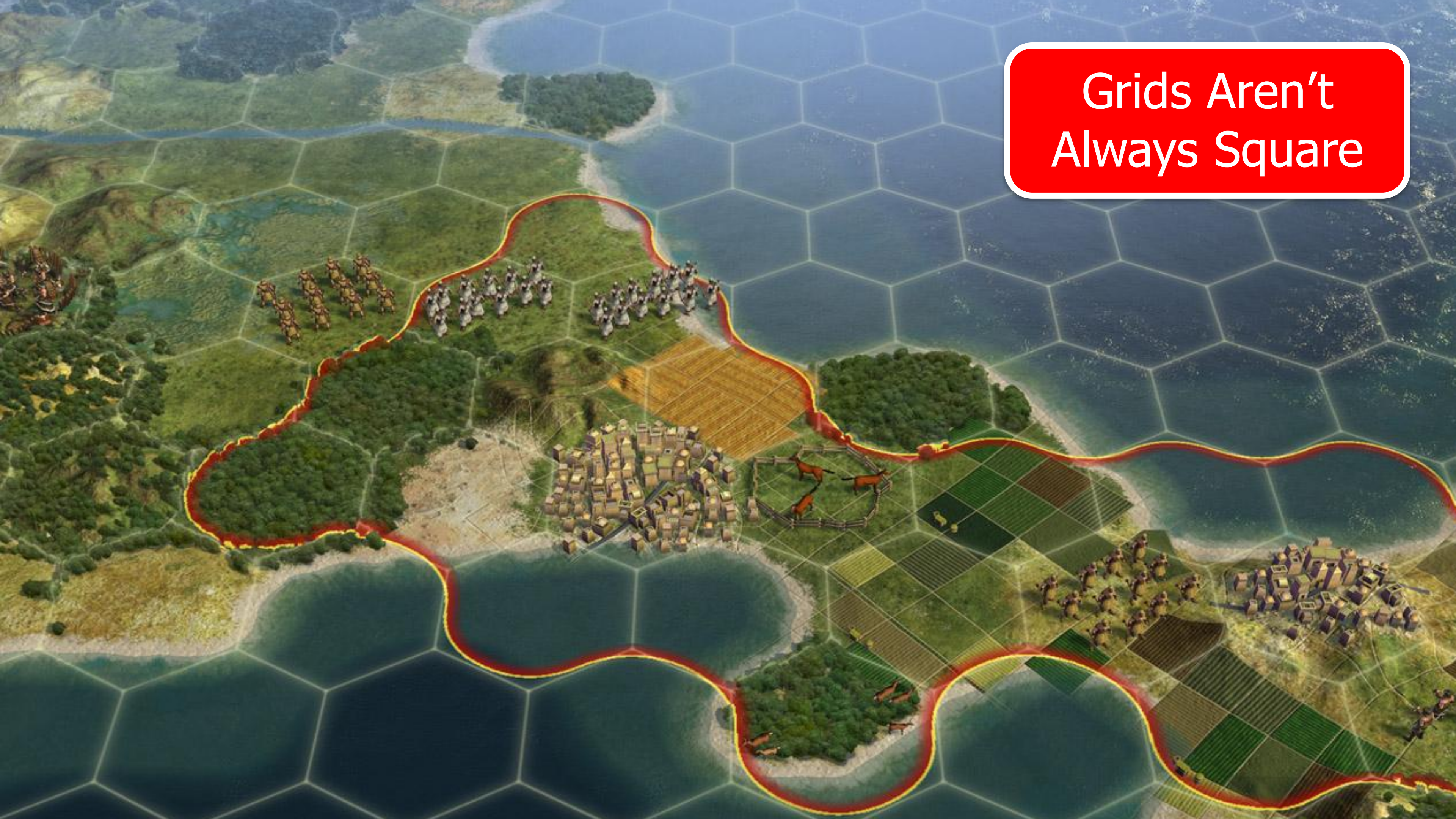
- Entities generally still live in game real-valued coordinate systems
- Must translate from game world position to cell position and back
- Ways to calculate cell from position
  - Truncate / round position
  - Divide position by grid size
- Ex: Pos(23.6, 44) Grid Size(10,10)
  - Cell(23.6/10, 44/10) = Cell(2,4)



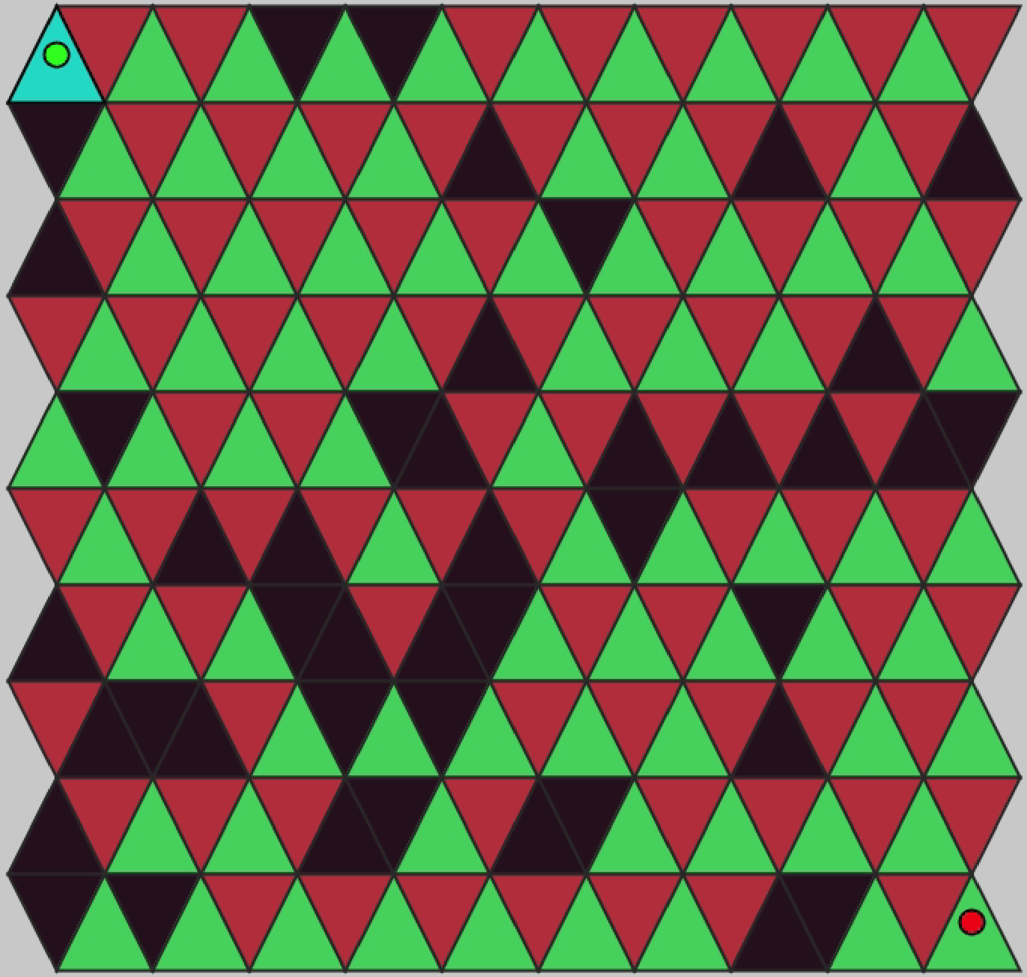
Grid World Games

- Some games take place completely within a grid
- Grids not always square
- These 'grid world' games need no translation
- Game space and actions are defined by the grid



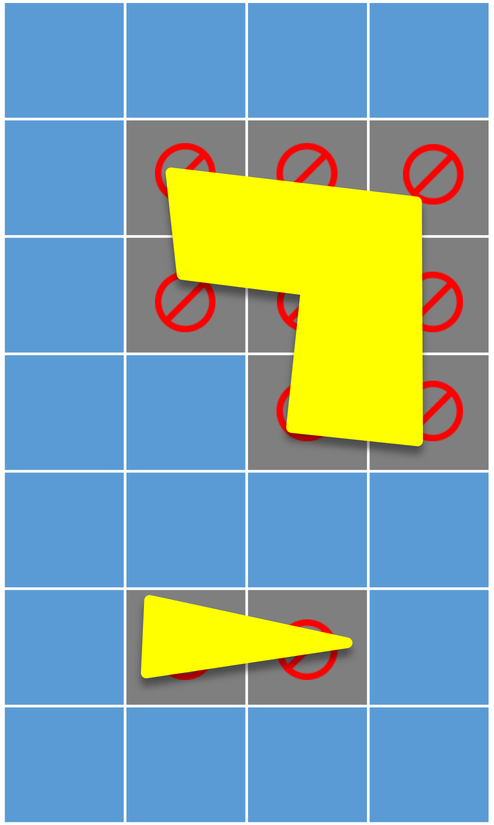
Grids Aren't Always Square



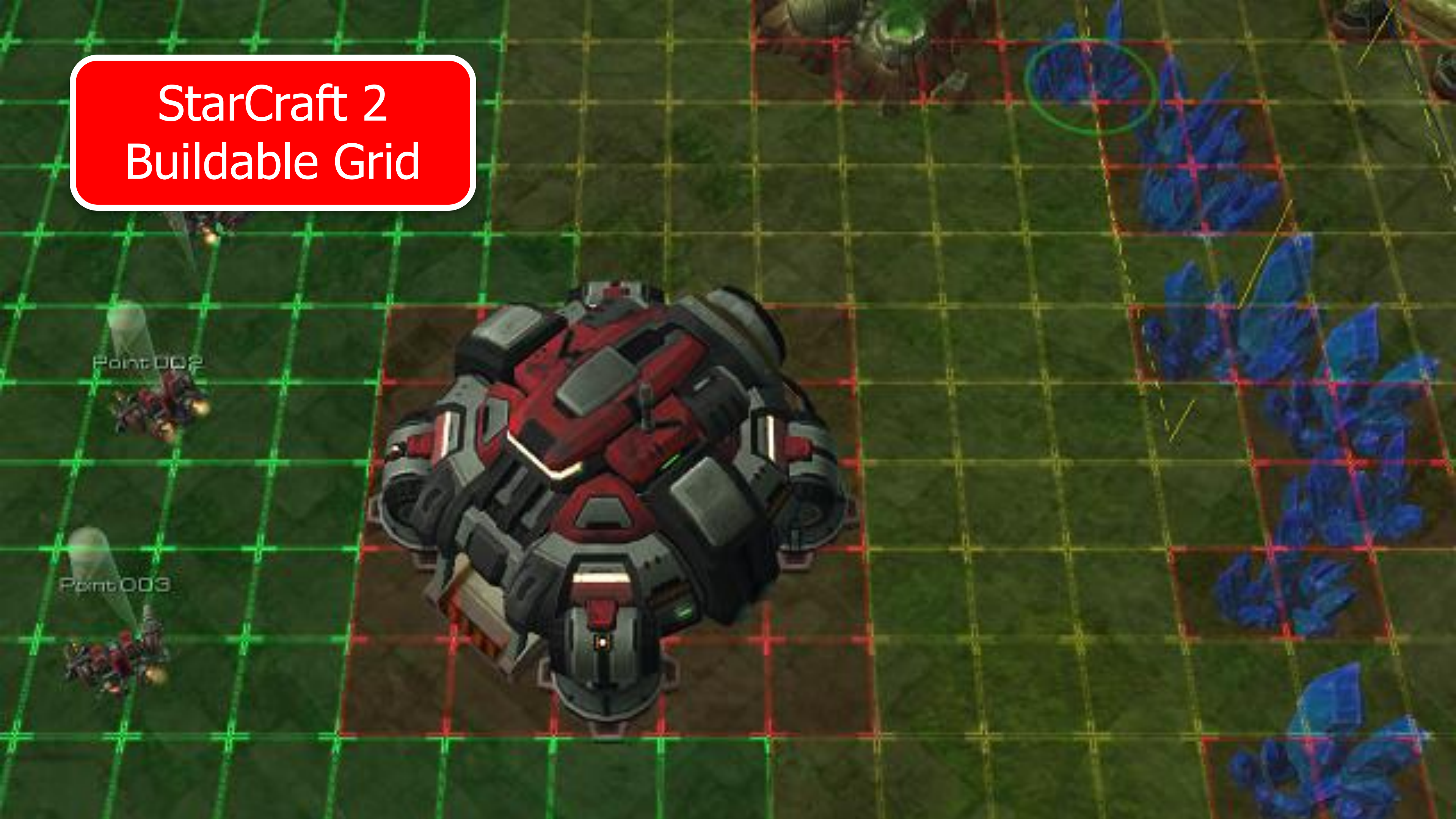
Grids Aren’t Always Square

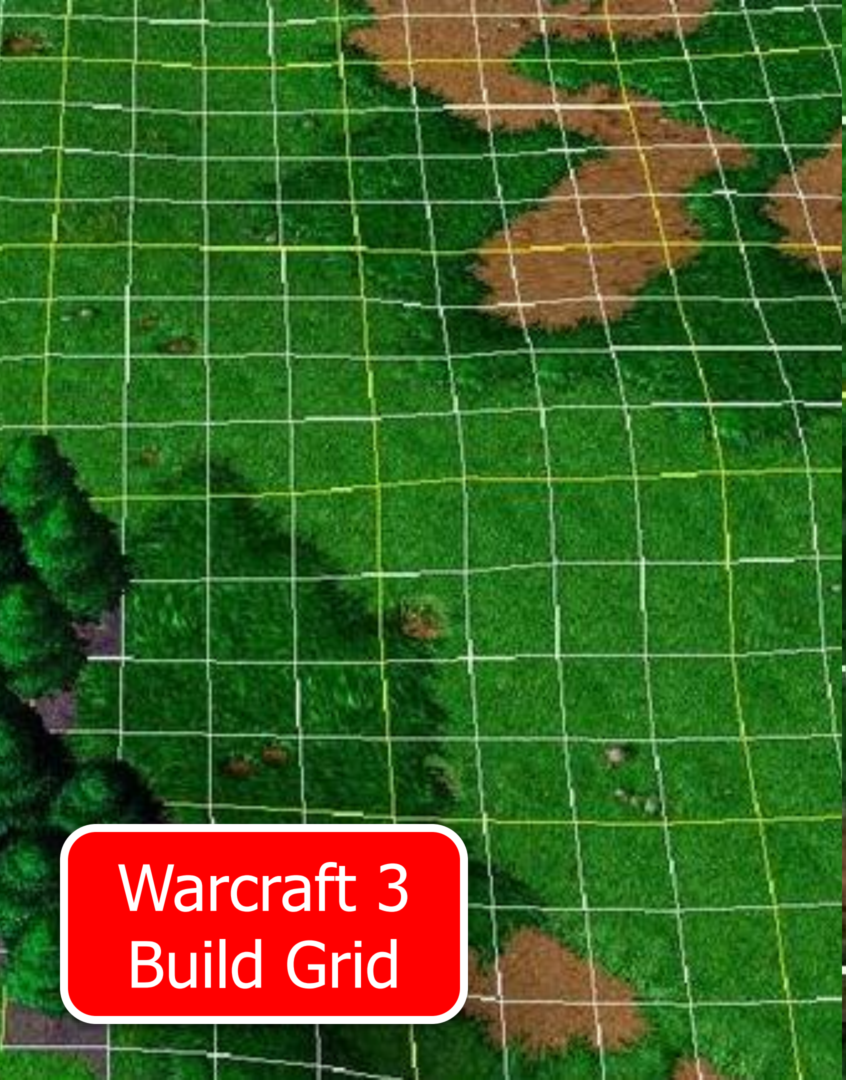
Task Location Grid

- Used to determine on which tiles we can perform a desired task (walk, build, etc)
- Initially, grid set entirely to true
- Loop over each object in the grid which blocks a given task from being completed, and set the underlying cells to false
- Grid can now be used as a constant-time lookup table to check if task is possible
- Note: In some games we may be trading accuracy for speed of computation

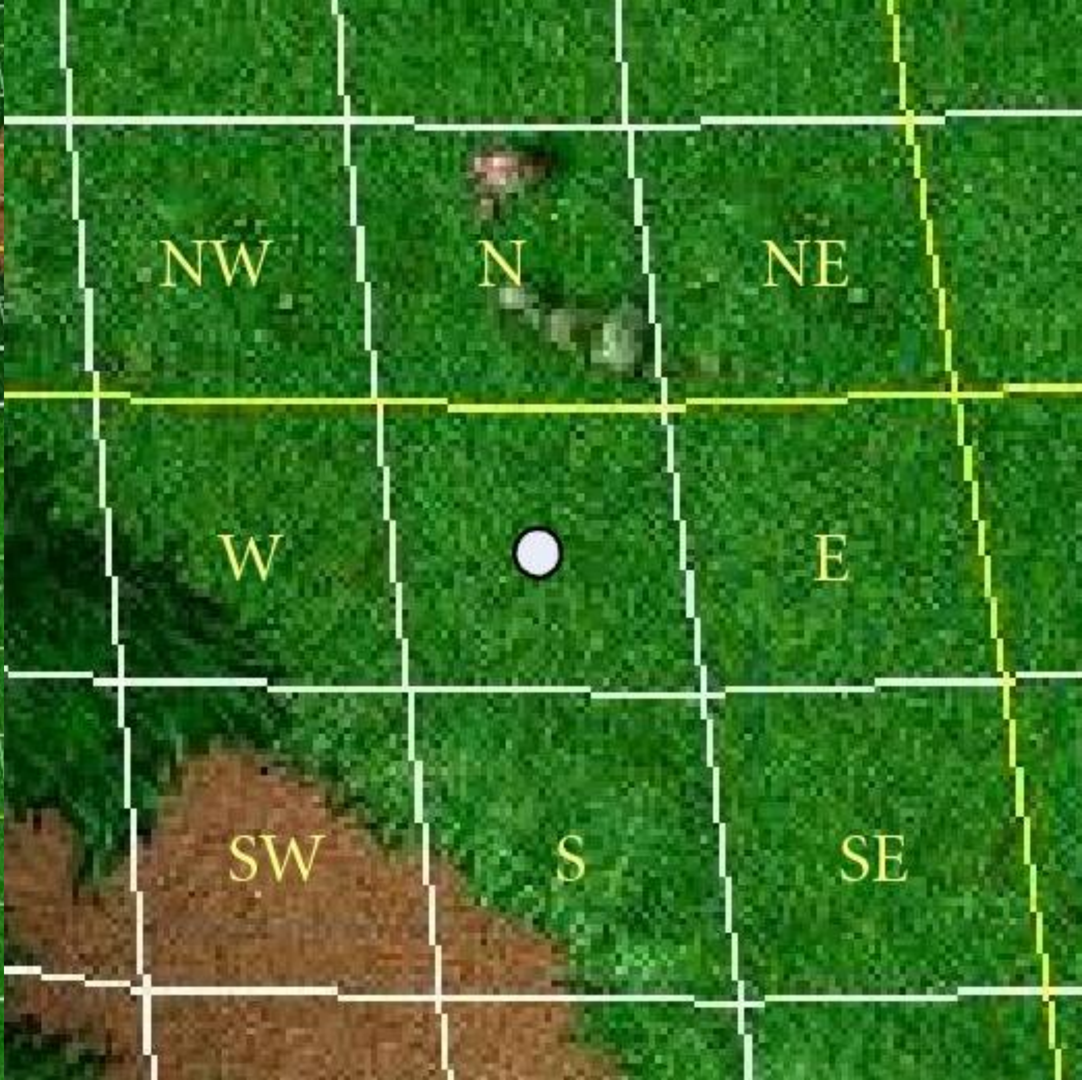


StarCraft 2 Buildable Grid

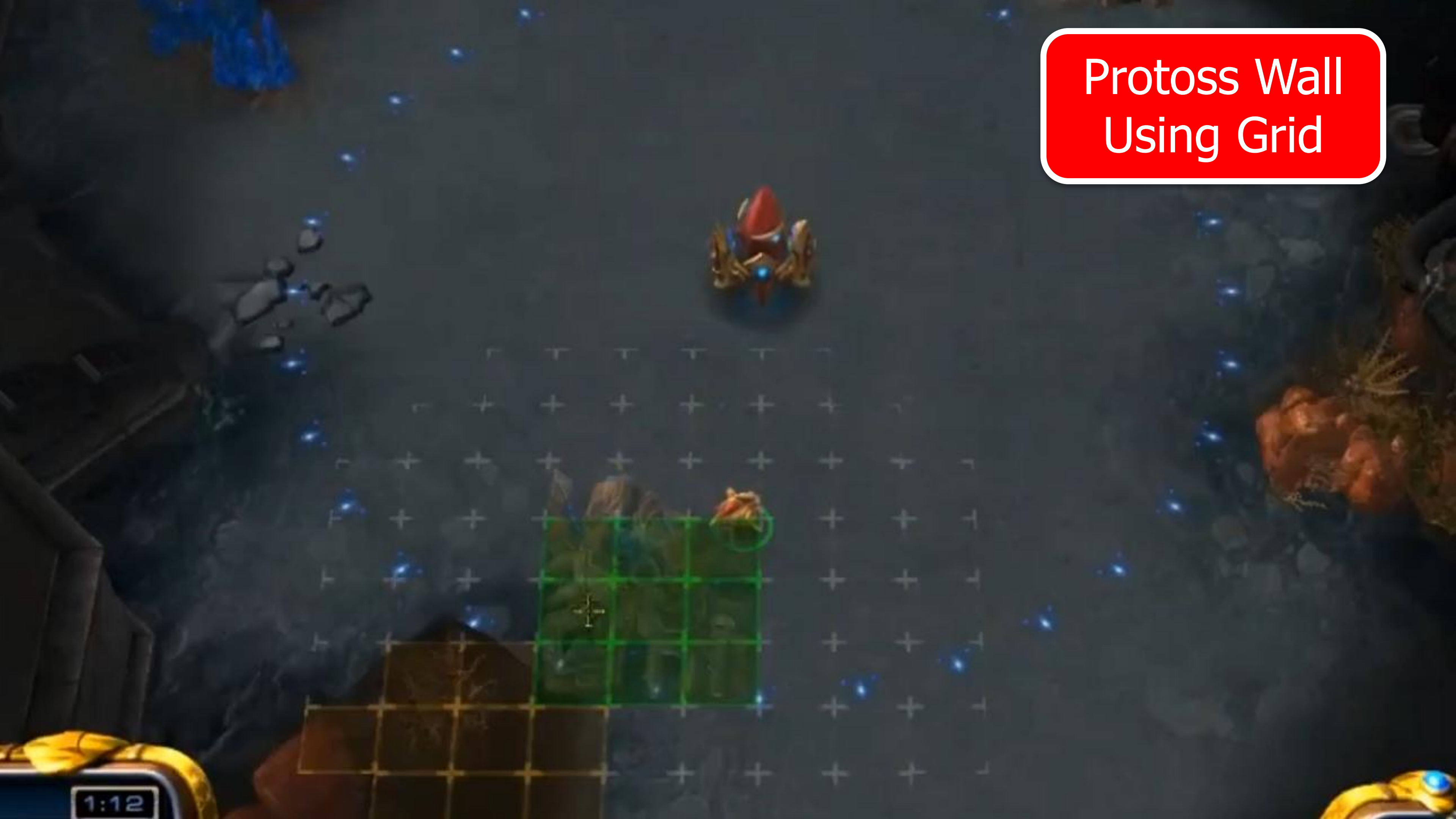




Warcraft 3 Build Grid

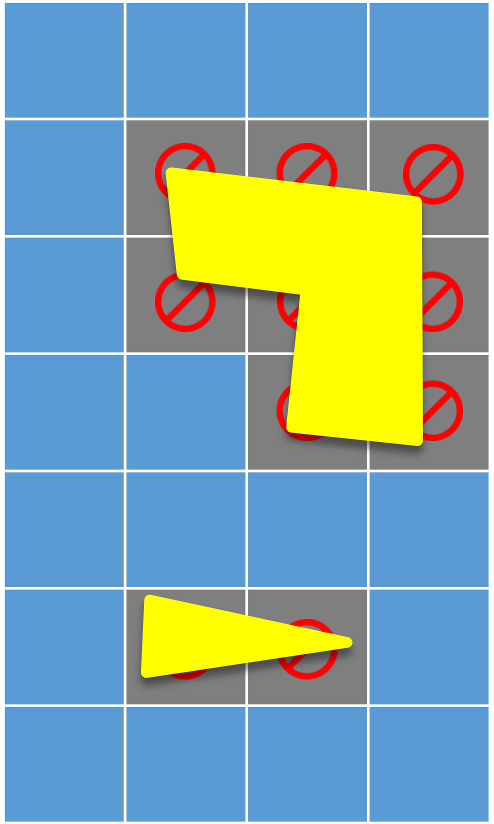


Protoss Wall Using Grid

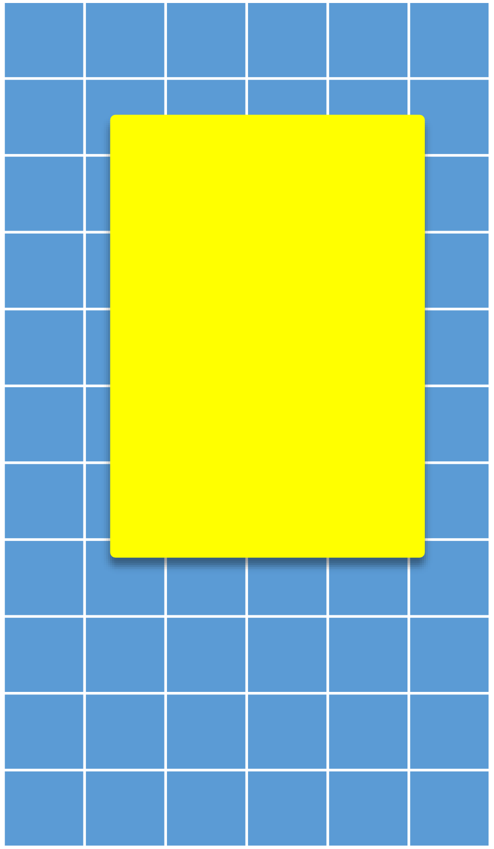


Task Location Grid

- In general, filling the grid may involve computing polygon edge line segment intersections with grid lines
- In the case of axis-aligned rectangles we can optimize this computation to be much faster

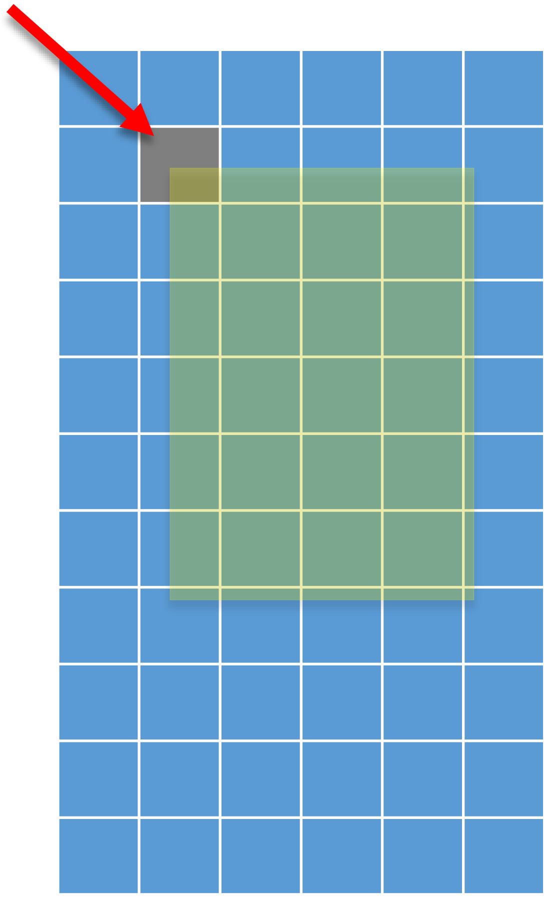


Rectangle Optimization



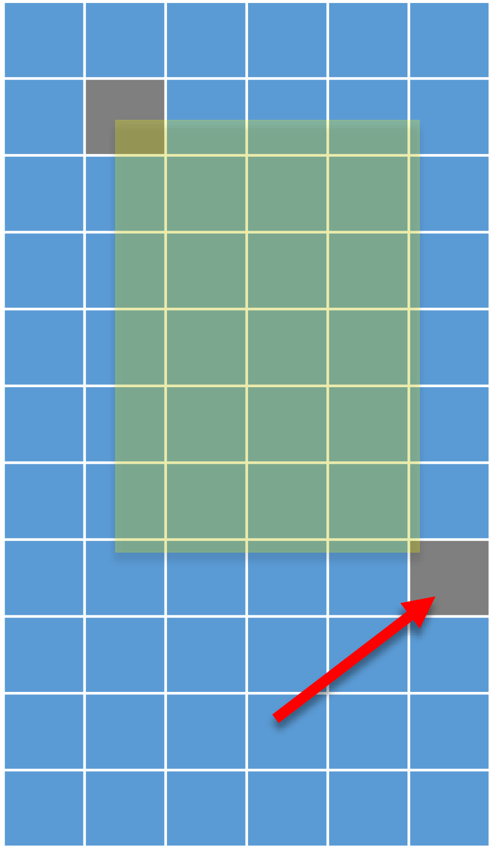
Rectangle Optimization

1. Compute cell of rect top-left
   `ctl = (tl.x / gridSize, tl.y/gridSize)`



Rectangle Optimization

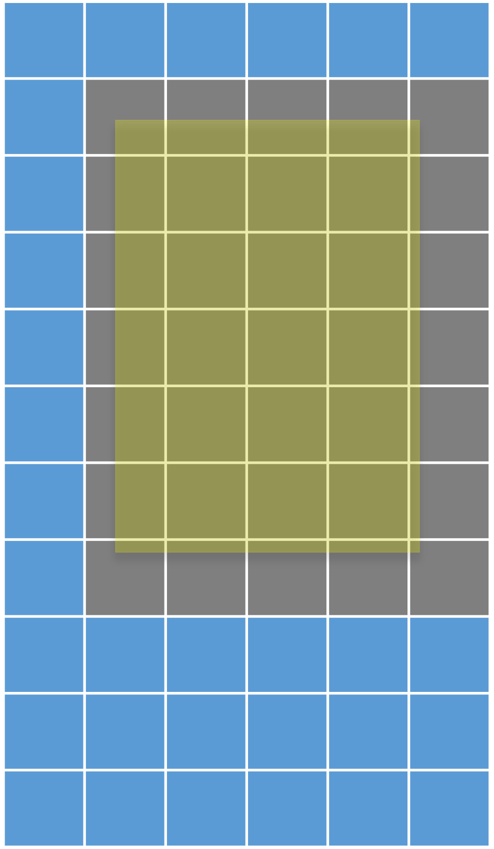
1. Compute cell of rect top-left
   `ctl = (tl.x / gridSize, tl.y/gridSize)`
2. Compute cell of rect bottom-right
   `cbr = (br.x / gridSize, br.y/gridSize)`



Rectangle Optimization

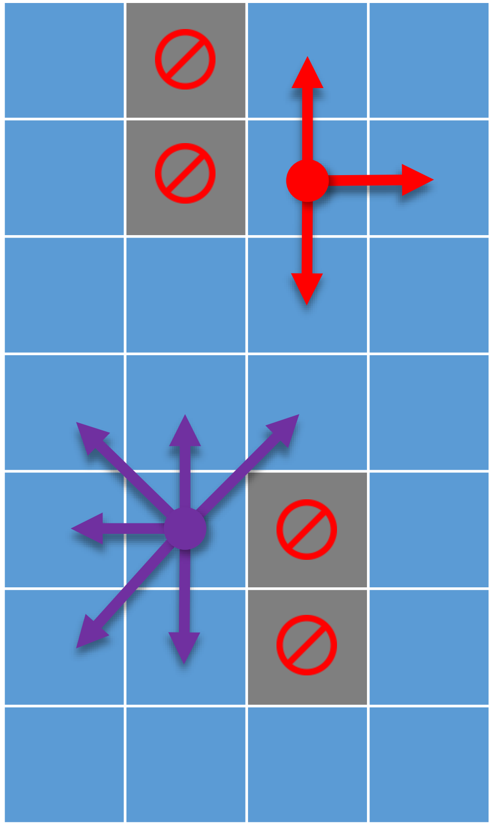
1. Compute cell of rect top-left
   `ctl = (tl.x / gridSize, tl.y/gridSize)`
2. Compute cell of rect bottom-right
   `cbr = (br.x / gridSize, br.y/gridSize)`
3. Loop from top left to bottom right and mark each cell as blocked

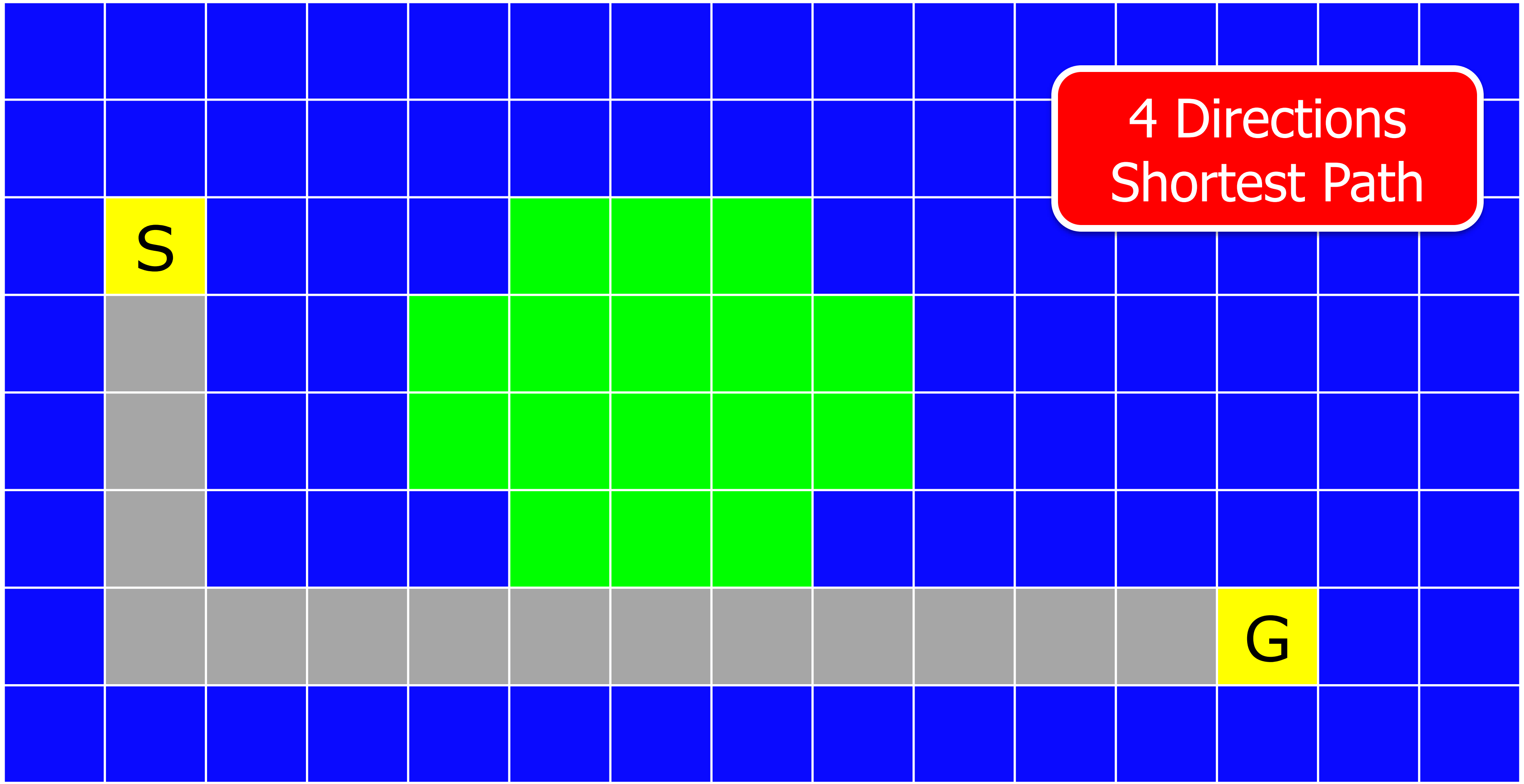
No line intersections necessary!

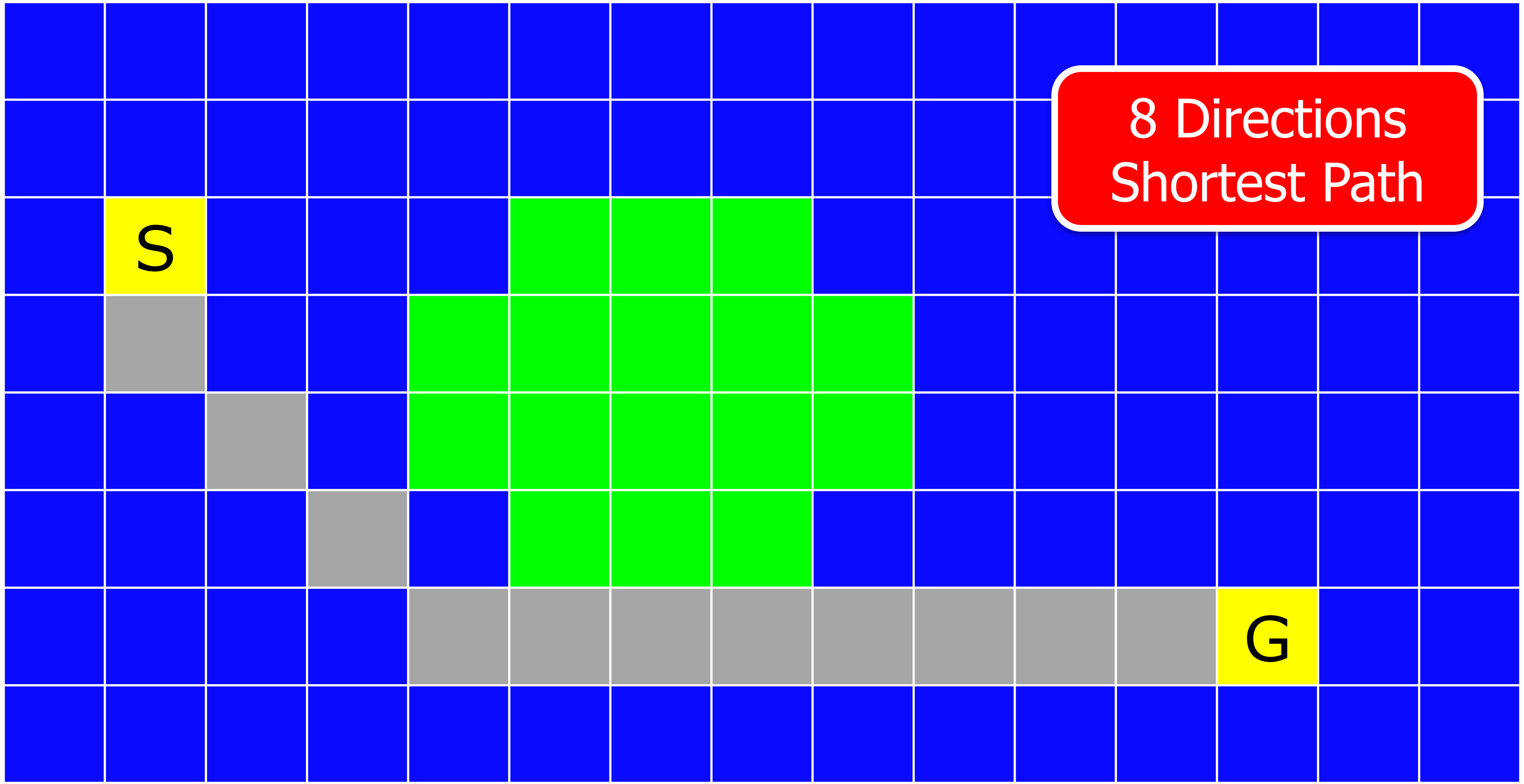


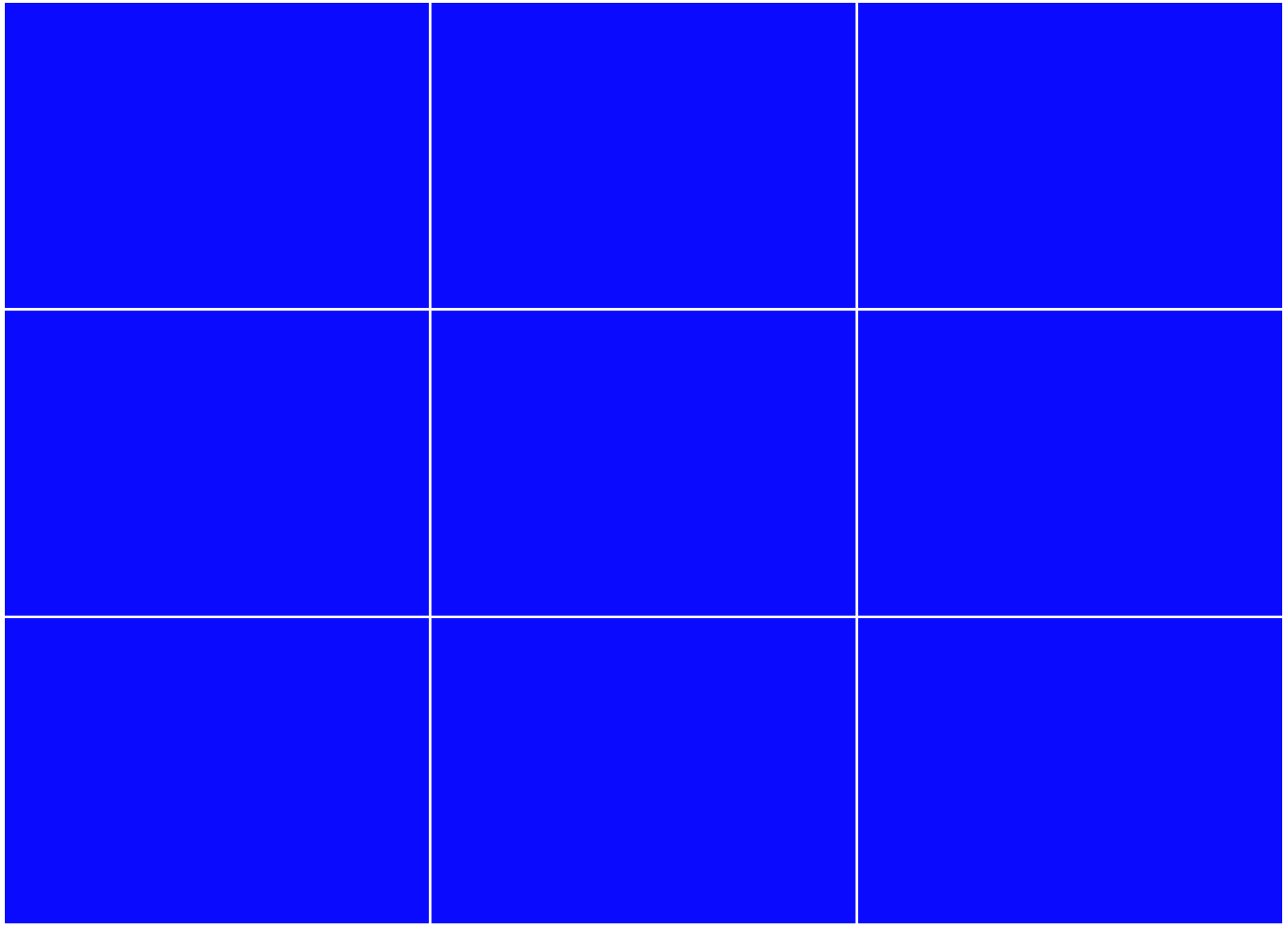
Grid Movement

- Grid movement is restricted to adjacent traversable cells
- Exceptions for games with teleporting, etc
- Legal Grid Actions
  - Cardinal (UDLR)
  - Octile (UDLR + Diagonal)




4 Directions
Shortest Path
S
G


8 Directions
Shortest Path
S
G

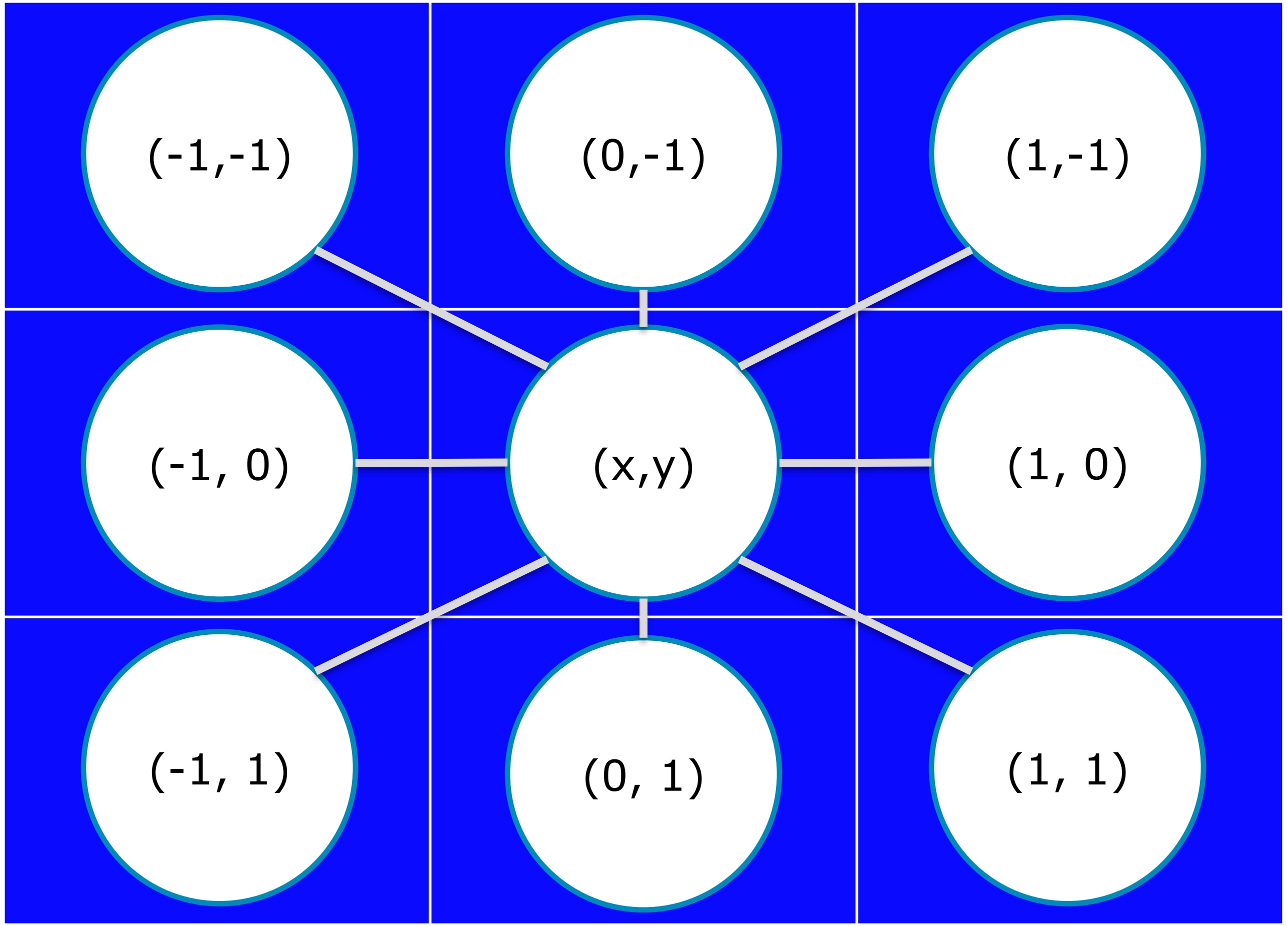


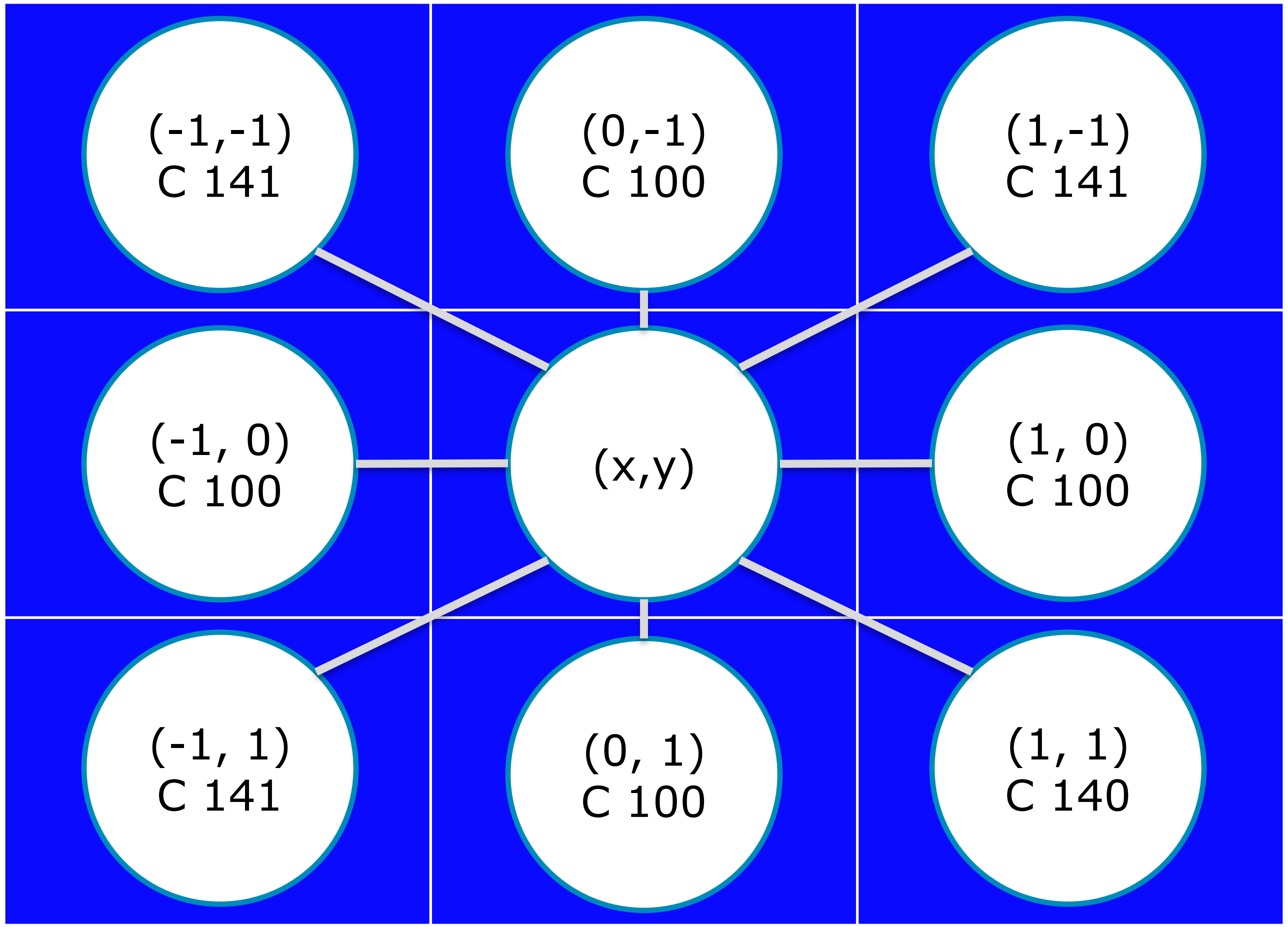
Grid Cells

Grid Cell Positions

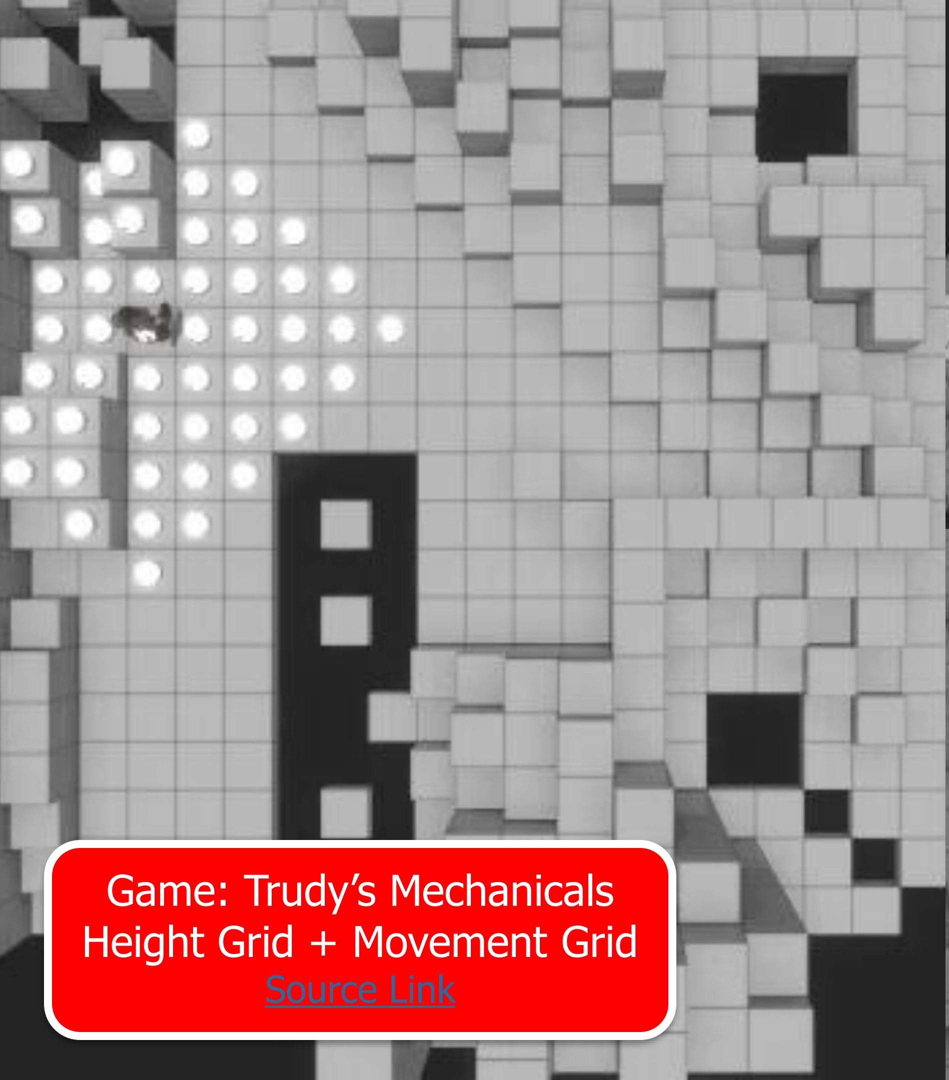
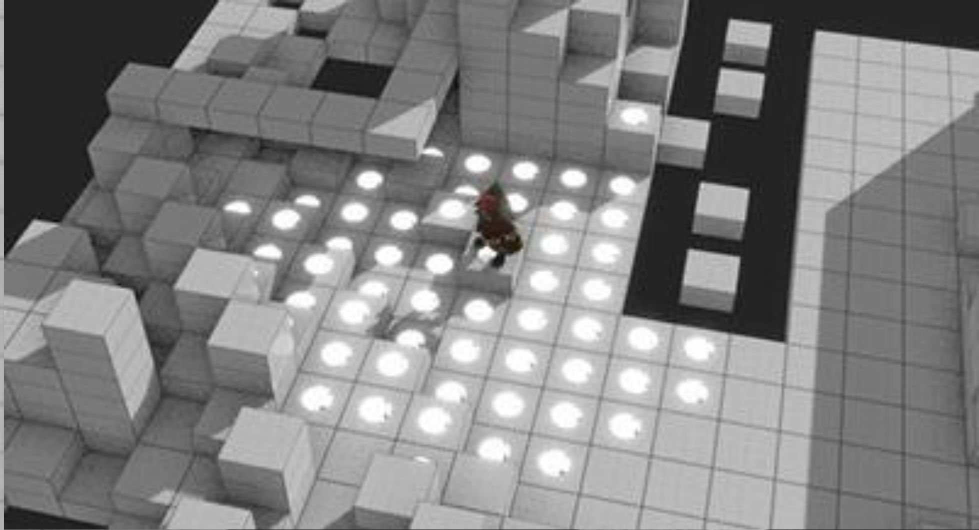
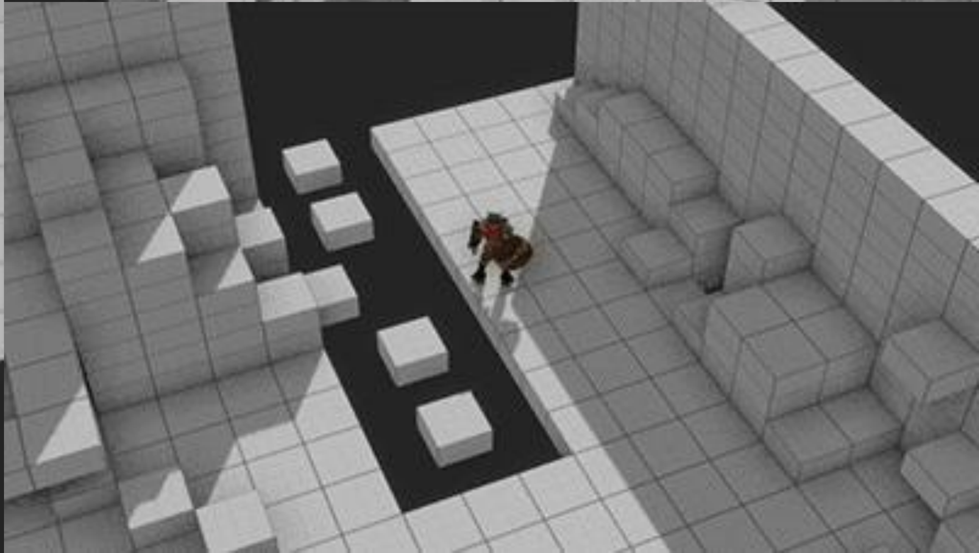
0,0	1,0	2,0
0,1	1,1	2,1
0,2	1,2	2,2

Grid Actions





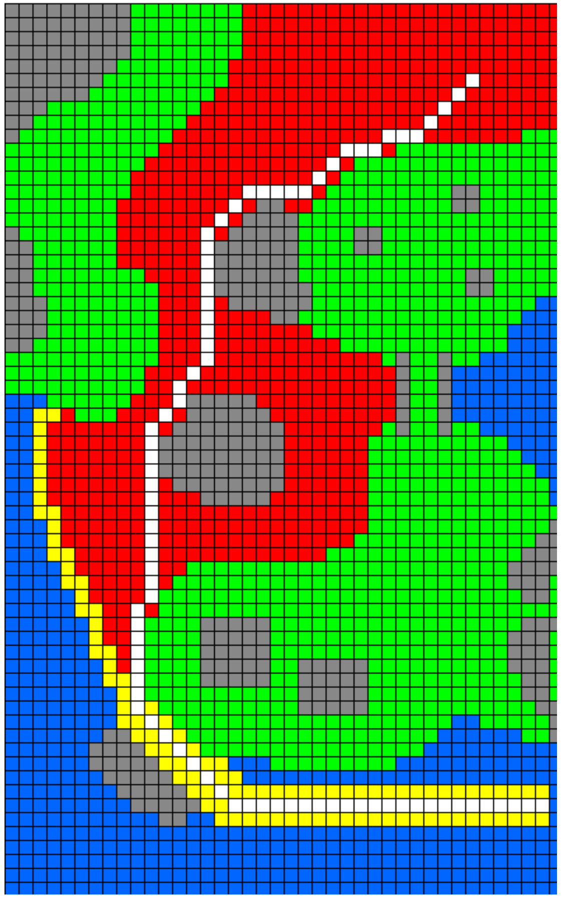
Action Cost C

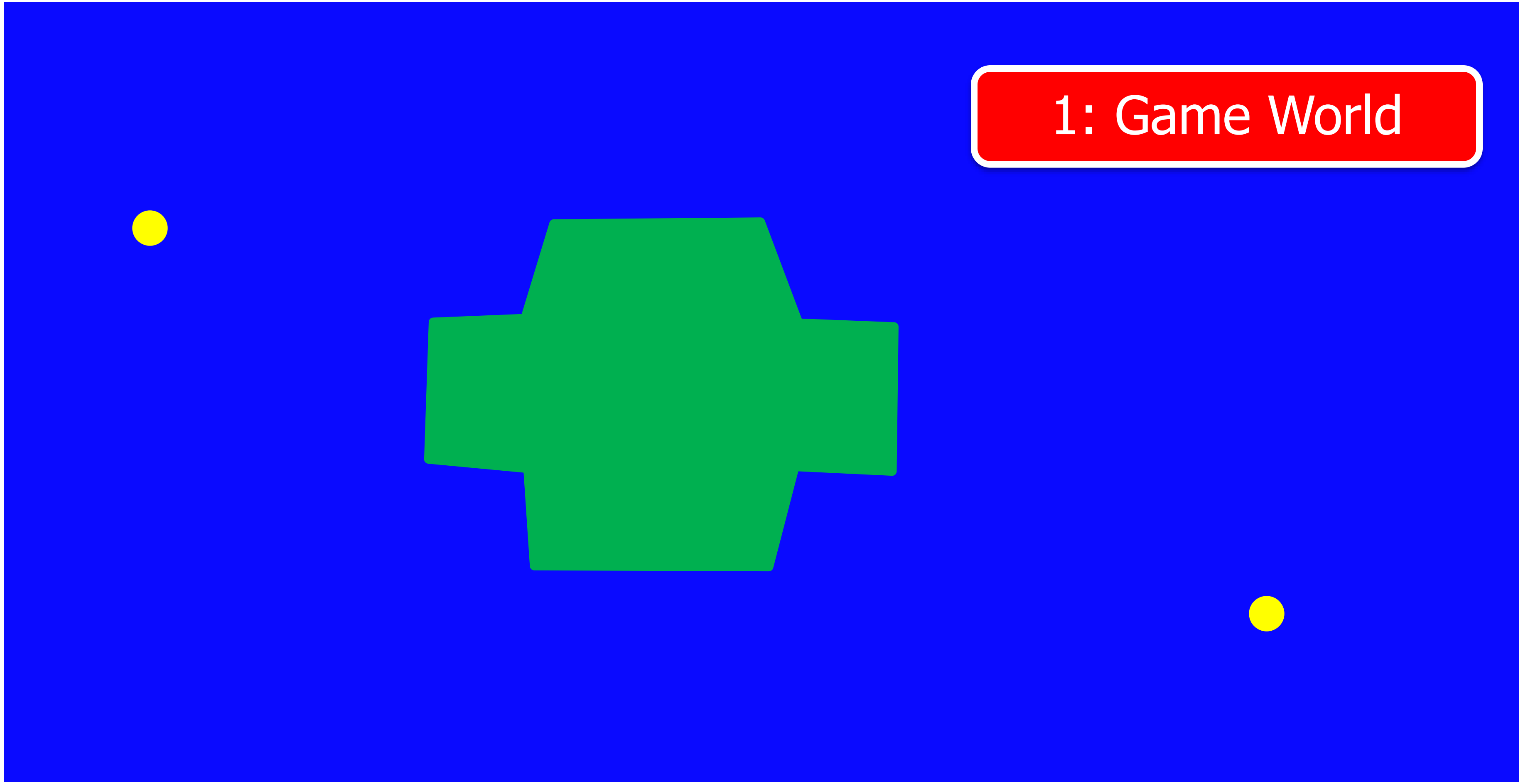




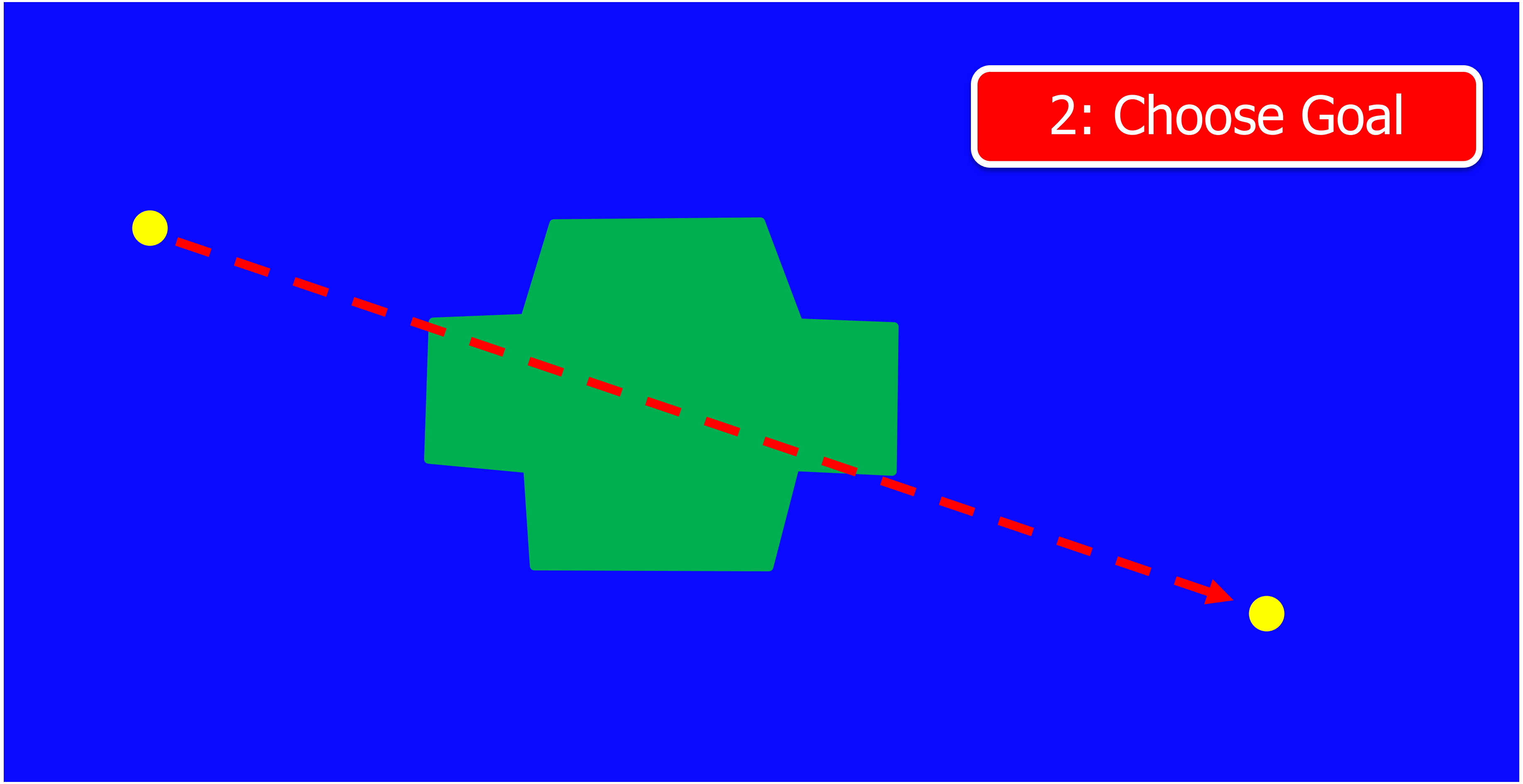
Game: Trudy's Mechanicals
Height Grid + Movement Grid
Source Link

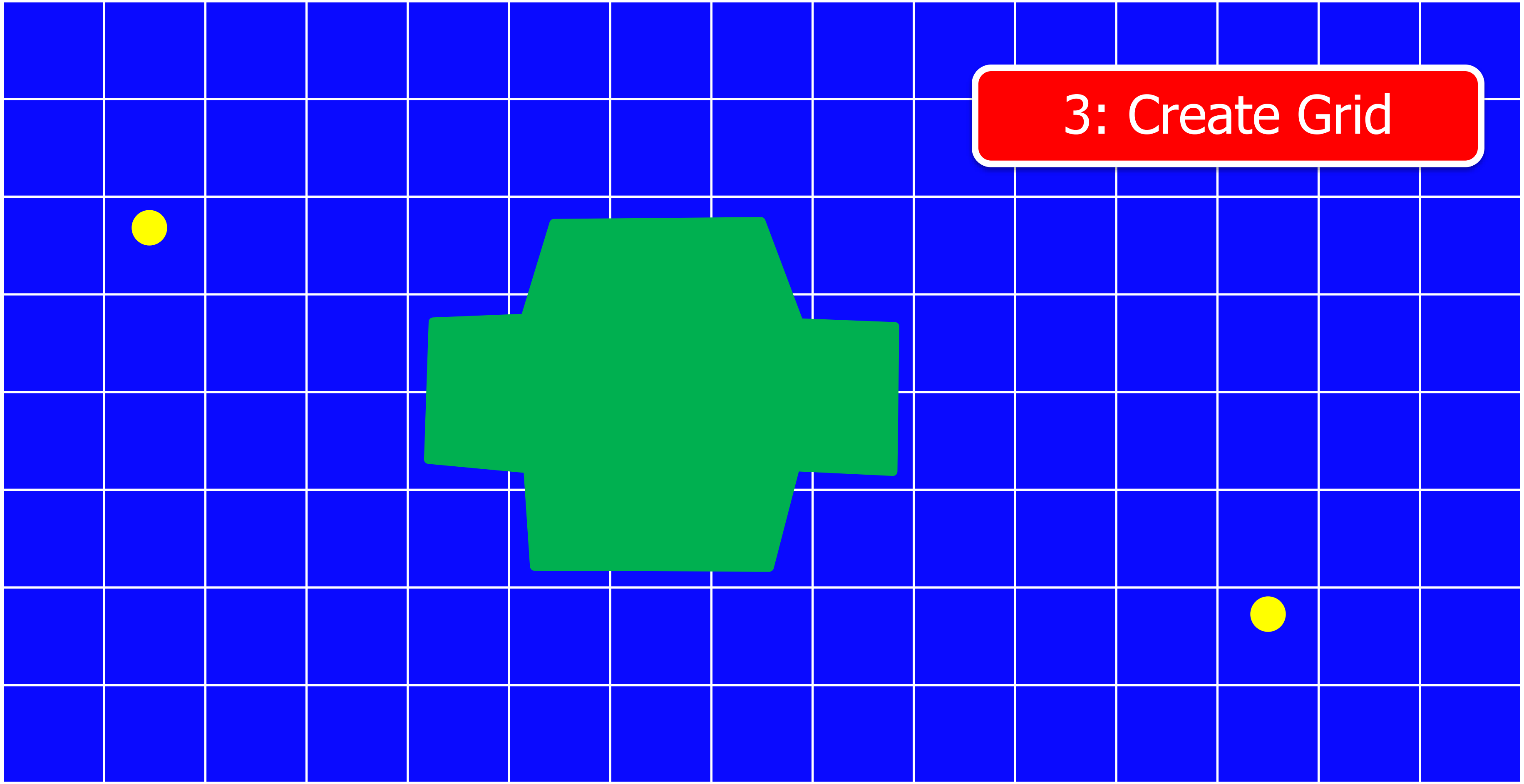
Grid-Pathfinding

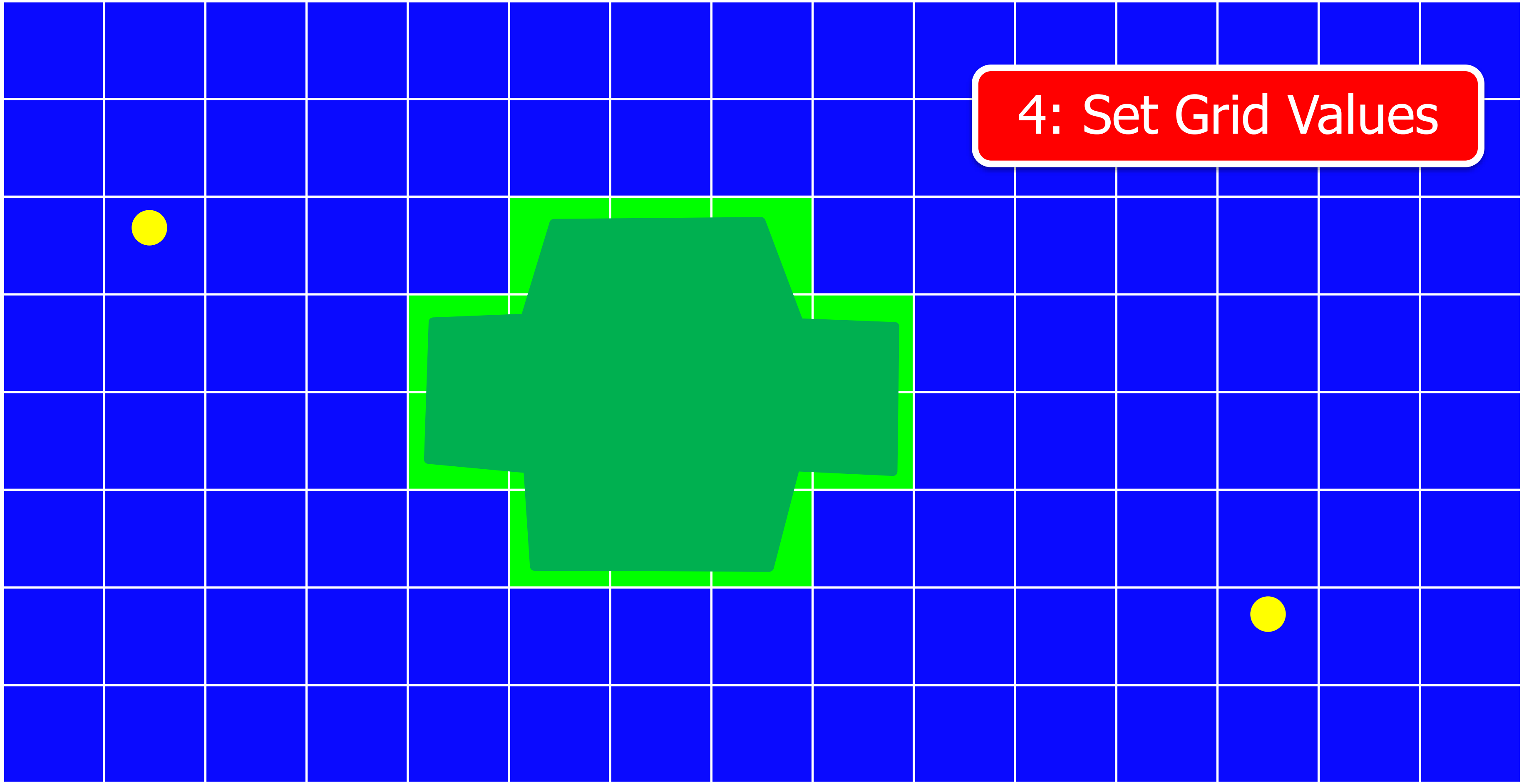
- One of the most common uses for grids is pathfinding
- Fewer positions and actions to consider in algorithm than native game world space
- How can we do grid-based pathfinding in nongrid world?

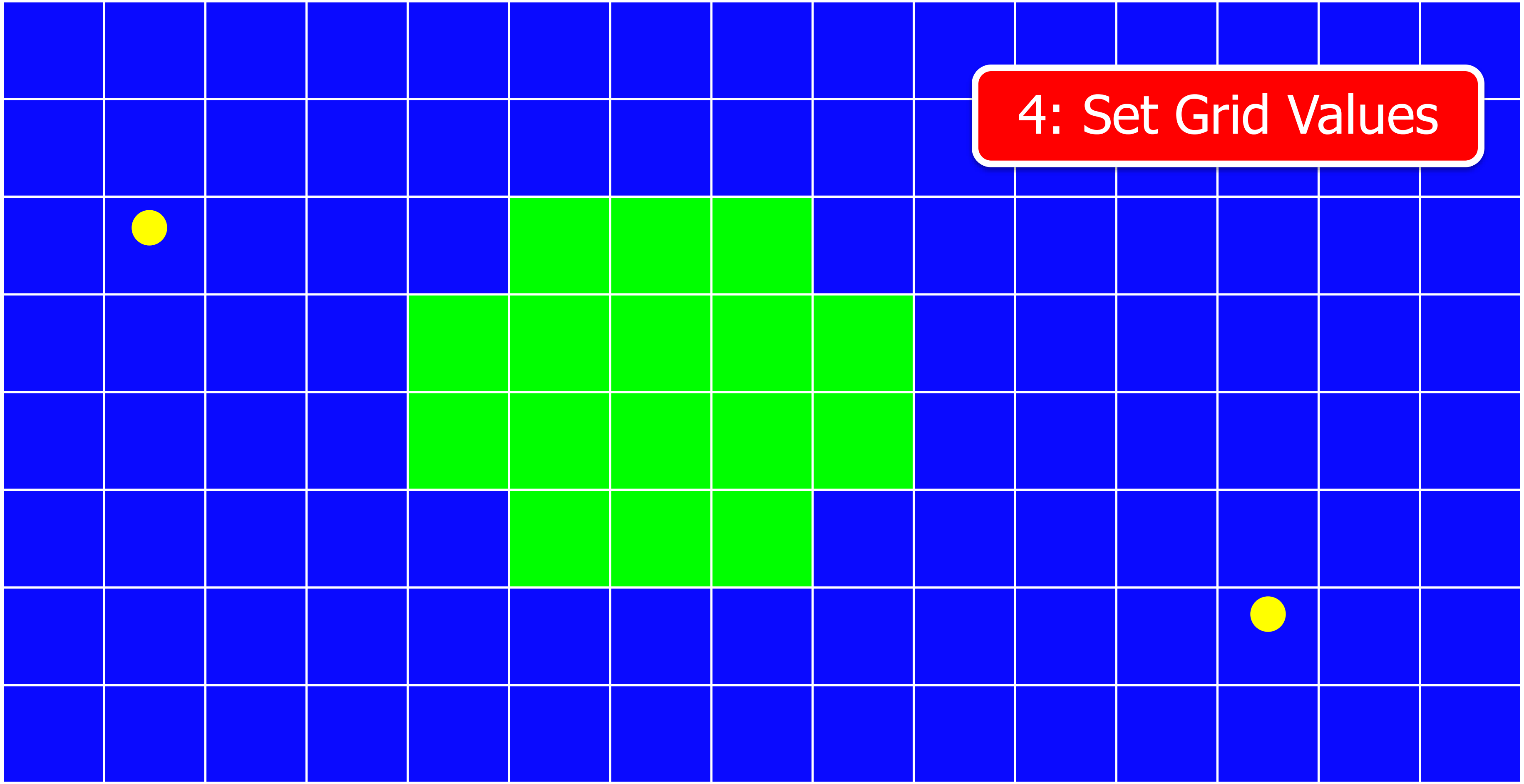



1: Game World

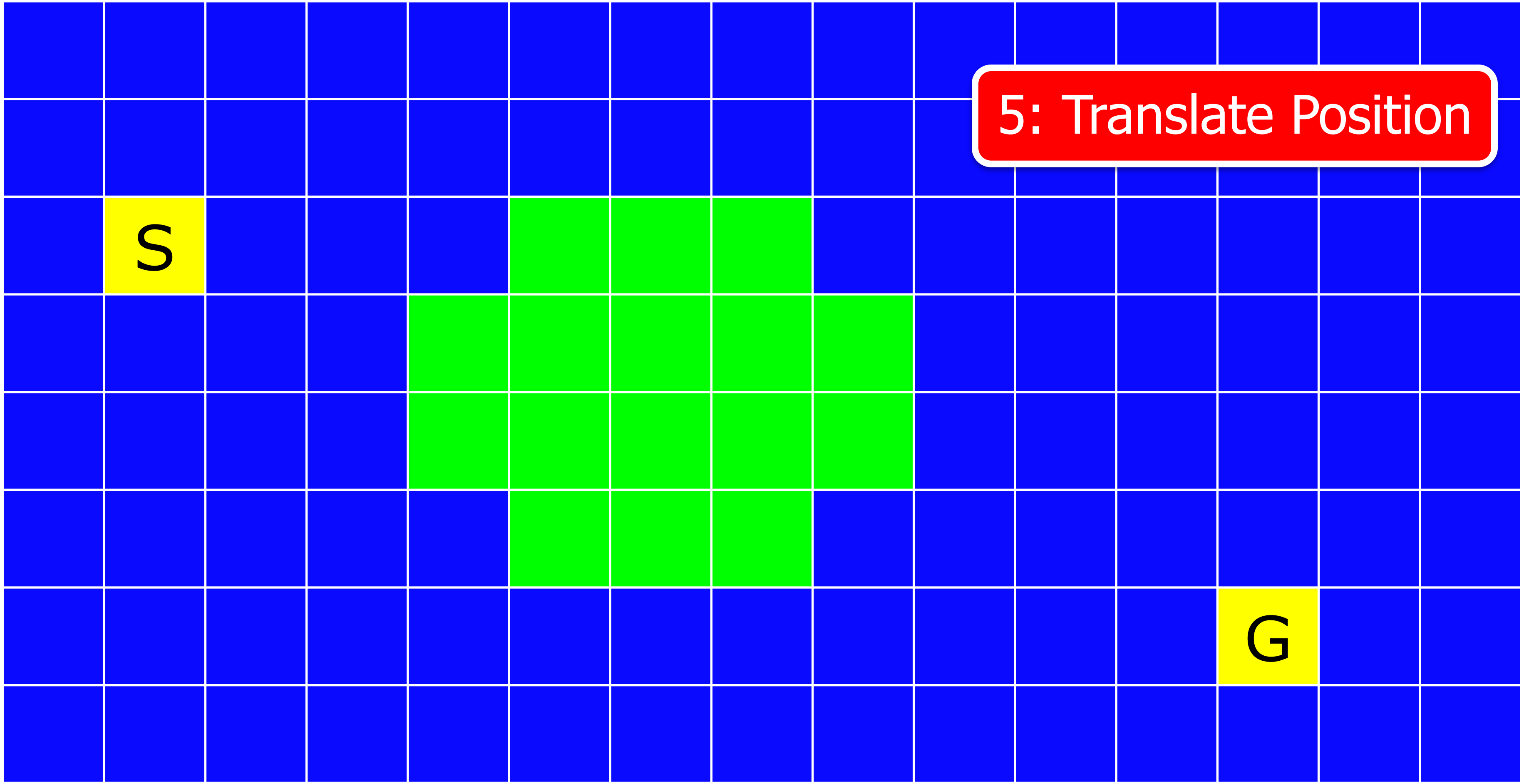

2: Choose Goal

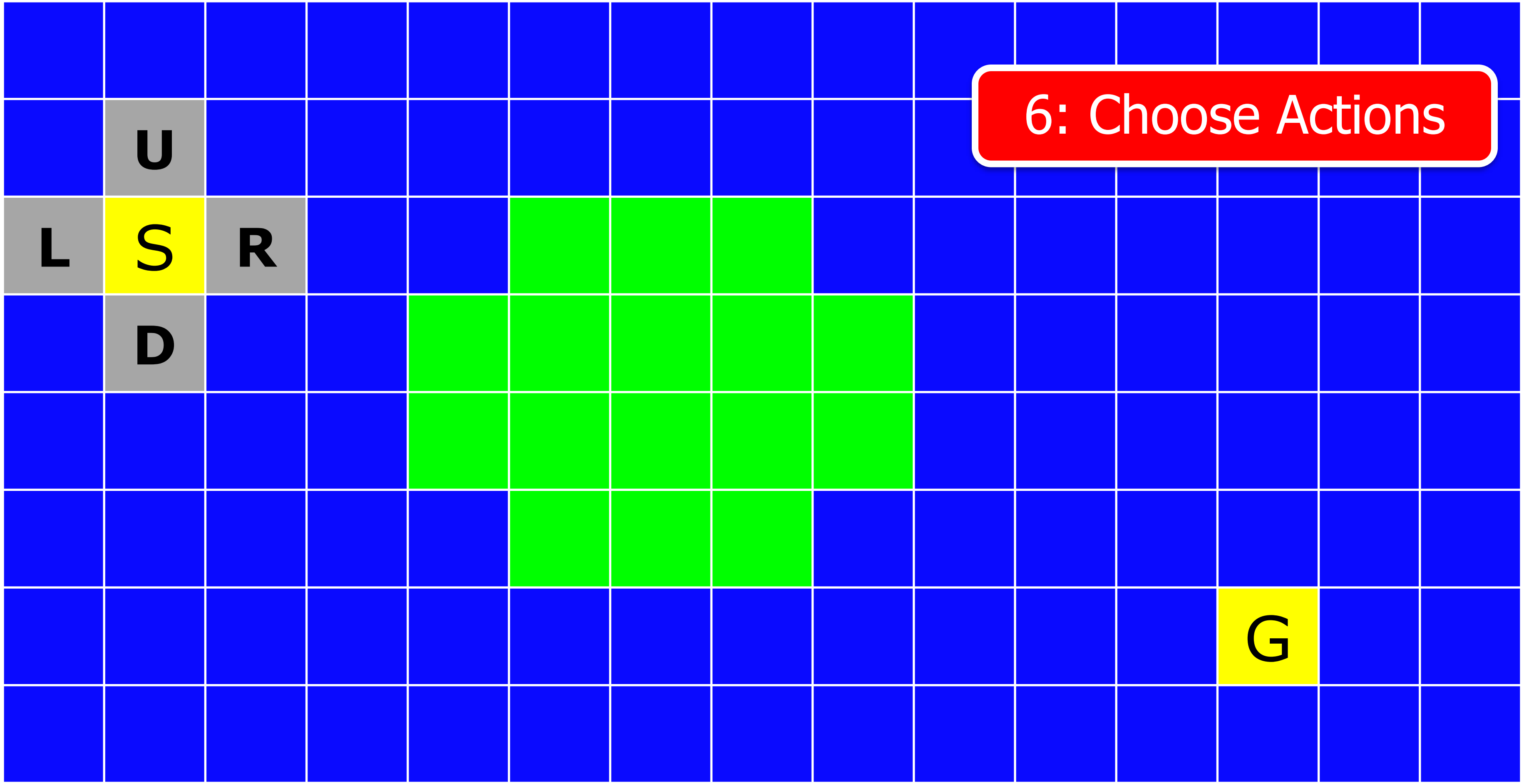

3: Create Grid


4: Set Grid Values

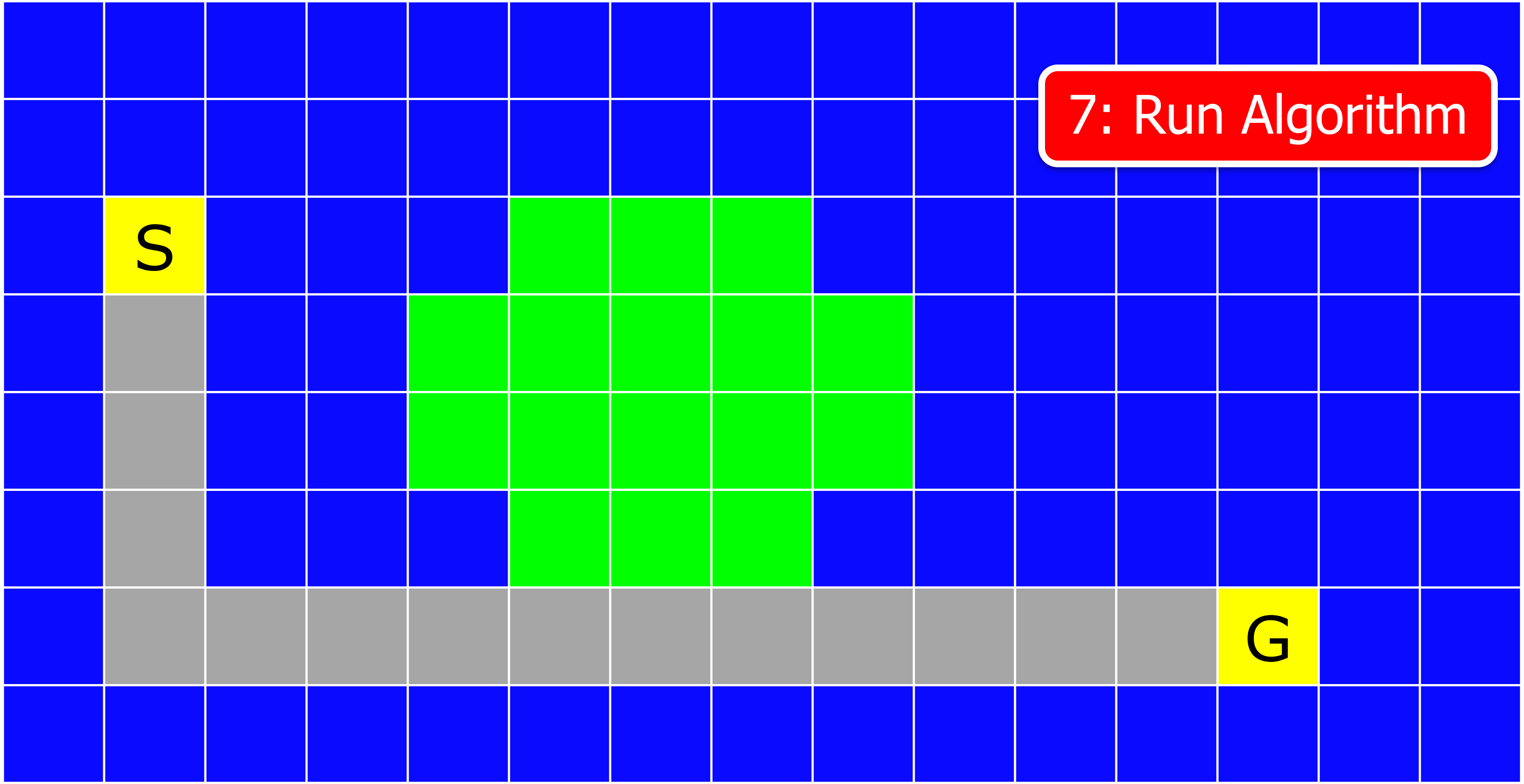

4: Set Grid Values

5: Translate Position

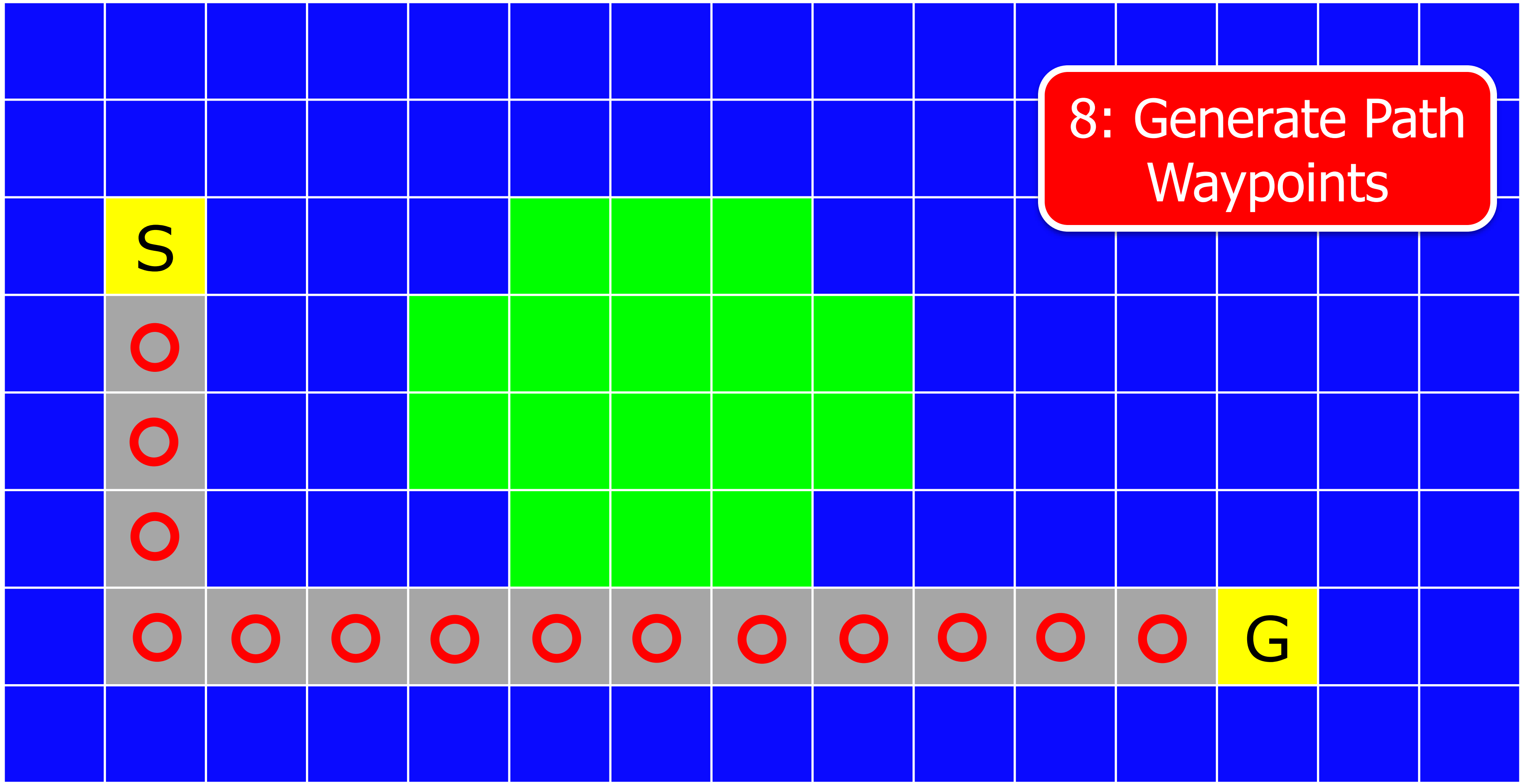


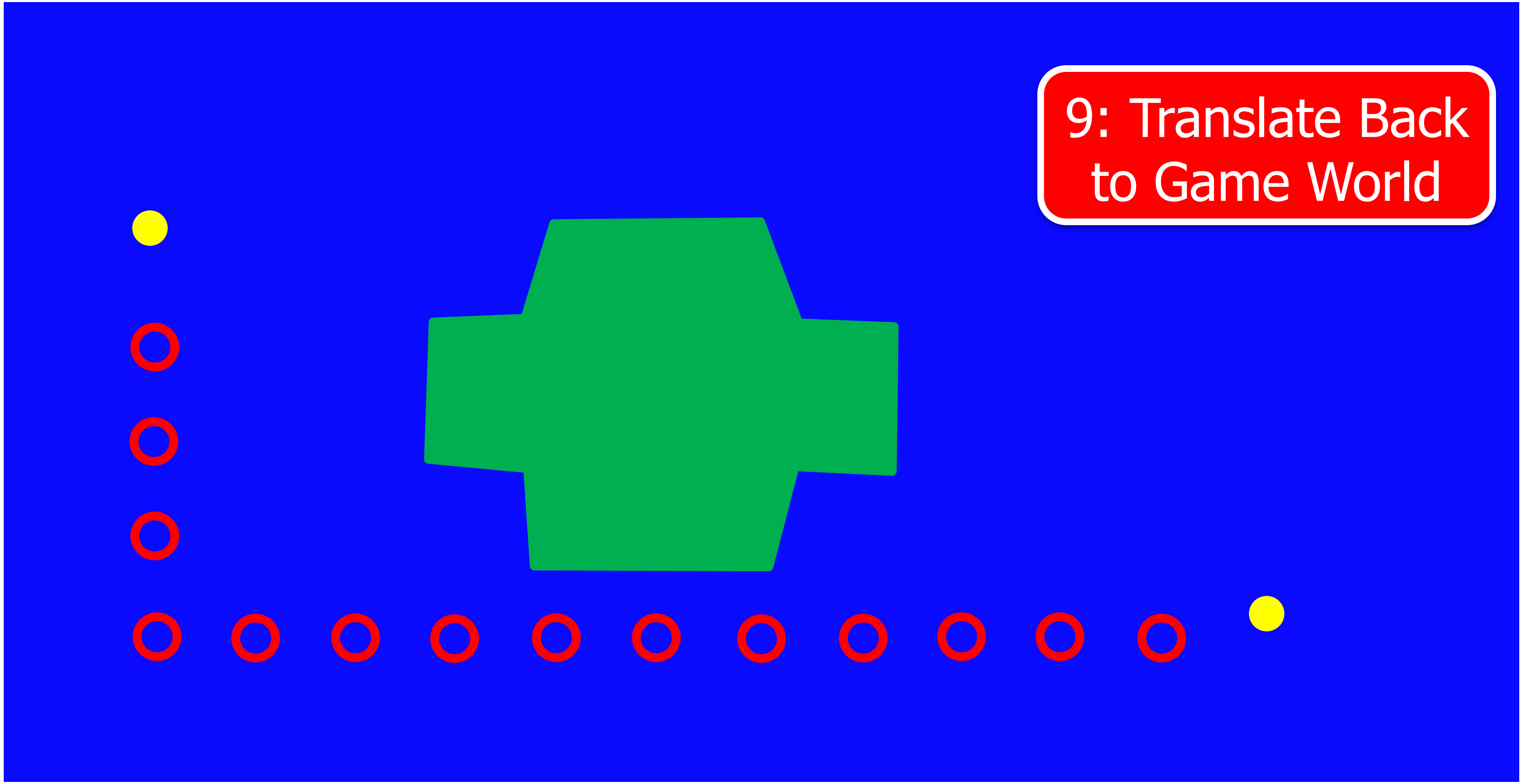

6: Choose Actions
U
L
S
R
D
G

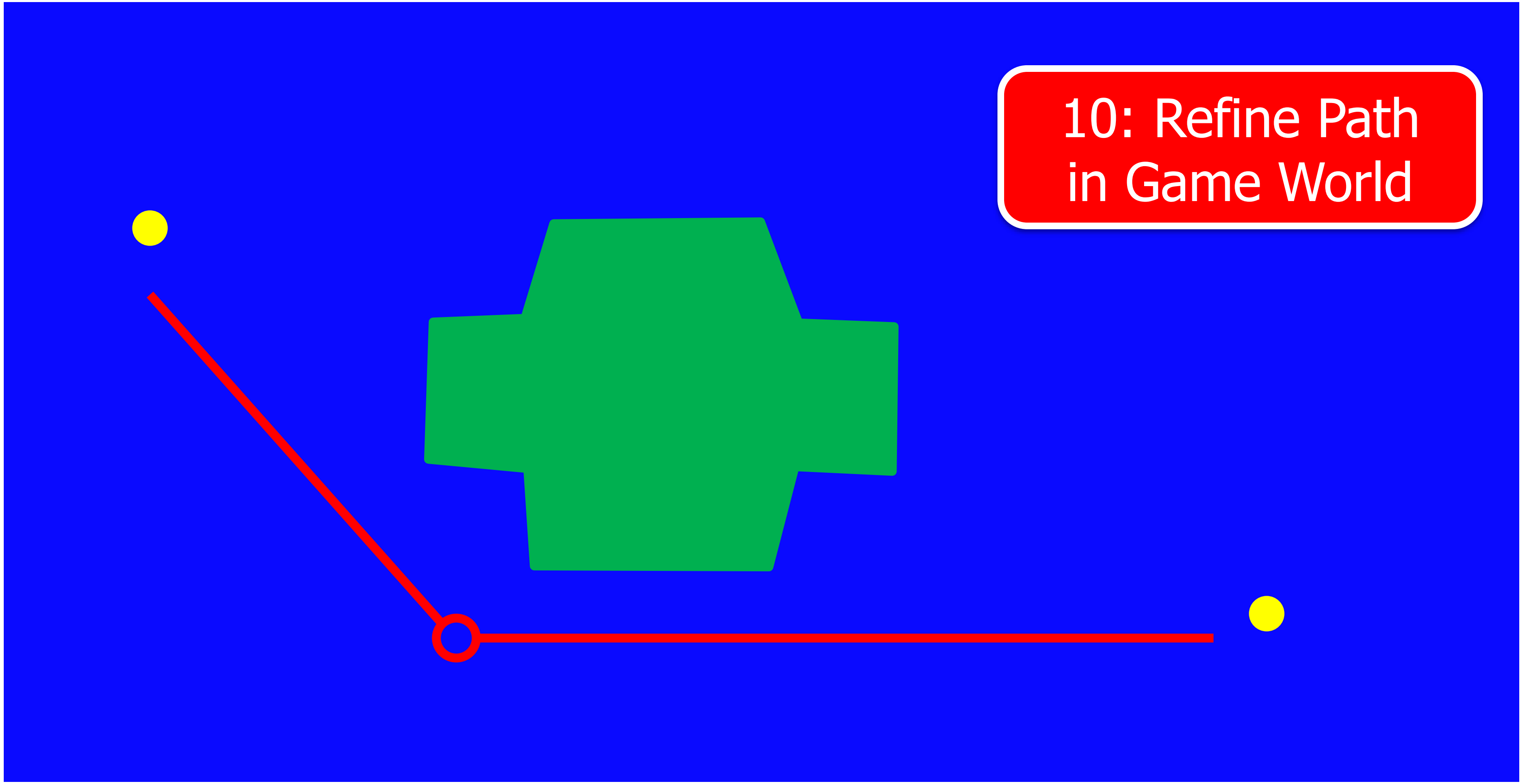
7: Run Algorithm



8: Generate Path Waypoints

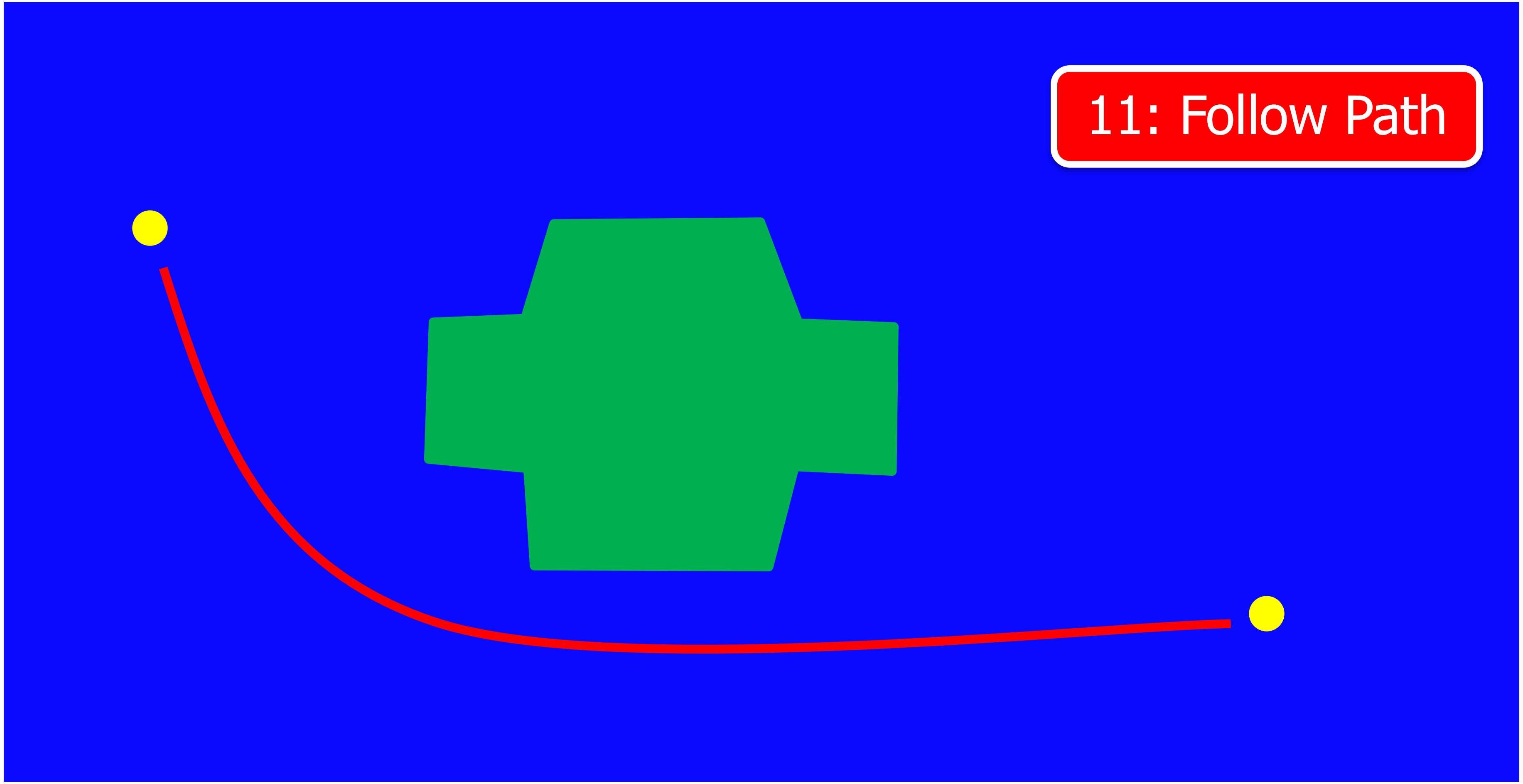



9: Translate Back to Game World



10: Refine Path in Game World

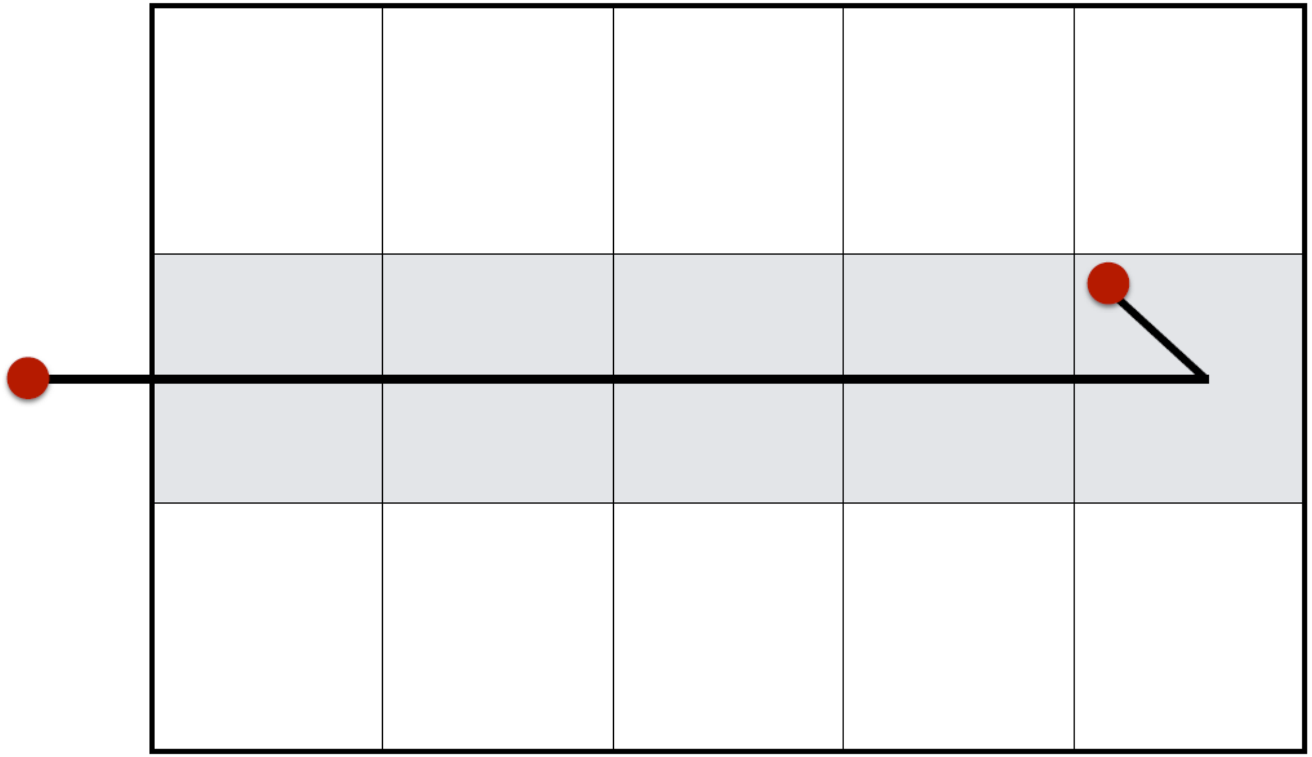
11: Follow Path



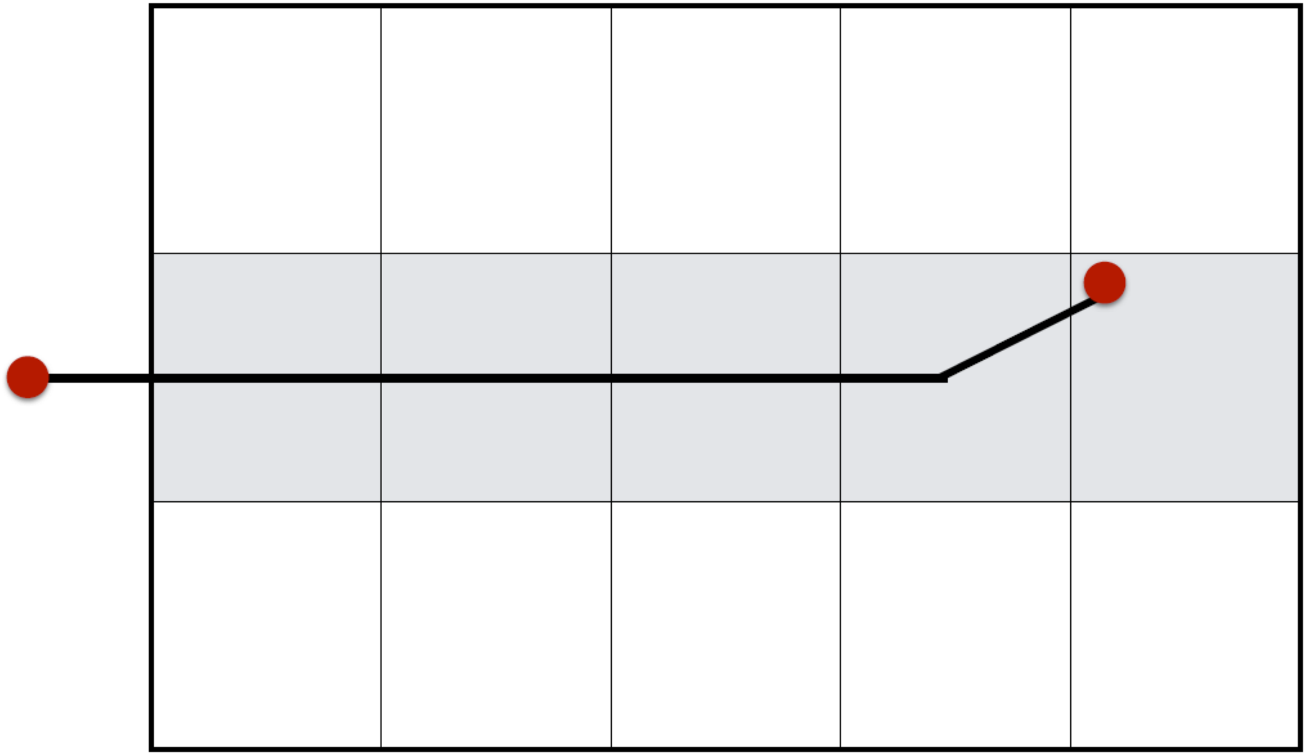
Grids: Advantages

- Very easy to implement (2D array)
- Easy to visualize / intuitive usage
- Quick to modify for dynamic changes
- Arrays are very cache-friendly
- Easy to specify / understand actions
  - 4 or 8 directions, adjacent cells
- Any graph-search algorithm works on grid
- Some algorithms ONLY work on grids

Grid Disadvantage: Localization

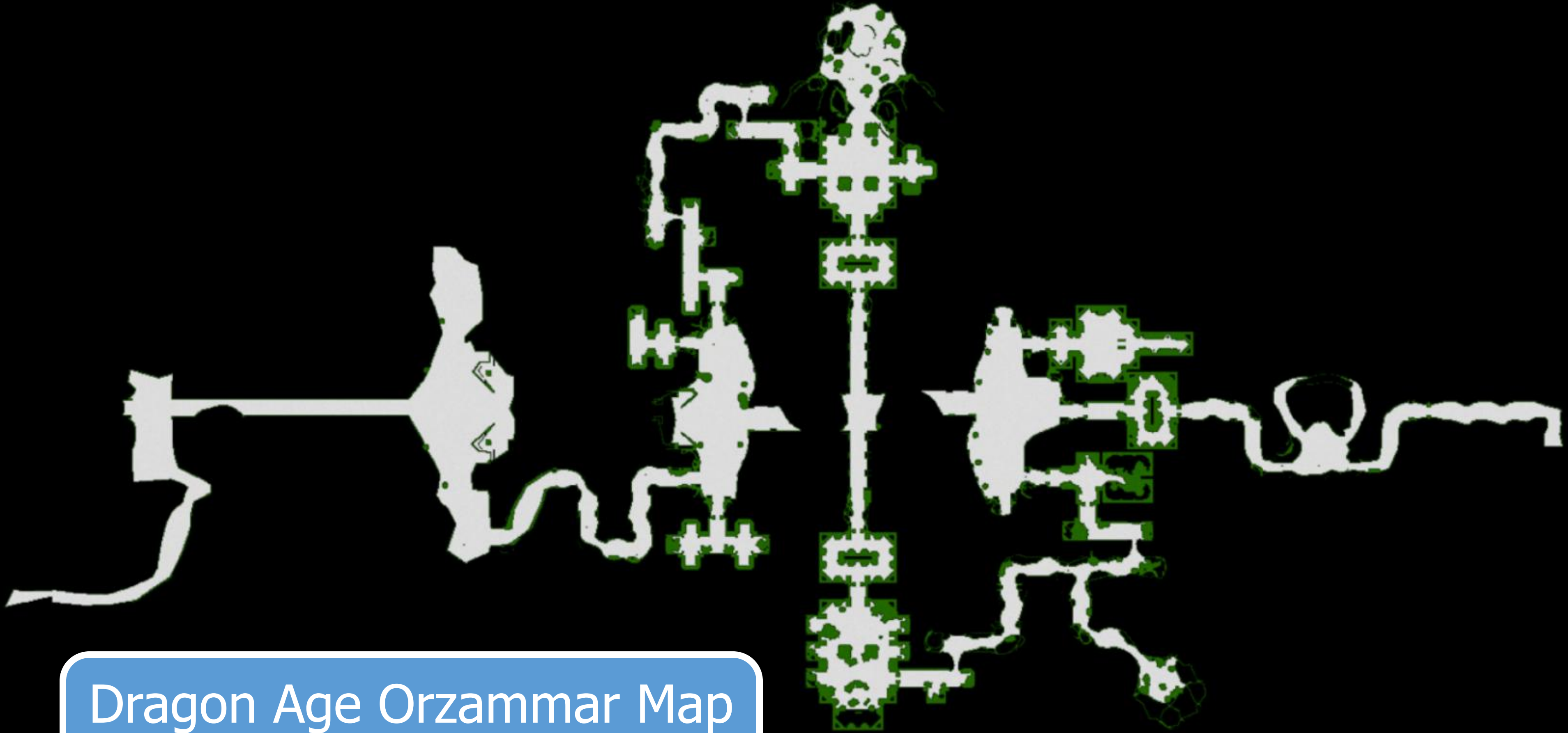


Grid Disadvantage: Localization

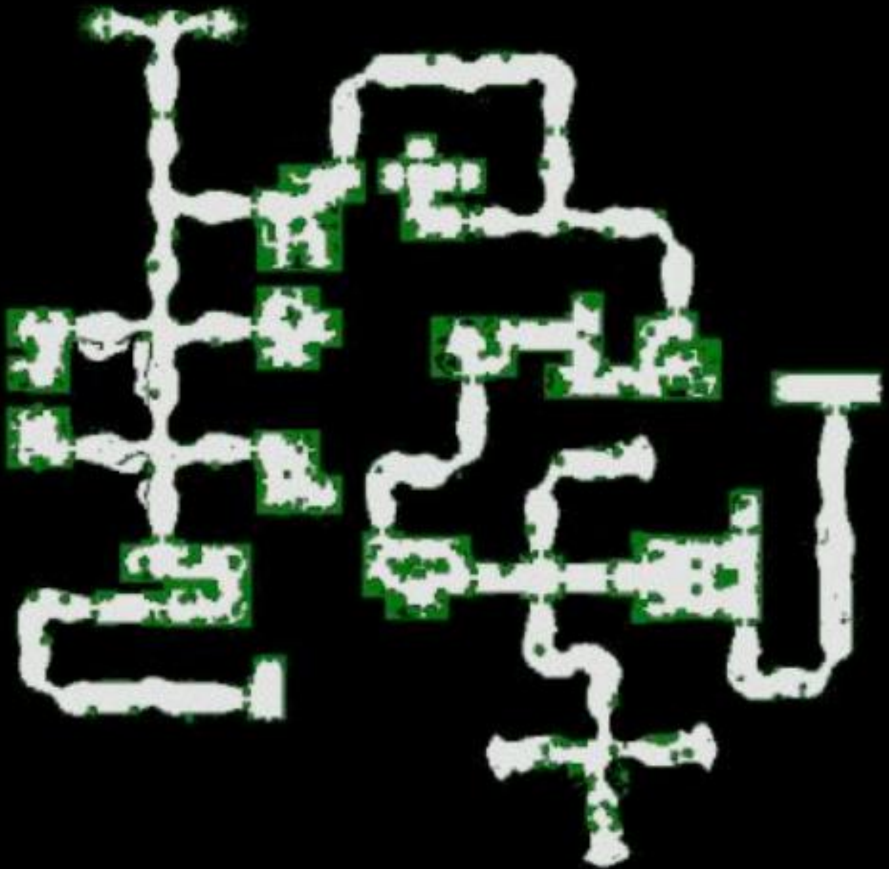


Grids: Disadvantages

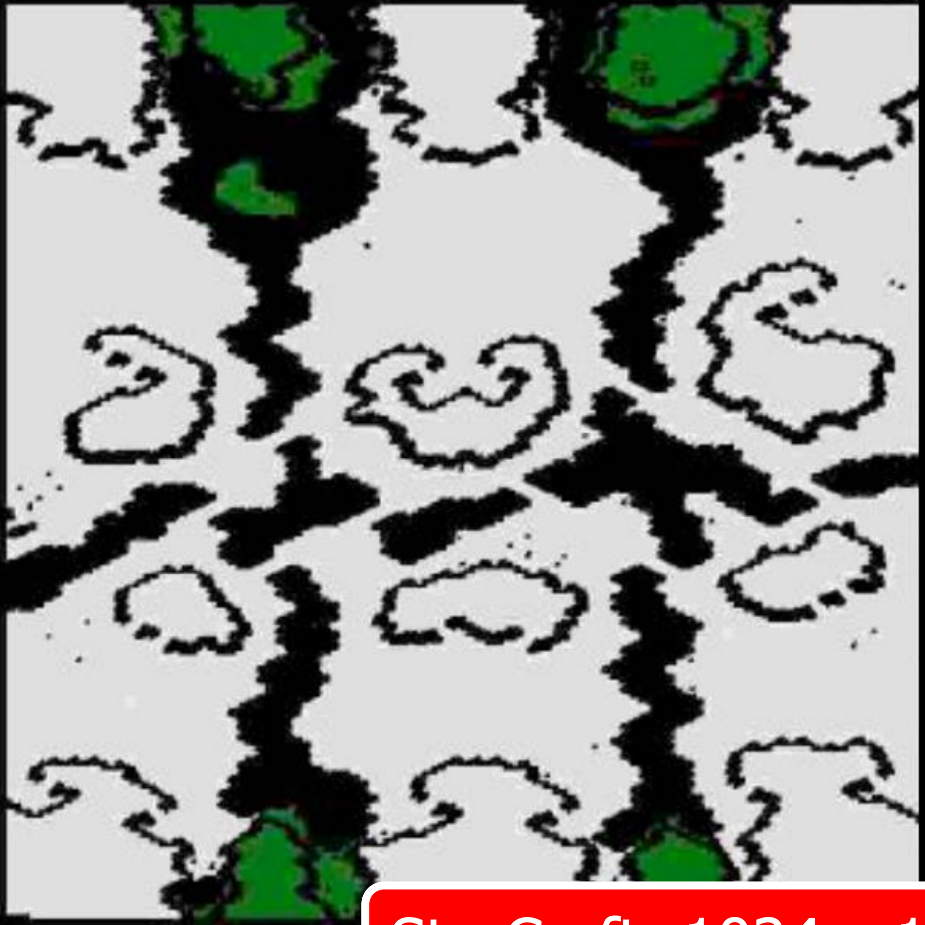
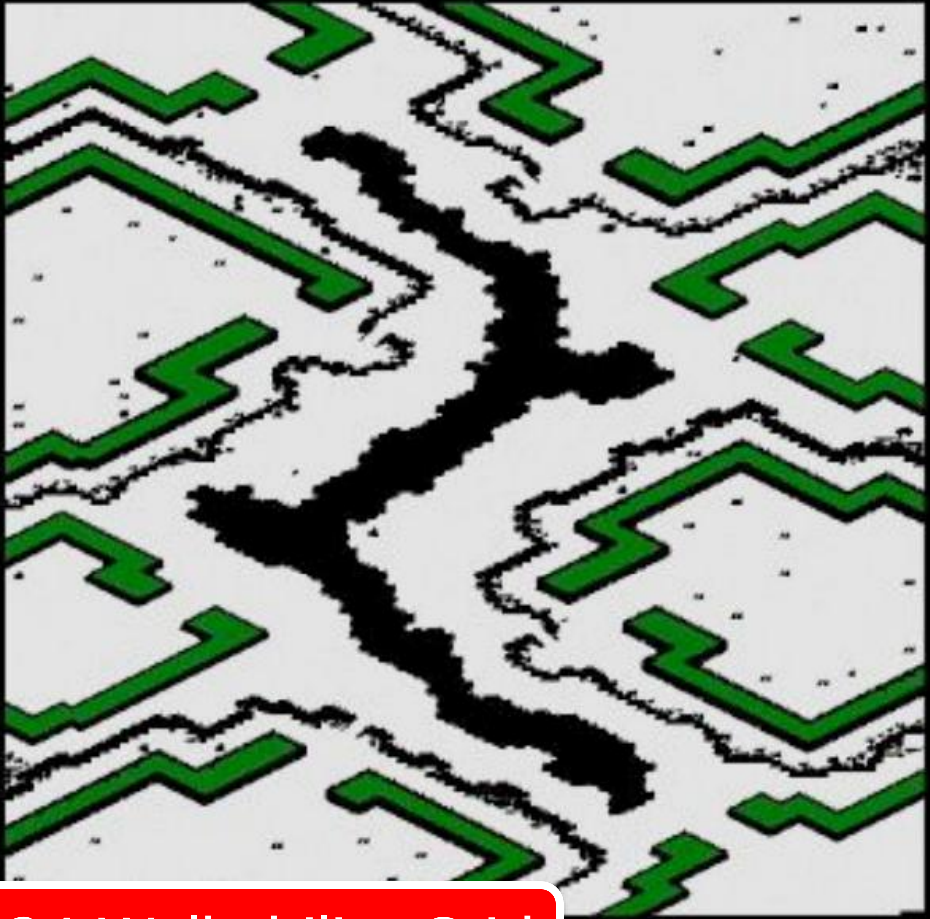
- Grid abstraction may lose precision
  - Game entities may have real-valued position
- Complex geometry not well captured
  - Rectangles don’t encapsulate all shapes well
- Grid memory usage can be high
  - Grids inefficient for sparse spaces
  - Ex: large level with just a few blocked cells



Dragon Age Orzammar Map
96,603 Walkable Tiles



Dragon Age: Origins



StarCraft: 1024 x 1024 Walkability Grid

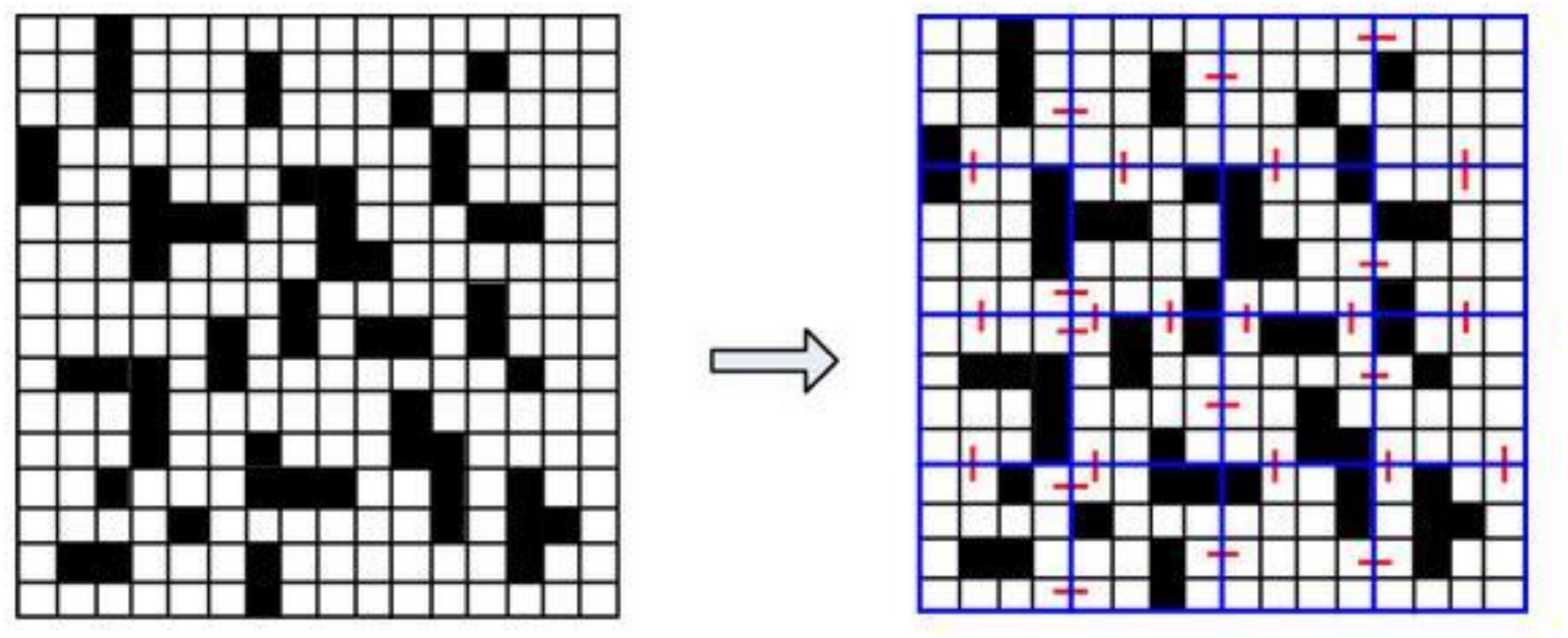
Representation Considerations

- Localization
  - How do you find where you are in the rep?
- Generation
  - How is the representation generated?
- Dynamic Changes
  - How do you handle dynamic changes?
- Planning / Pathfinding
  - How are actions computed / represented?
- Memory / Speed
  - How much memory / time required to compute paths?

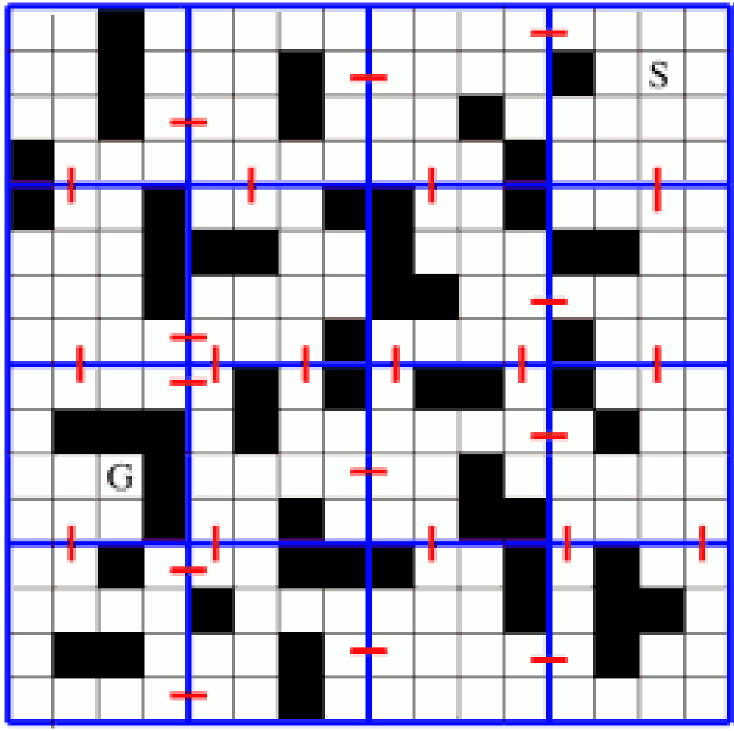
Dealing With Large Grids

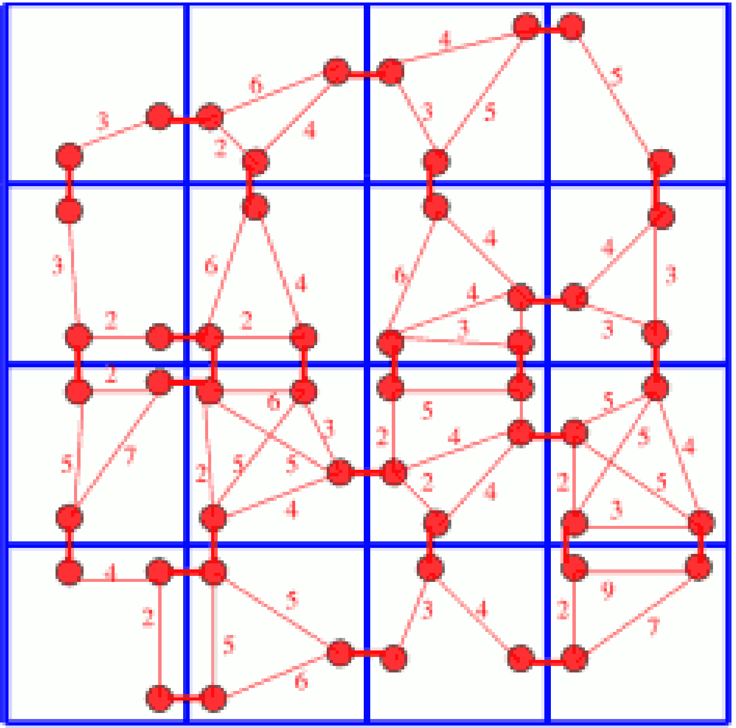
- Large grids take a lot of memory, and potentially hold very few values
- Large grids also cause algorithms to run quite slowly / cost more memory
  - StarCraft map: 1 million walk cells!
- If possible, avoid such large grids
- However, can we still use them?

Hierarchical Grids



Grid Sector Connections



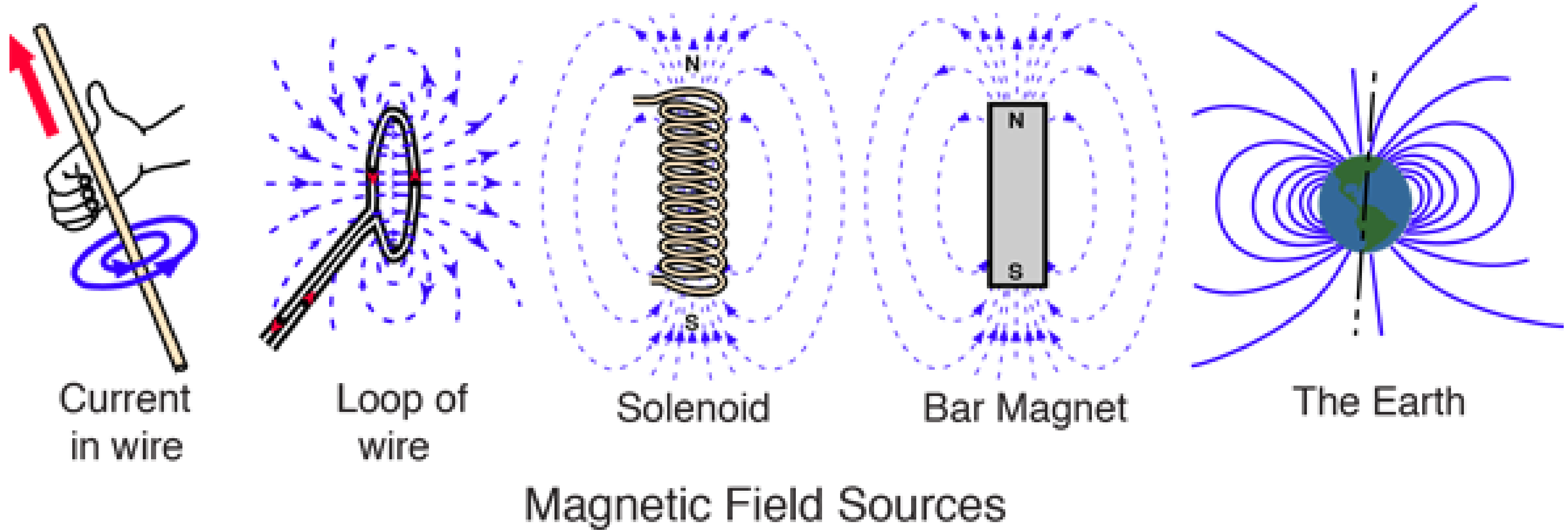


Vector Fields

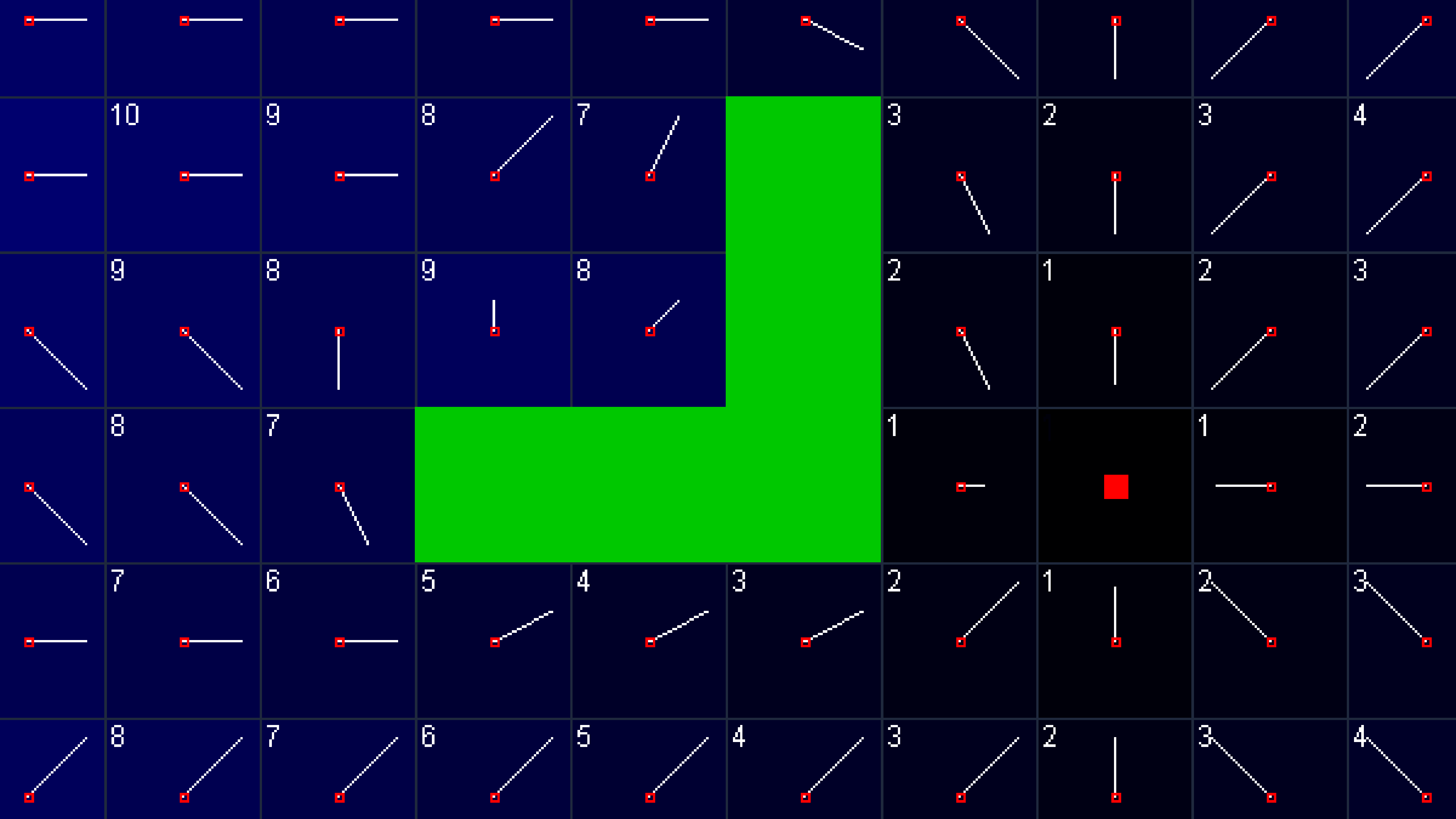
Vector Fields

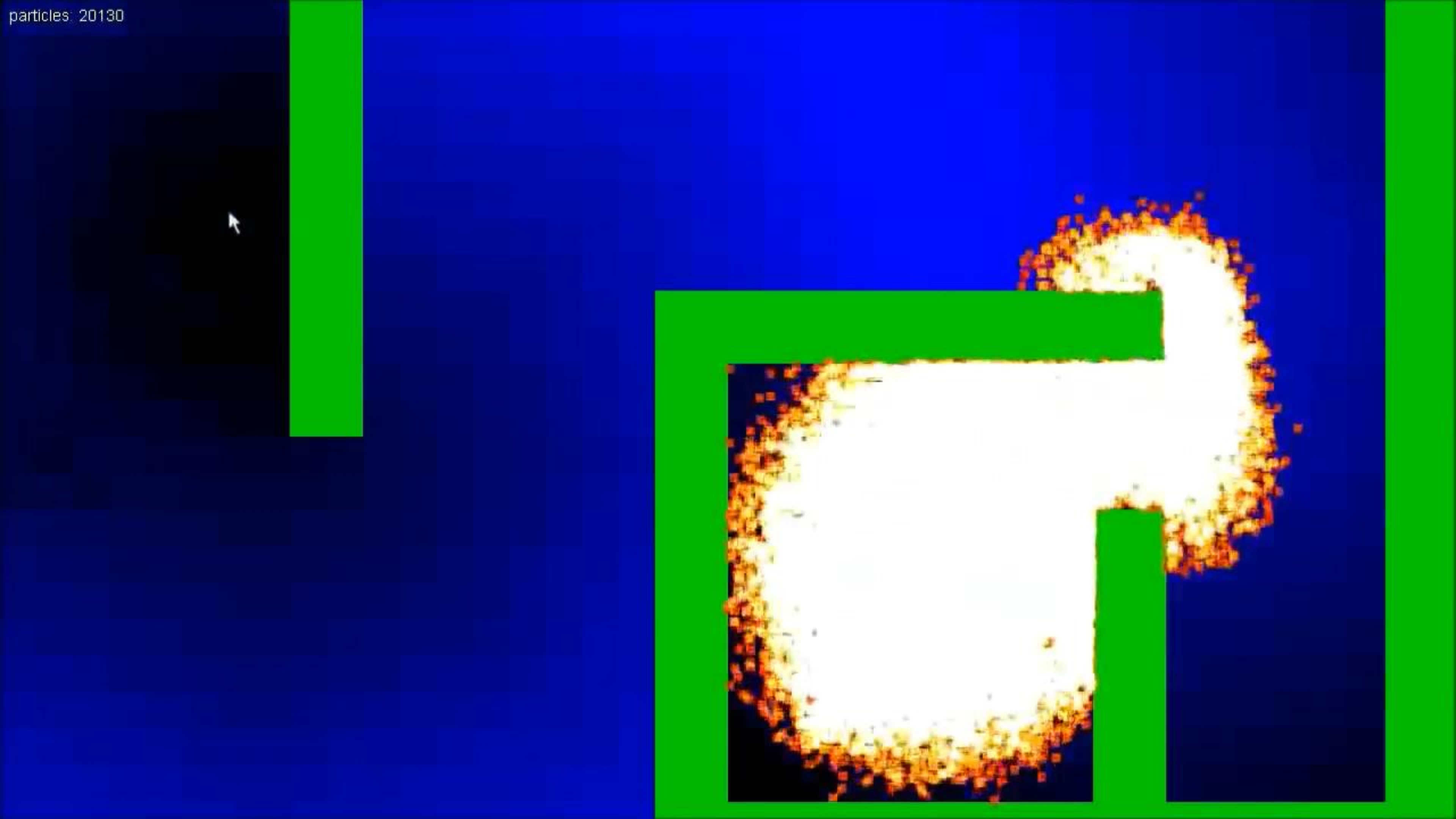
- In math, a vector field is an assignment of a vector to each point in a space
  - Vector calculus, differential eqn, physics
- In game programming, a vector will be assigned to each cell of a grid
- This vector denotes the desired action or direction of movement from that cell

Vector Fields in Science



http://hyperphysics.phy-astr.gsu.edu/hbase/magnetic/magfie.html


10
9
8
7
3
2
3
4
9
8
9
8
2
1
2
3
8
7
1
1
2
7
6
5
4
3
2
1
2
3
8
7
6
5
4
3
2
3
4


particles: 20130

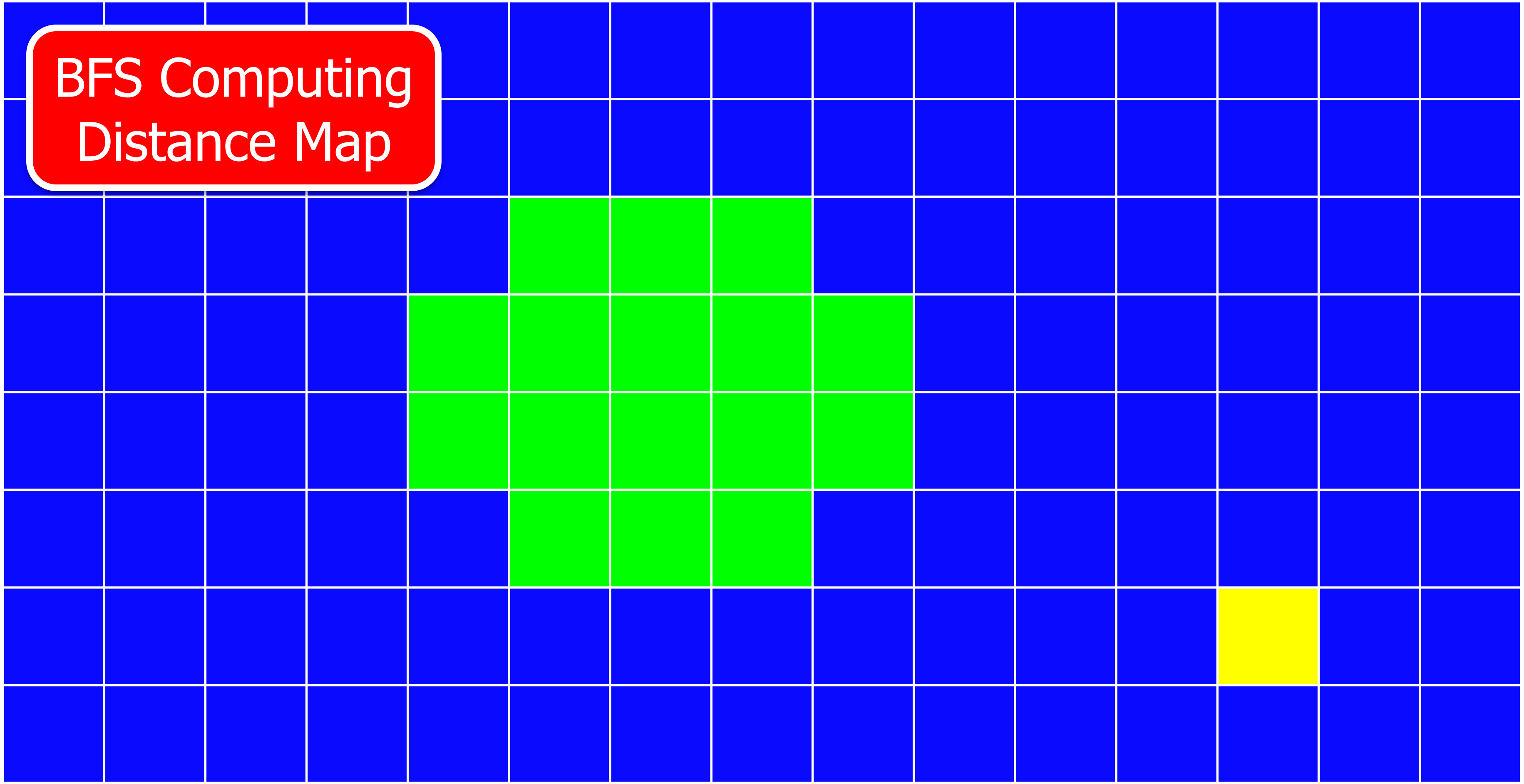
Vector Fields for Games

- Flow fields, influence maps, vector maps
- Very efficient structure for path-finding in case of ‘many paths to single goal’
- Any case where multiple NPCs need to go to a single location, consider using these
- Can be used to accomplish a number of tasks, not just path-finding

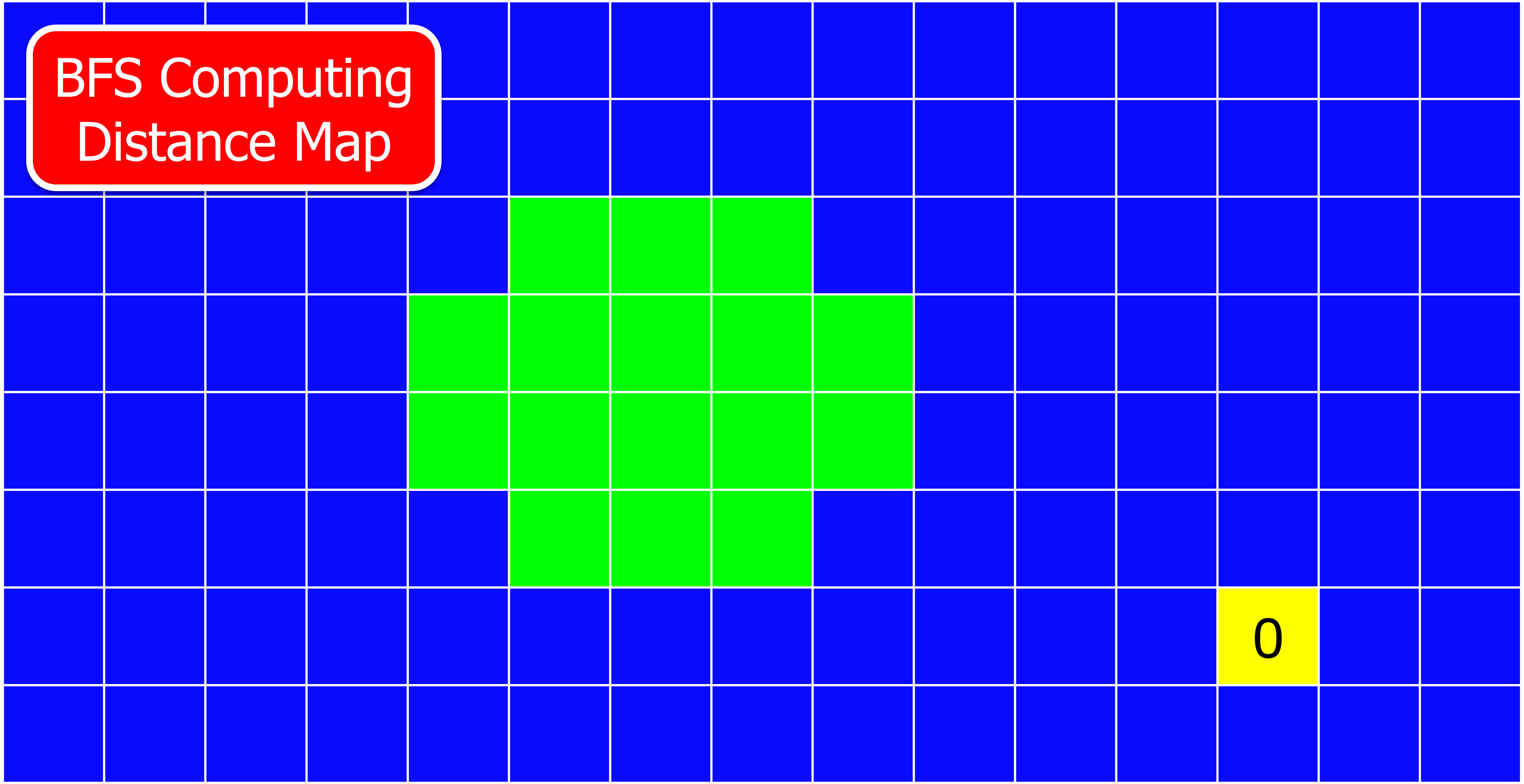
Distance Map

- Before we can compute the vector field, we need to compute a Distance Map (grid)
- Each cell of this grid will store the 4D shortest path distance to a single goal cell
- In practice, we will need one Distance Map per path-finding destination, and we can cache the maps if the env. is static

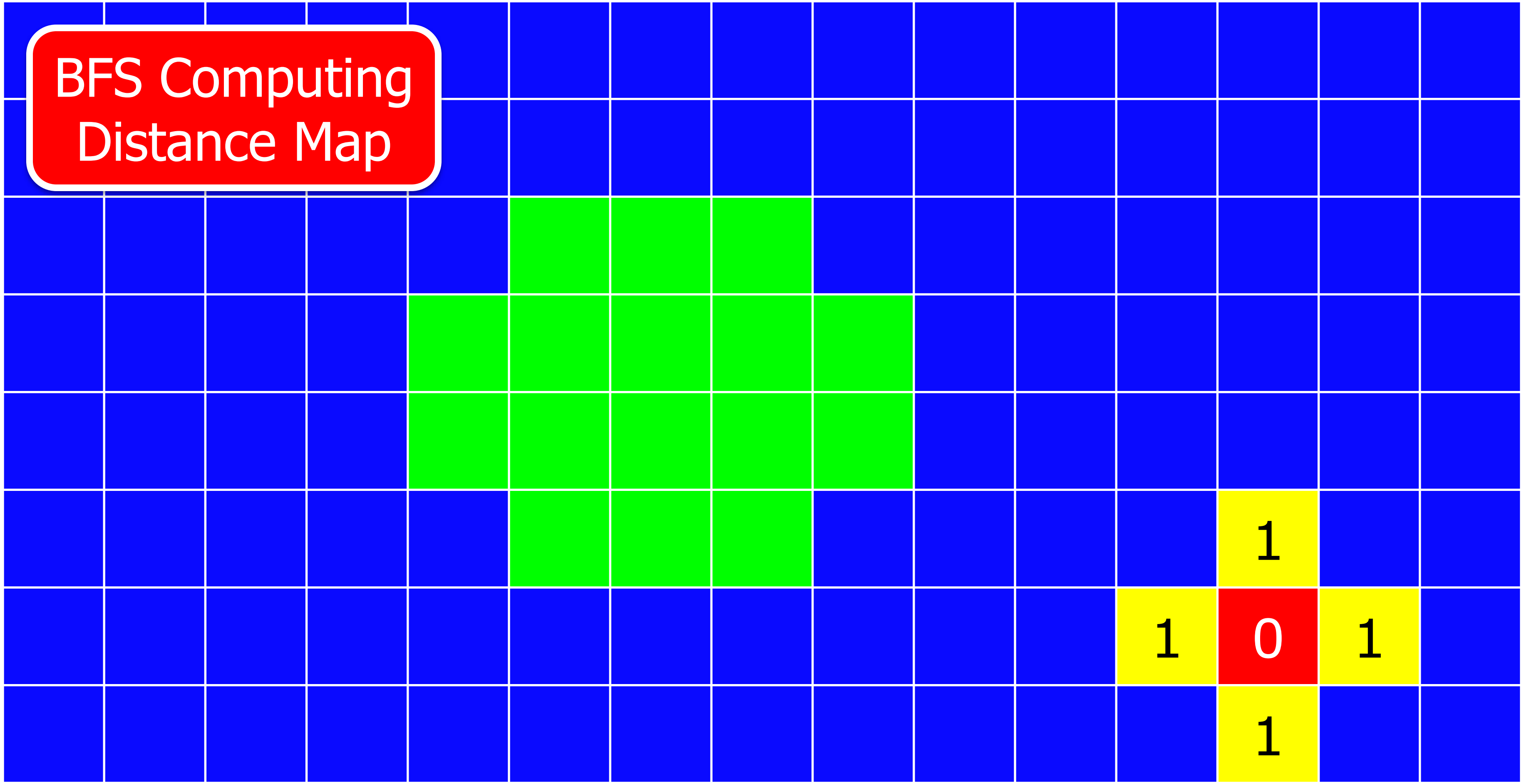
BFS Computing Distance Map



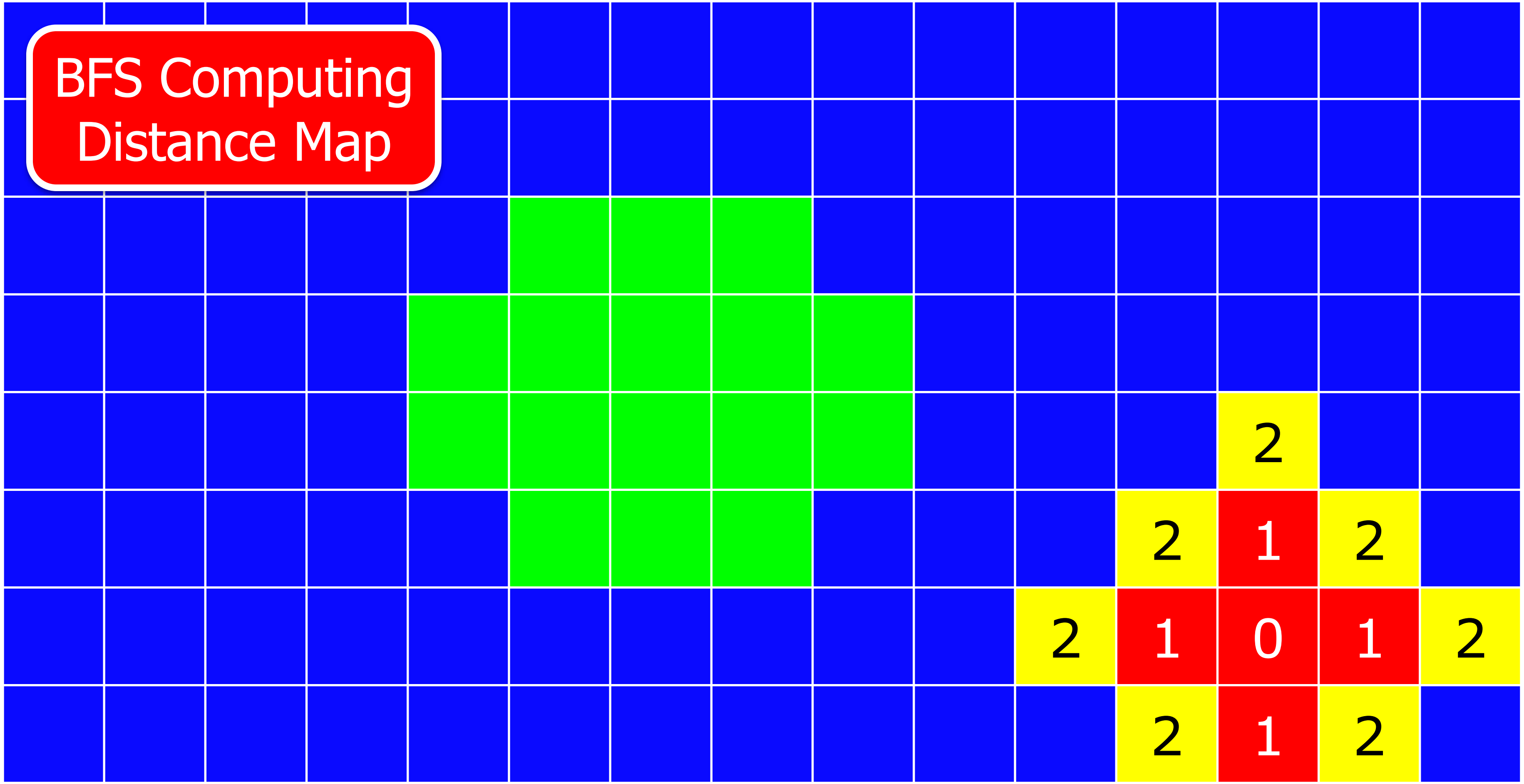
BFS Computing Distance Map



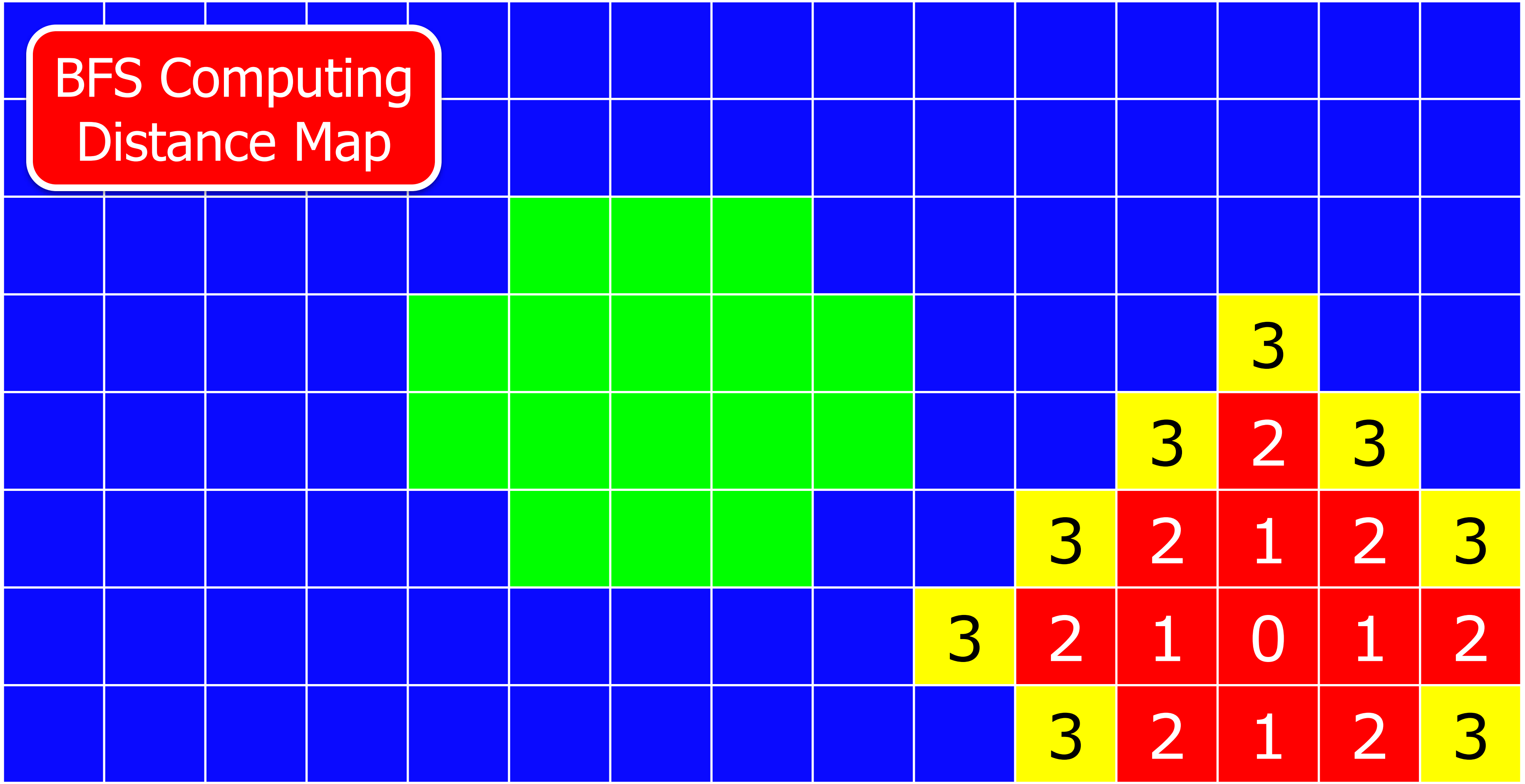
BFS Computing Distance Map



BFS Computing Distance Map



BFS Computing Distance Map



Distance Map

18	17	16	15	14	13	12	11	10	9	8	7	6	7	8
17	16	15	14	13	12	11	10	9	8					
16	15	14	13	14	🟩	🟩	🟩	8	7	6	5	4	5	6
15	14	13	12	🟩	🟩	🟩	🟩	🟩	6	5	4	3	4	5
14	13	12	11	🟩	🟩	🟩	🟩	🟩	5	4	3	2	3	4
13	12	11	10	9	🟩	🟩	🟩	5	4	3	2	1	2	3
12	11	10	9	8	7	6	5	4	3	2	1	0	1	2
13	12	11	10	9	8	7	6	5	4	3	2	1	2	3

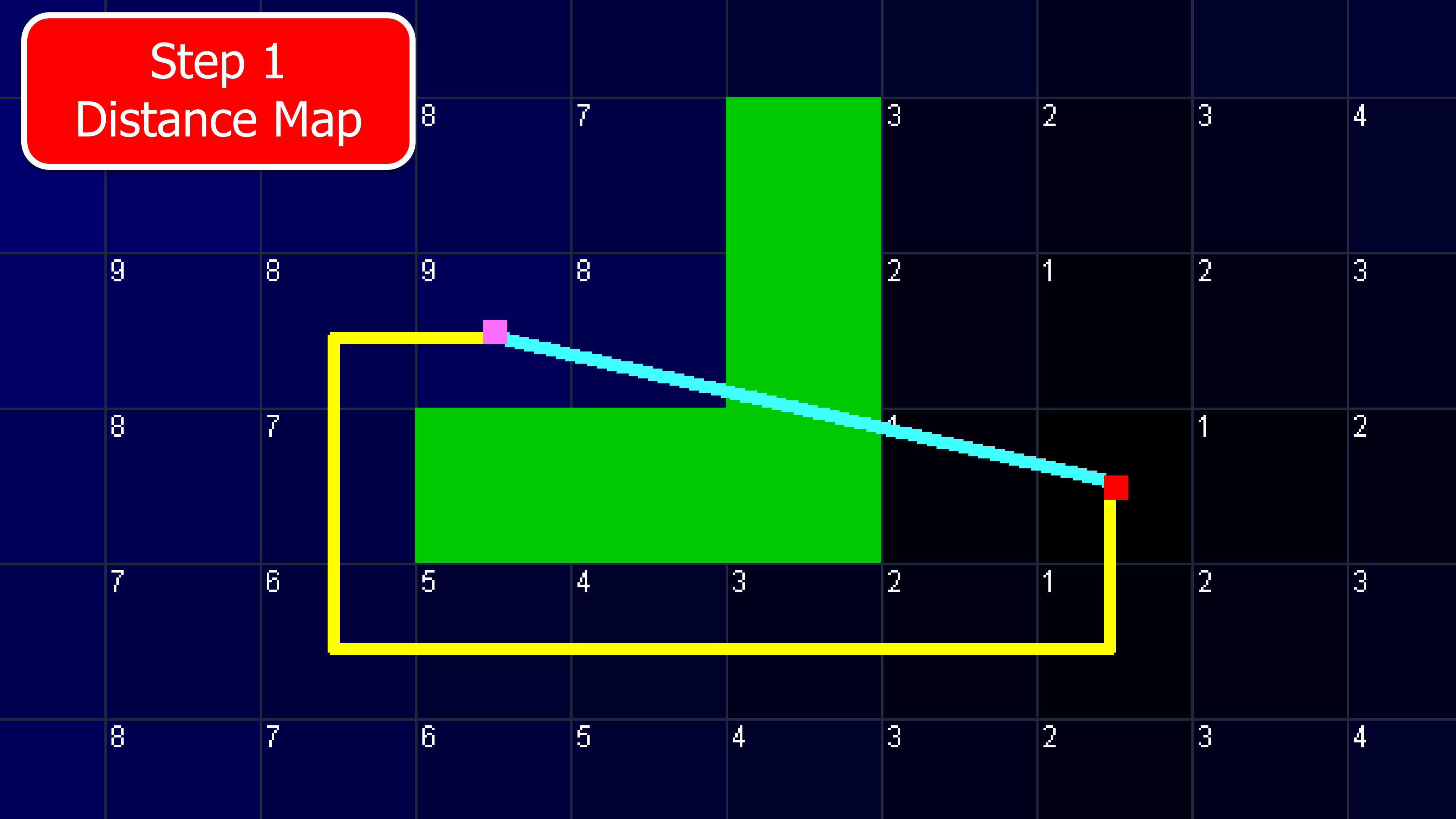
Computing Distance Map

- Distance map is computed via BFS
- Each child cell gets given a distance value of parent + 1
- Give distance map cells initial value -1
- Any distance map cell that still has a value of -1 is not reachable from the goal

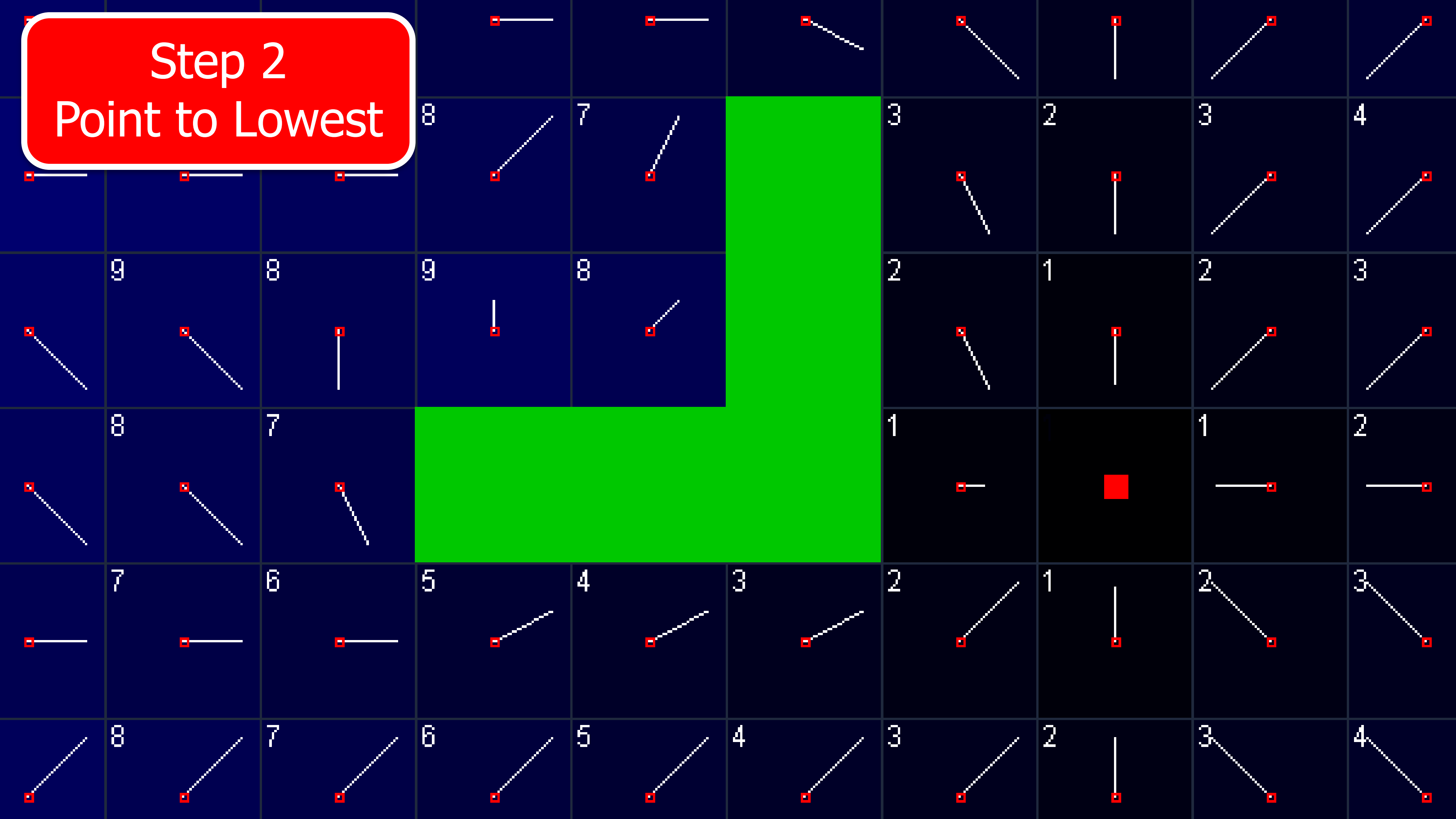
Vector Field Computation

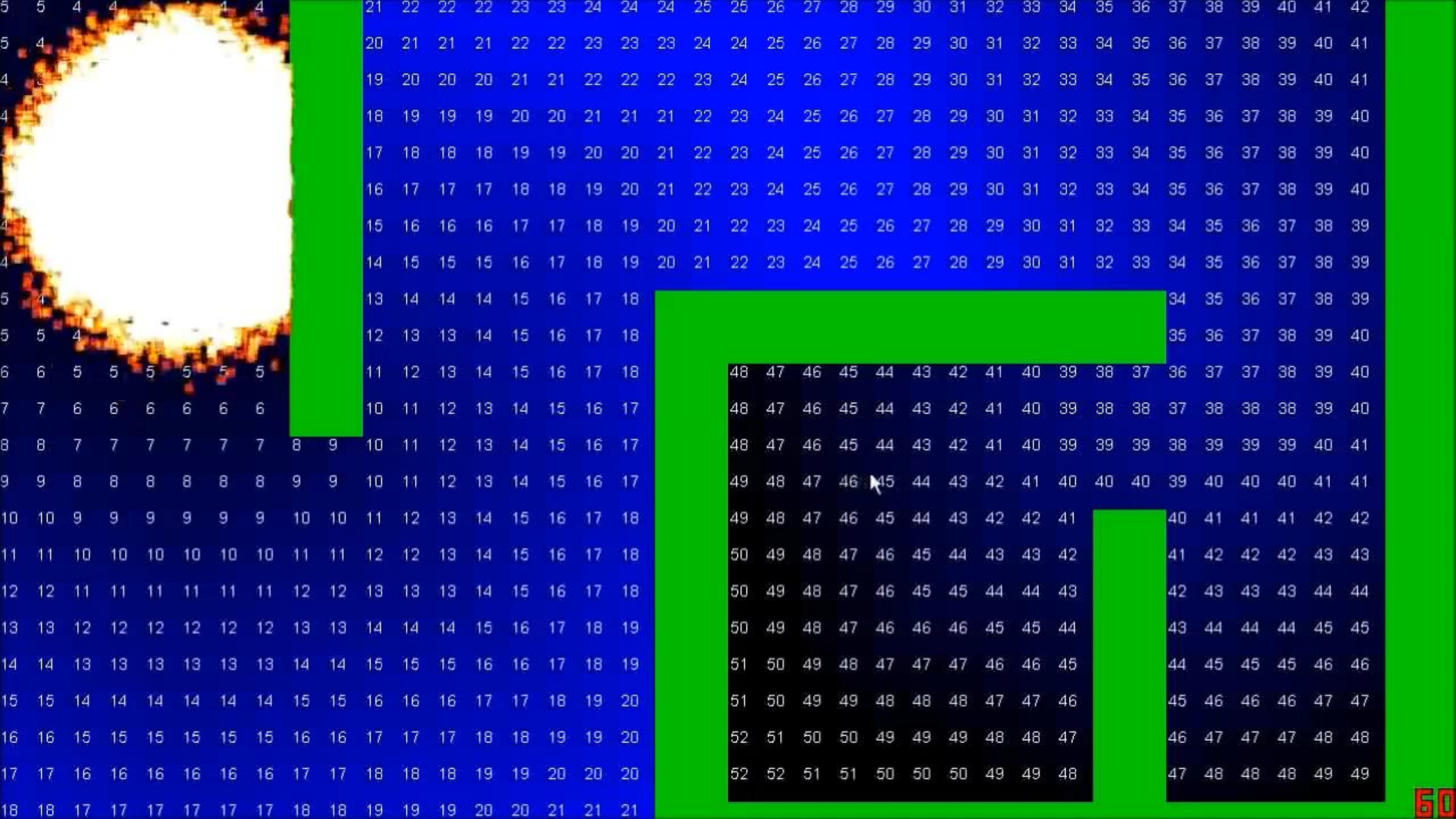
- From a distance map, it is easy to construct a vector field for pathfinding
- Each cell gets assigned a vector (or set of vectors) pointing to lowest neighbors
- (Optional) if a cell points to both left and up, instead replace with diagonal up/left
- When moving, follow the vectors!

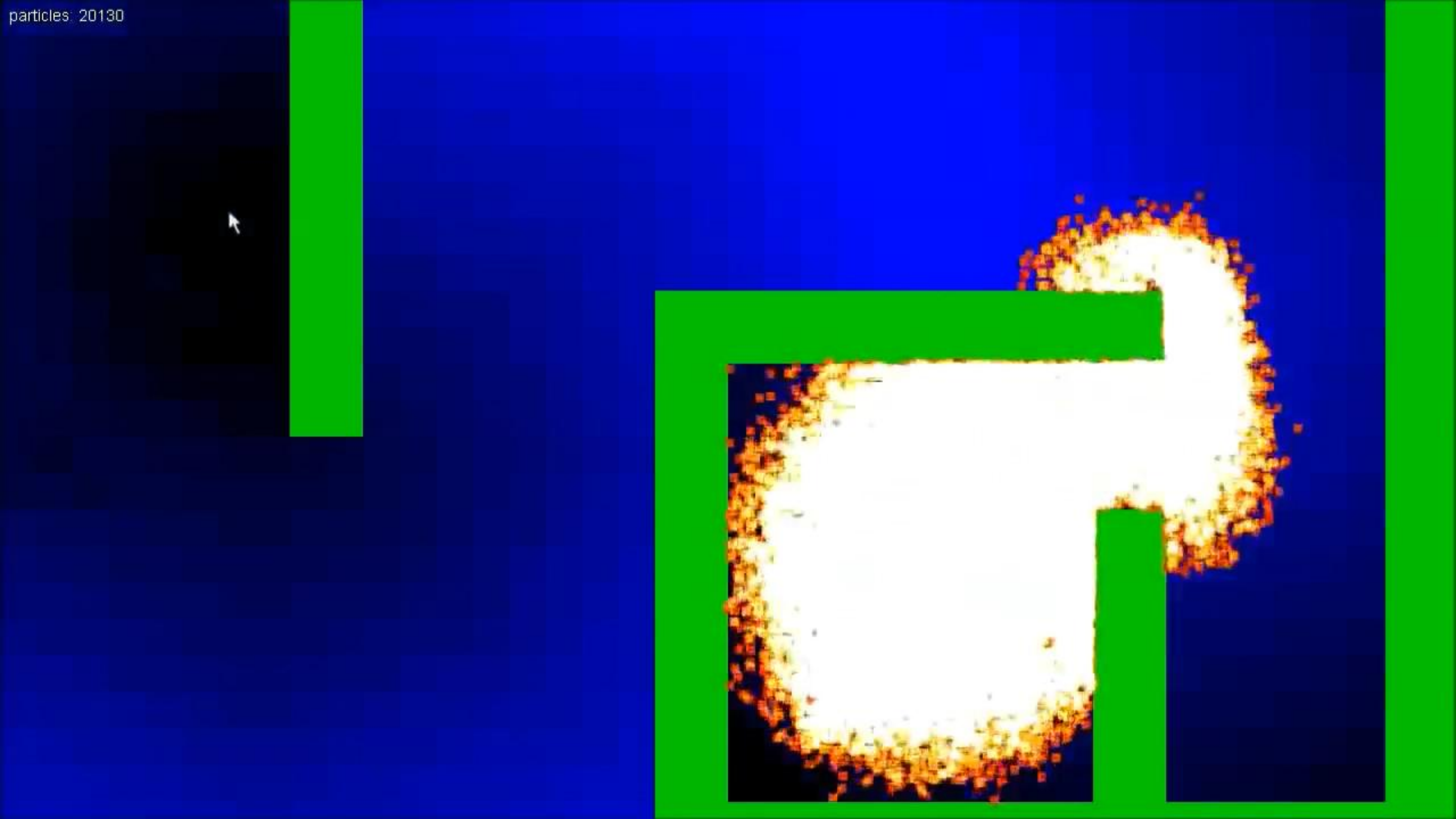
Step 1
Distance Map



Step 2
Point to Lowest




60

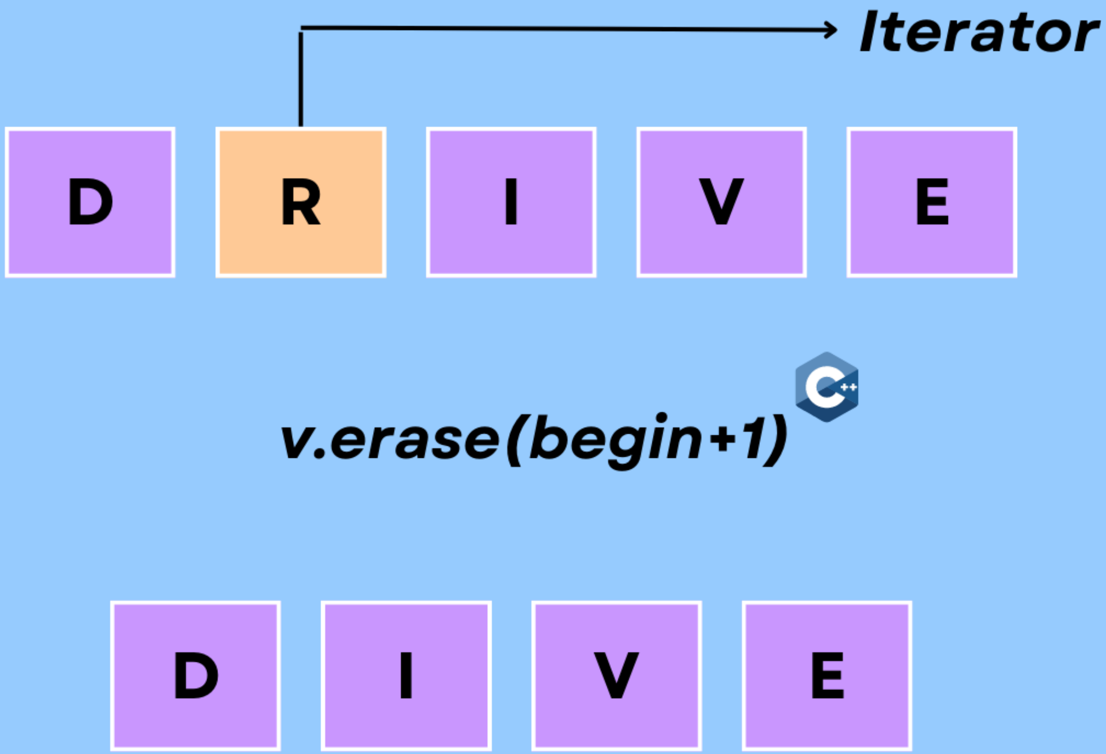

particles: 20130

Vector Fields

- Last 4 Slides:
  - https://gamedevelopment.tutsplus.com/tutorials/understanding-goal-based-vector-field-pathfinding--gamedev-9007

BFS Performance Notes

- Implementing BFS iteratively with a Queue results in poor performance
- Array-based queues need to left shift all elements to pop from the front, which can be very slow in practice

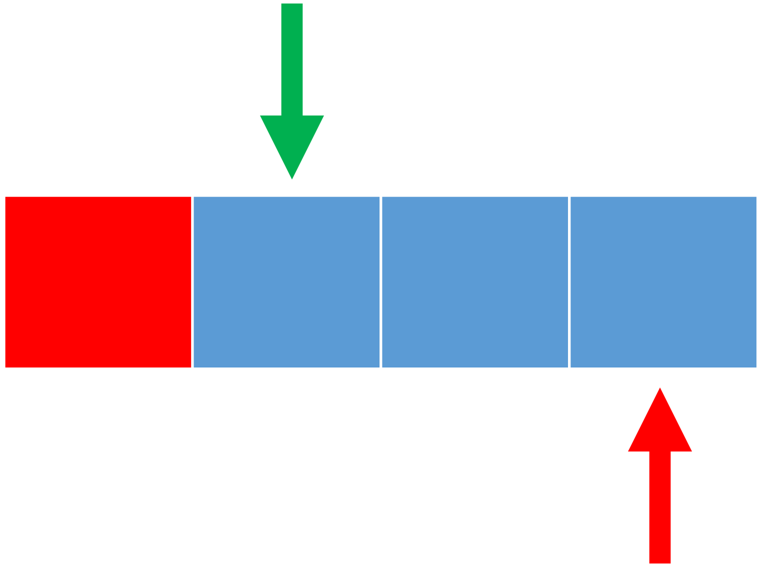

Iterator
D
R
I
V
E
v.erase(begin+1)
D
I
V
E

BFS Optimizations (Avoid Queue)

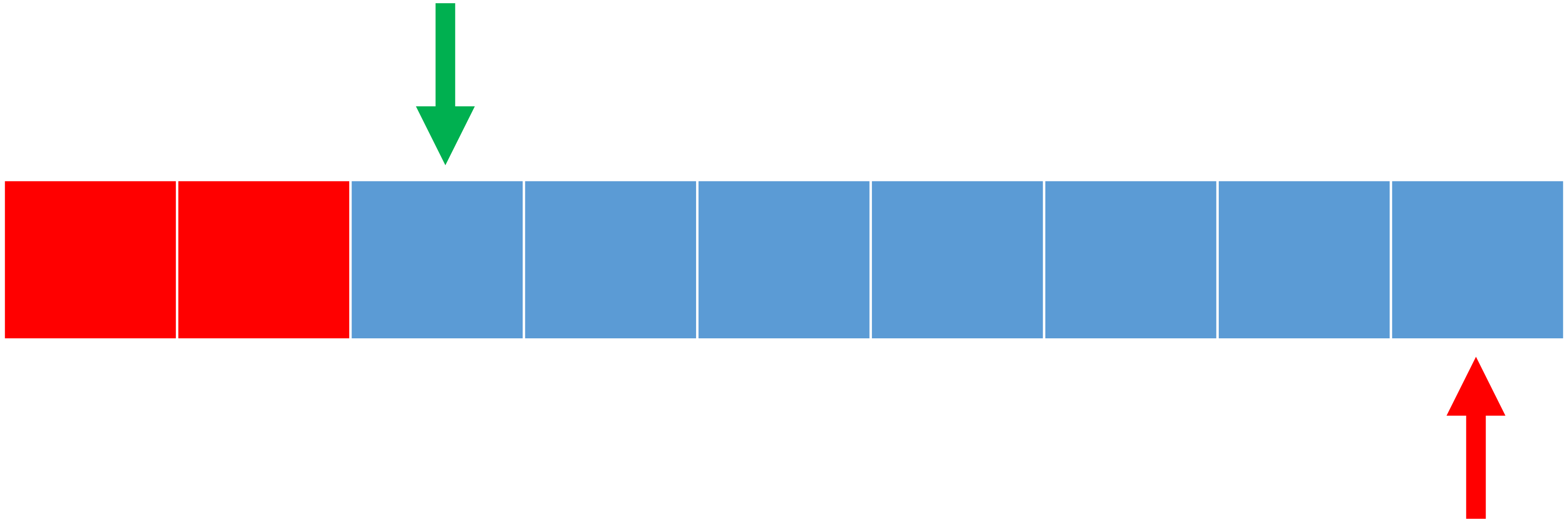
- We will simulate a Queue using an array
- To insert into the Queue, push the state into the back of the array as normal
- Instead of removing from the front of the Queue, simply advance an index pointer
- Instead of checking if the Queue is empty, check if the pointer is pointing to the end

BFS Vector as Queue

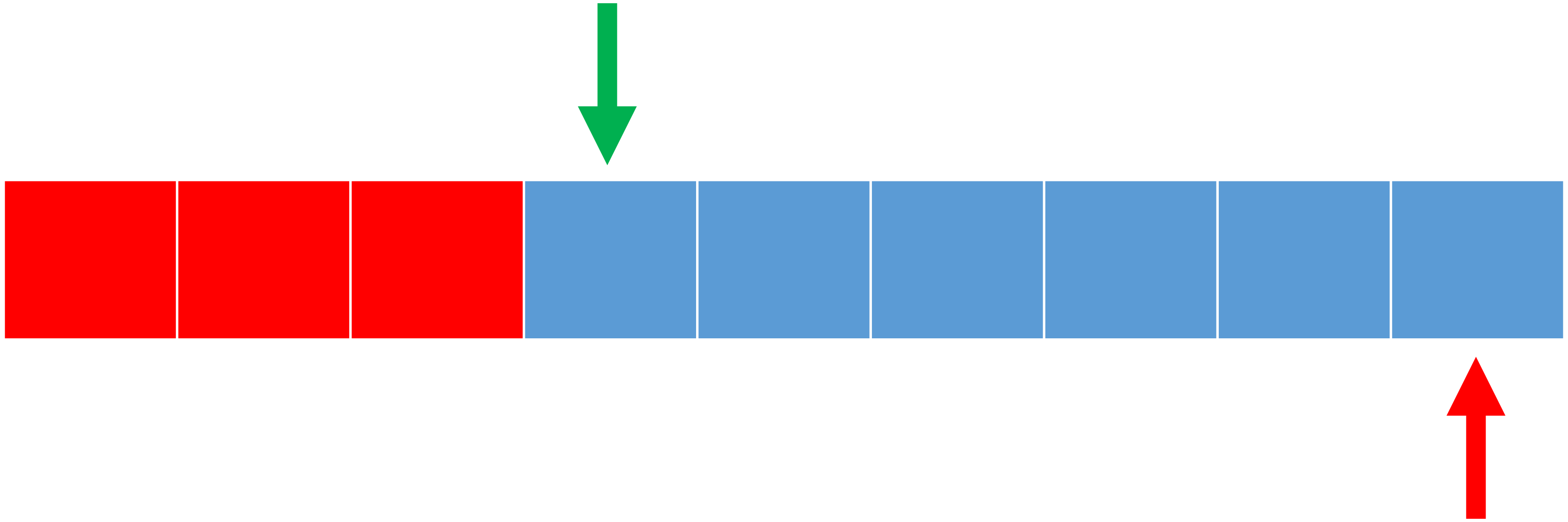
BFS Vector as Queue



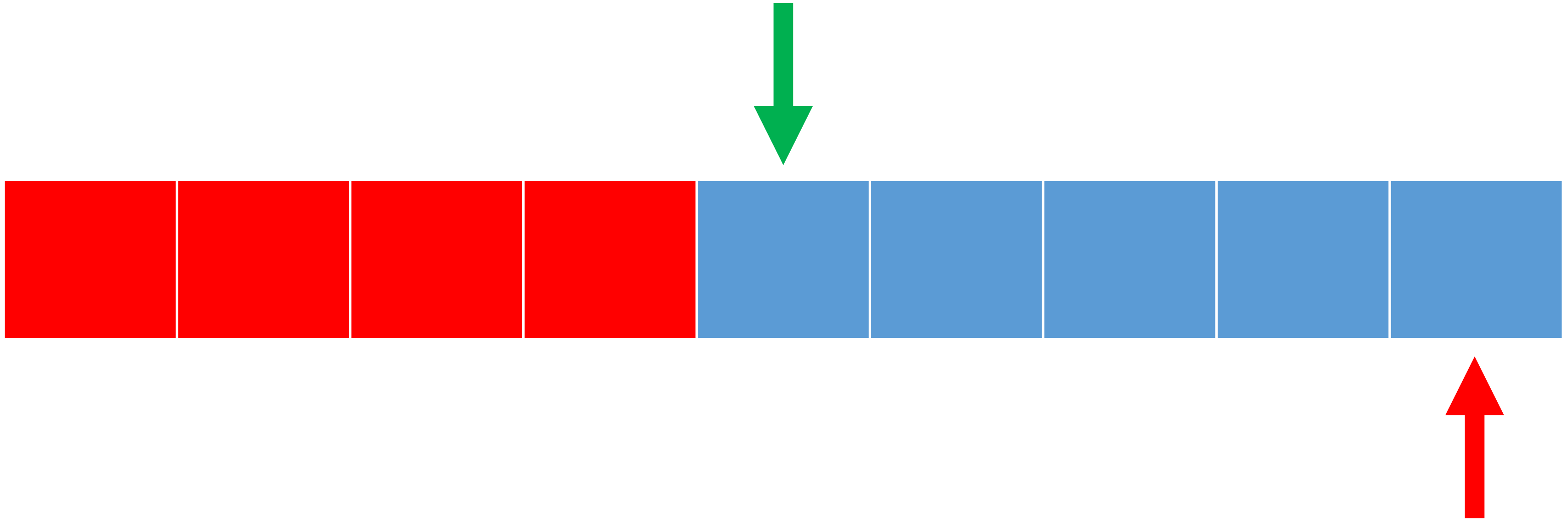
BFS Vector as Queue



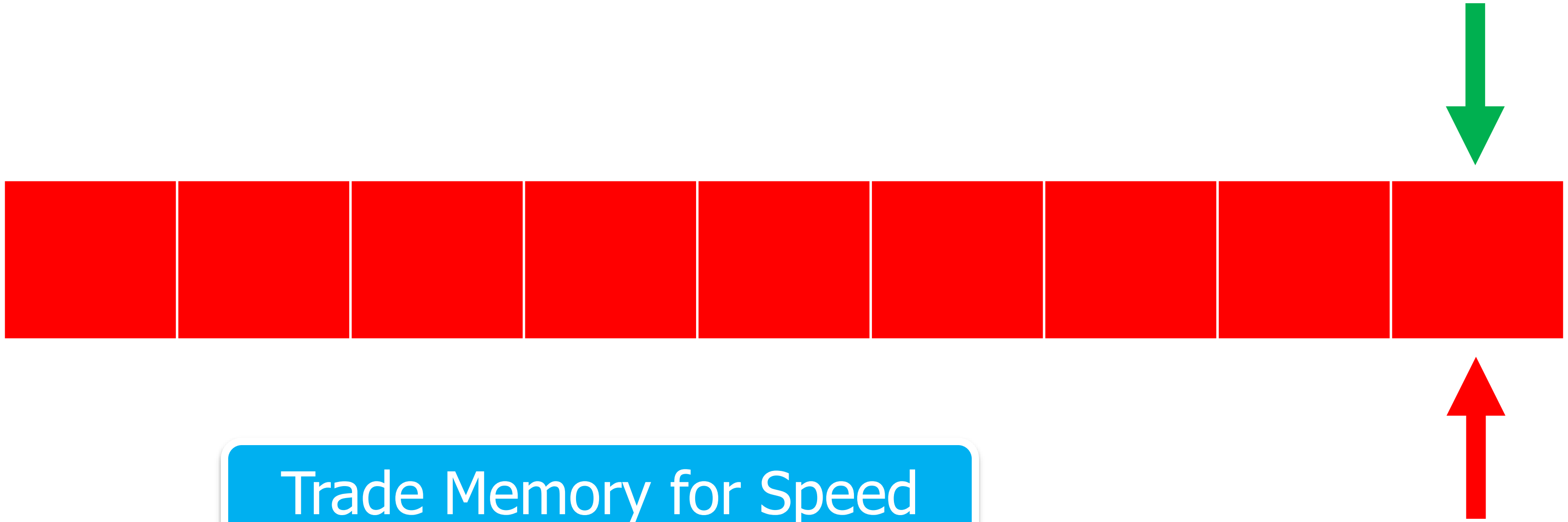
BFS Vector as Queue



BFS Vector as Queue



BFS Vector as Queue

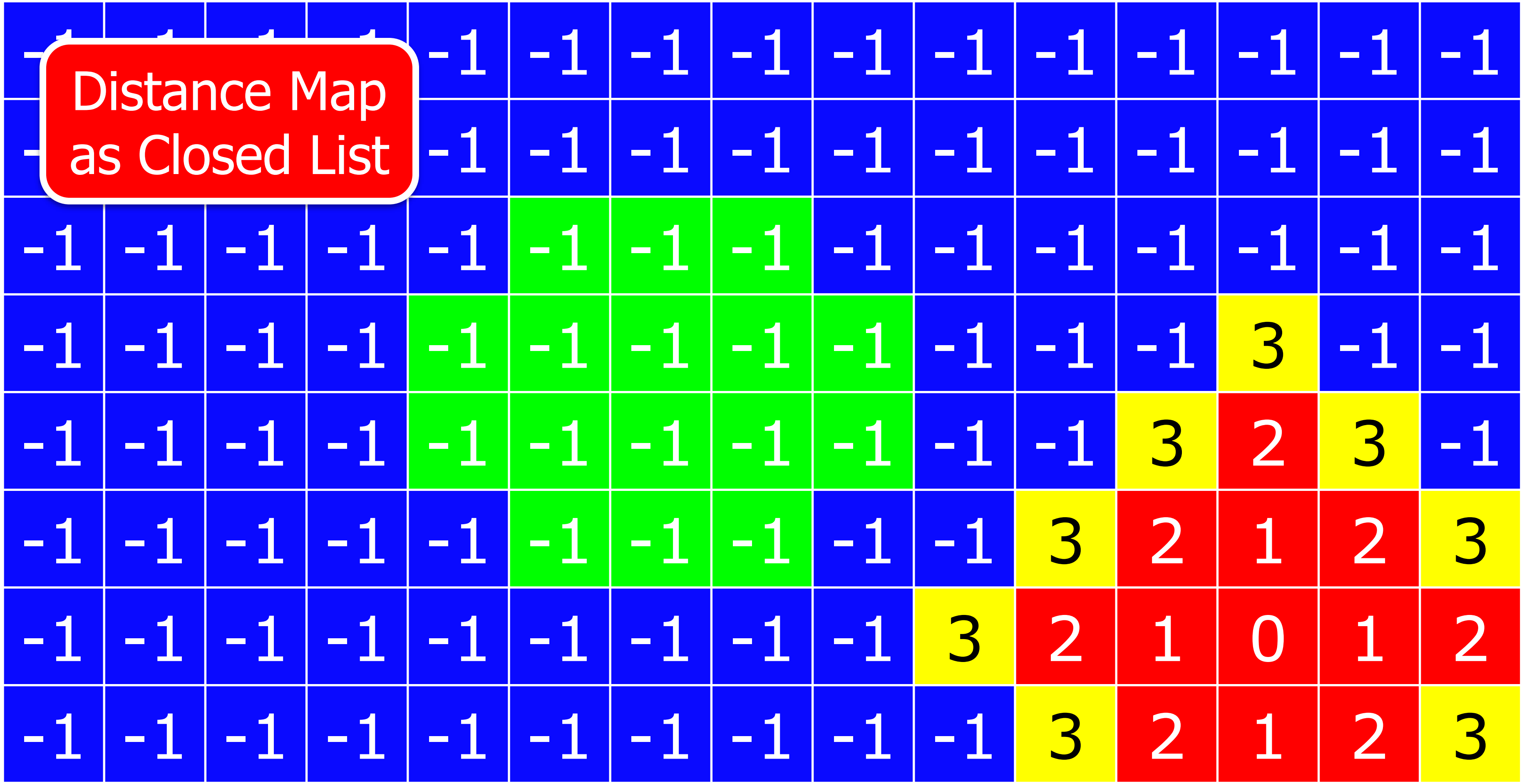


Trade Memory for Speed

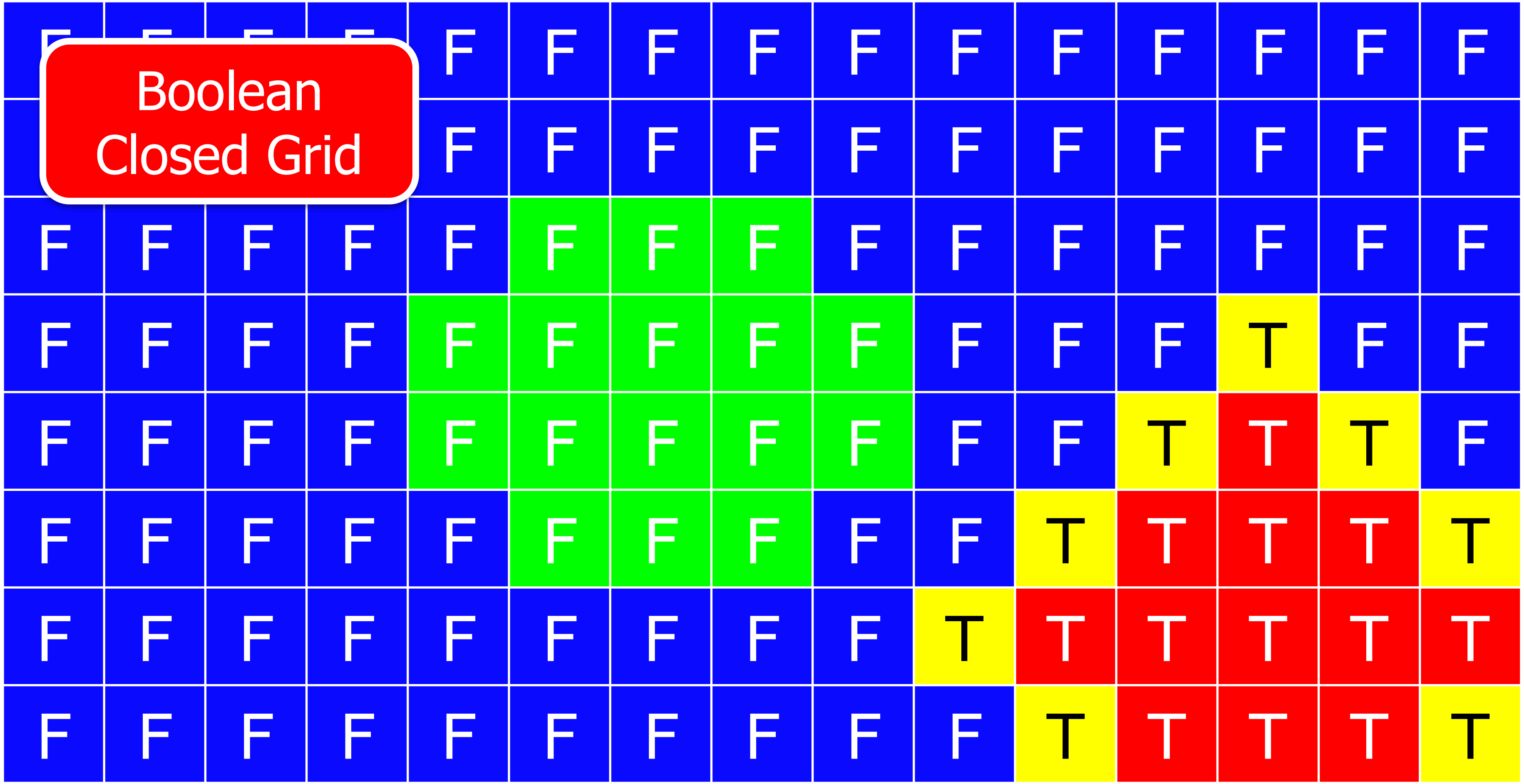
BFS Optimizations (Closed List)

- We can use the Distance map's initial values as a constant-time closed list query in lieu of a set
- Initially, we will assign the distance map cells value of something unique (ex: -1)
- Any state which has not been visited still has -1
- To query if a state has already been seen by the BFS, check if the distance map is not -1
- If you can't do this, use a separate Boolean grid

Distance Map as Closed List



-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-				-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	3	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	3	2	3	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	3	2	1	2	3
-1	-1	-1	-1	-1	-1	-1	-1	-1	3	2	1	0	1	2
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	3	2	1	2	3



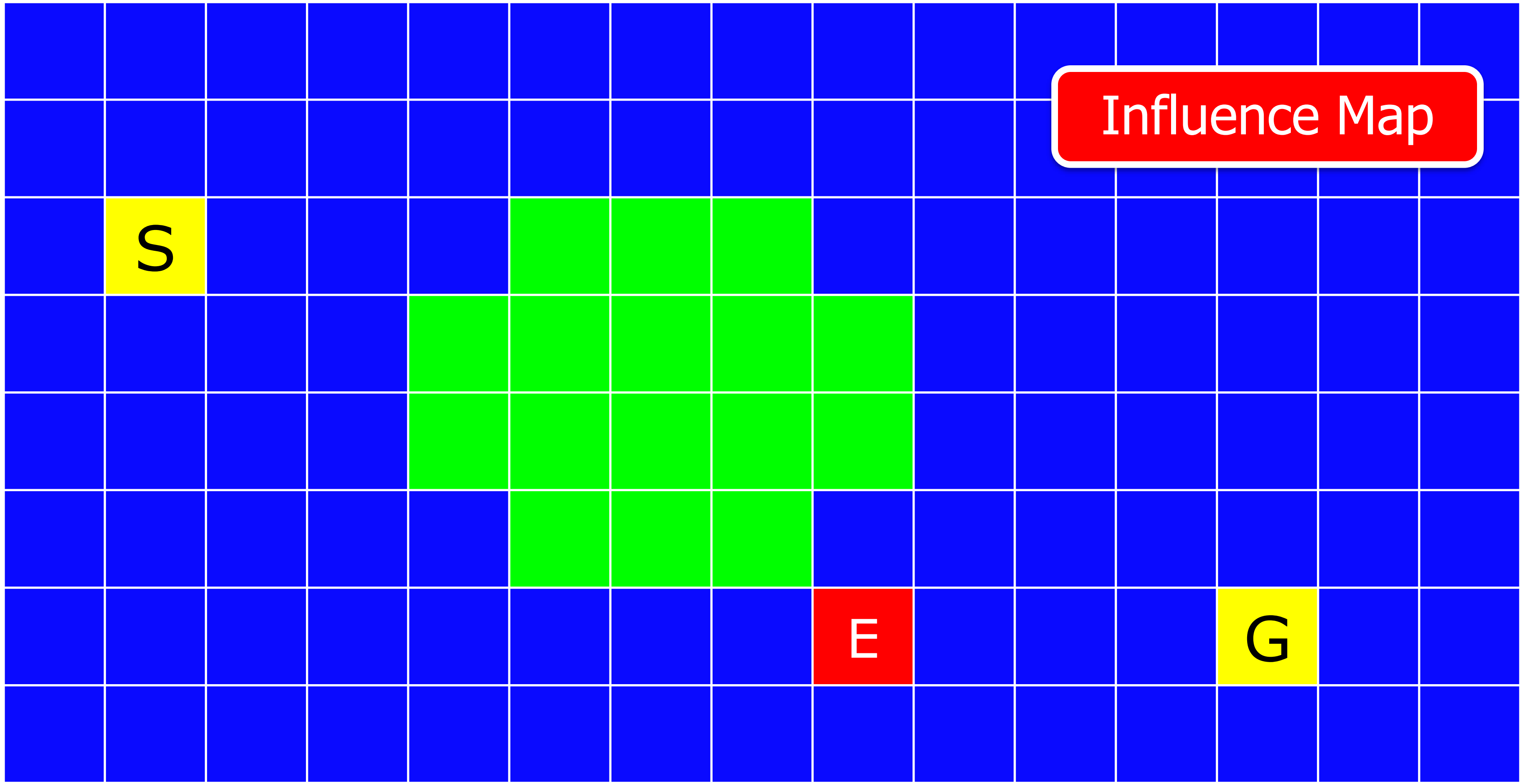
Boolean Closed Grid

F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
				F	F	F	F	F	F	F	F	F	F	F
F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
F	F	F	F	F	F	F	F	F	F	F	F	T	F	F
F	F	F	F	F	F	F	F	F	F	F	T	T	T	F
F	F	F	F	F	F	F	F	F	F	T	T	T	T	T
F	F	F	F	F	F	F	F	F	T	T	T	T	T	T
F	F	F	F	F	F	F	F	F	F	T	T	T	T	T

Influence Maps

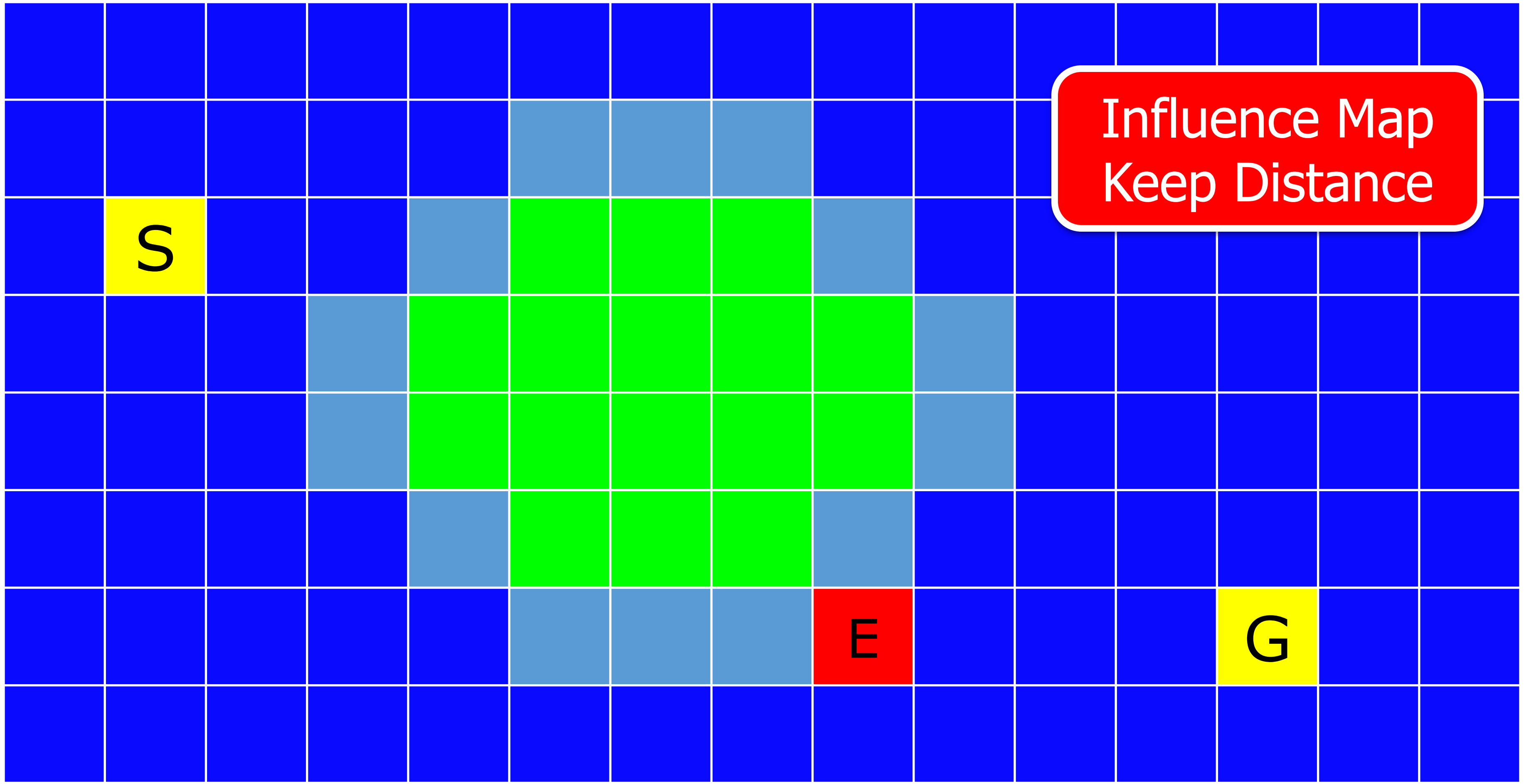
- Influence maps typically incorporate other logic than simple distance to goal
  - Game desire has 'influence' on movement
- Can be used when it may be difficult to construct an objective function for other algorithms
- Some examples of influence include:
  - Keep distance from obstacles / enemies
  - Avoid attack damage from enemies
  - Any combination of things

Influence Map

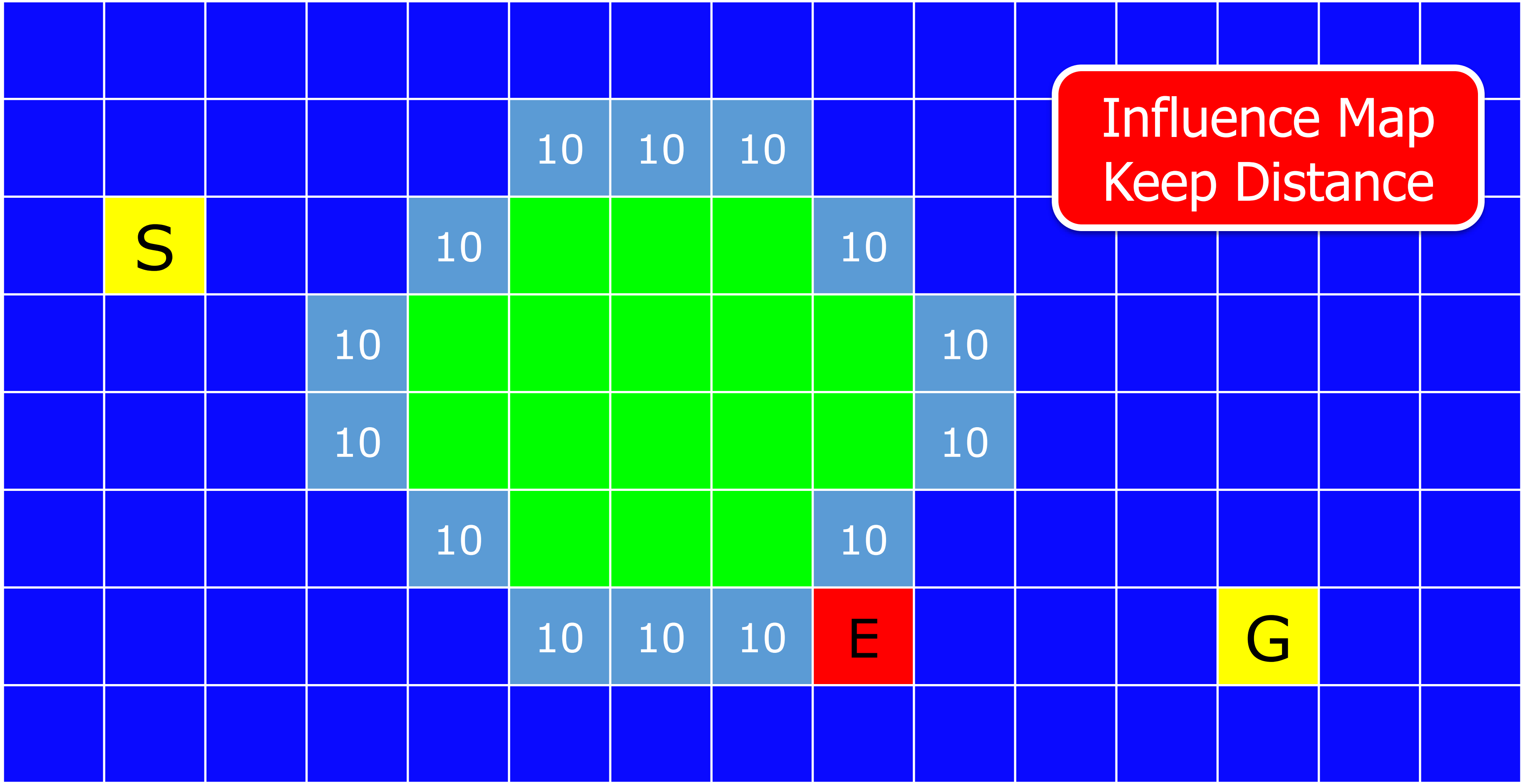


Influence Map
Avoid Enemy

18	17	16	15	14	13	12	11	10	9	8	7	6	7	8
17	16	15	14	13	12	11	10	9	8					
18	S	16	15	14	🟩	🟩	🟩	8	7	6	5	4	5	6
17	16	15	14	🟩	🟩	🟩	🟩	🟩	6	5	4	3	4	5
16	15	14	13	🟩	🟩	🟩	🟩	🟩	5	4	3	2	3	4
15	14	13	12	11	🟩	🟩	🟩	5	4	3	2	1	2	3
14	13	12	11	10	9	8	7	E	3	2	1	G	1	2
13	12	11	10	9	8	7	6	5	4	3	2	1	2	3


Influence Map
Keep Distance
S
E
G

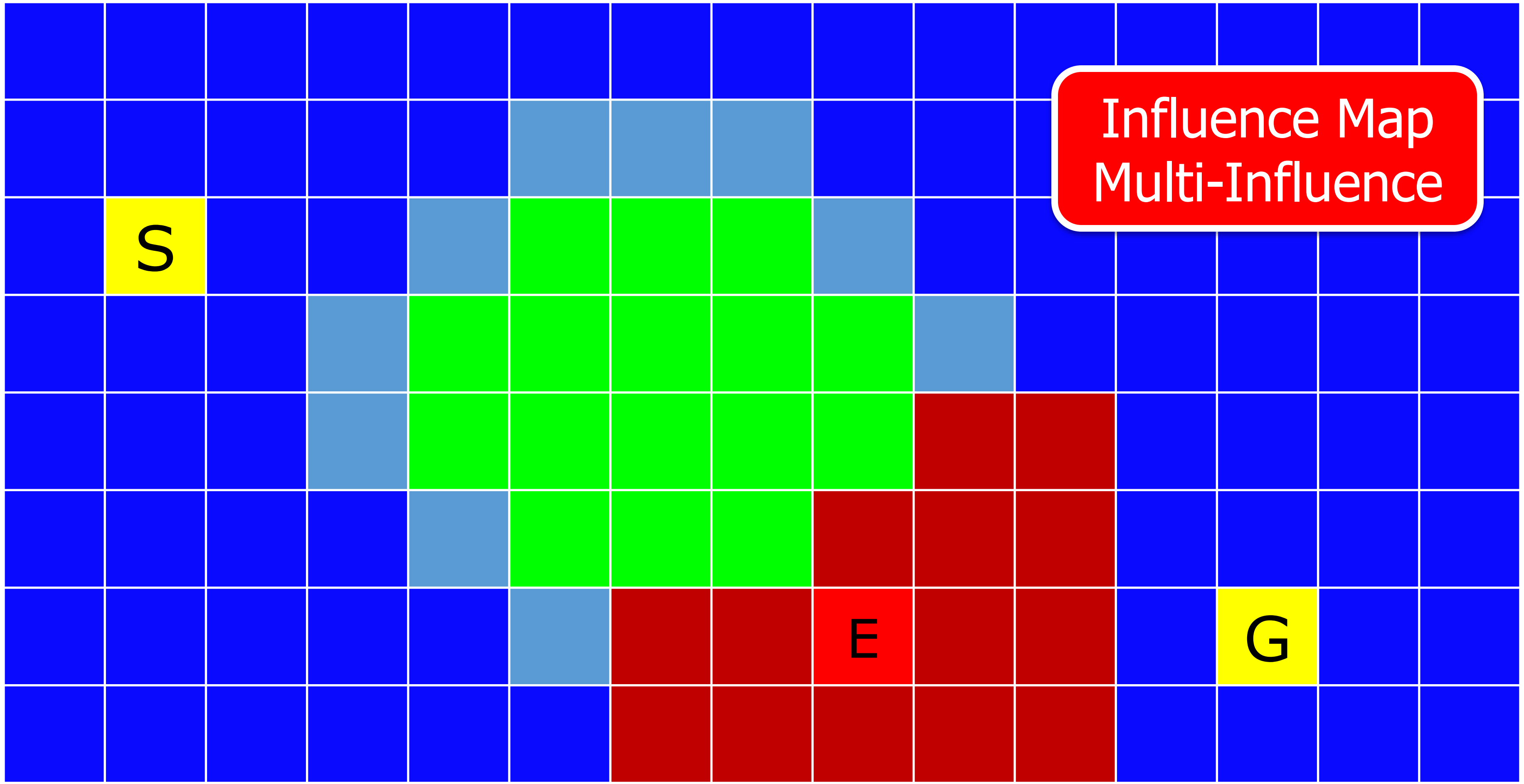
Influence Map
Keep Distance



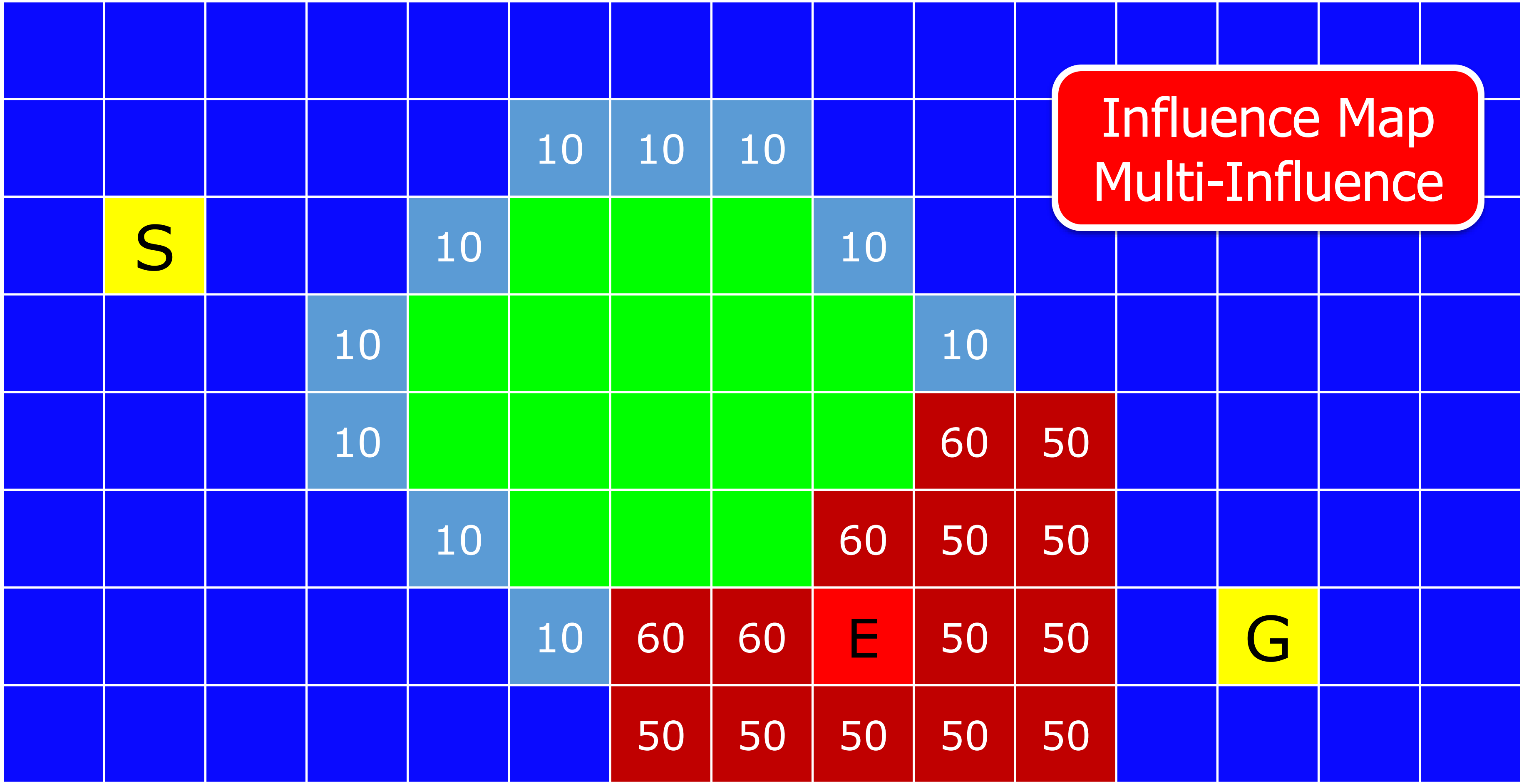
Influence Map Keep Distance

18	17	16	15	14	13	12	11	10	9	8	7	6	7	8
17	16	15	14	13	24	23	20	9	8					
18	S	16	15	24	🟩	🟩	🟩	18	7	6	5	4	5	6
17	16	15	26	🟩	🟩	🟩	🟩	🟩	16	5	4	3	4	5
16	15	14	23	🟩	🟩	🟩	🟩	🟩	15	4	3	2	3	4
15	14	13	12	21	🟩	🟩	🟩	15	4	3	2	1	2	3
14	13	12	11	10	19	18	17	E	3	2	1	G	1	2
13	12	11	10	9	8	7	6	5	4	3	2	1	2	3

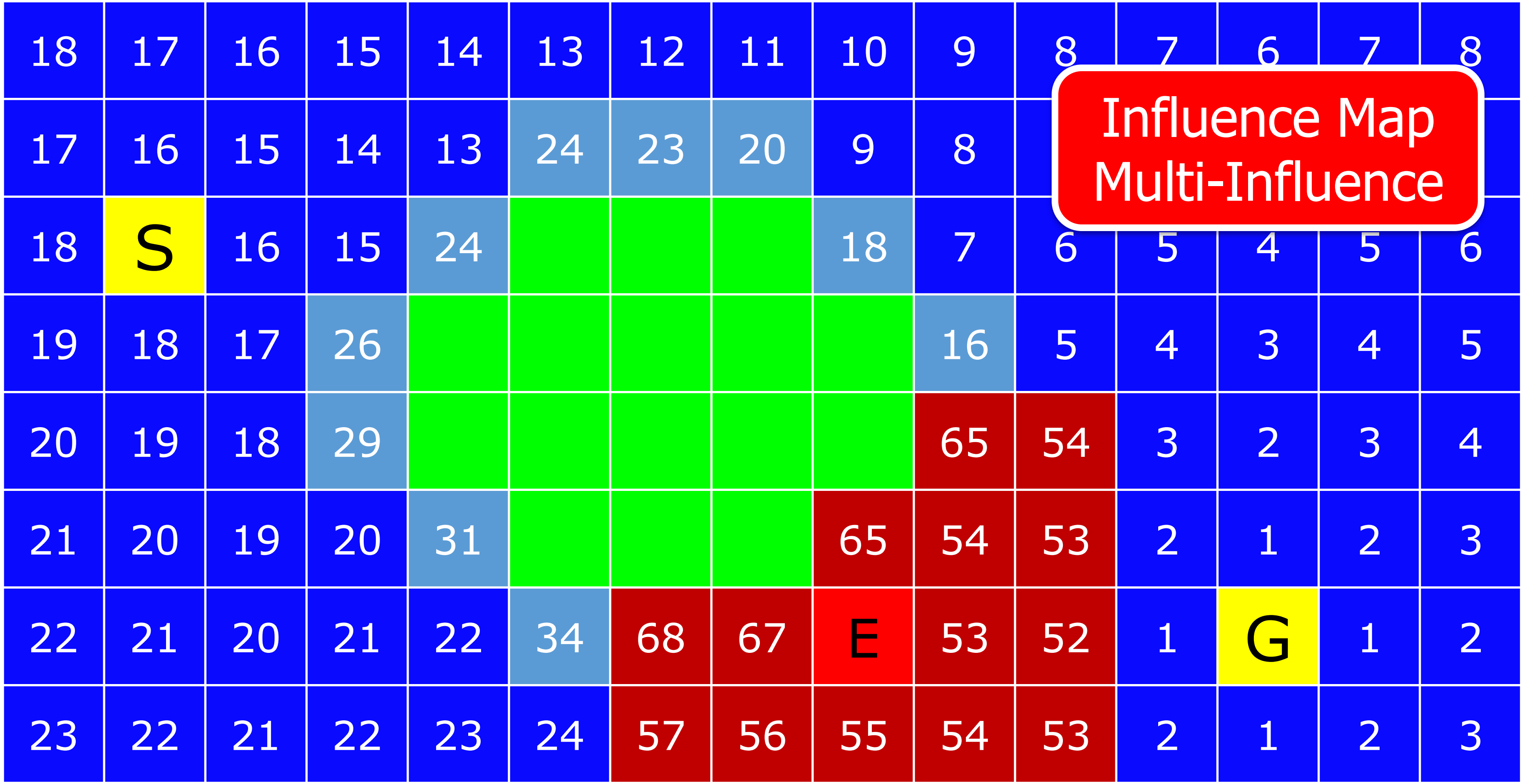
Influence Map Multi-Influence



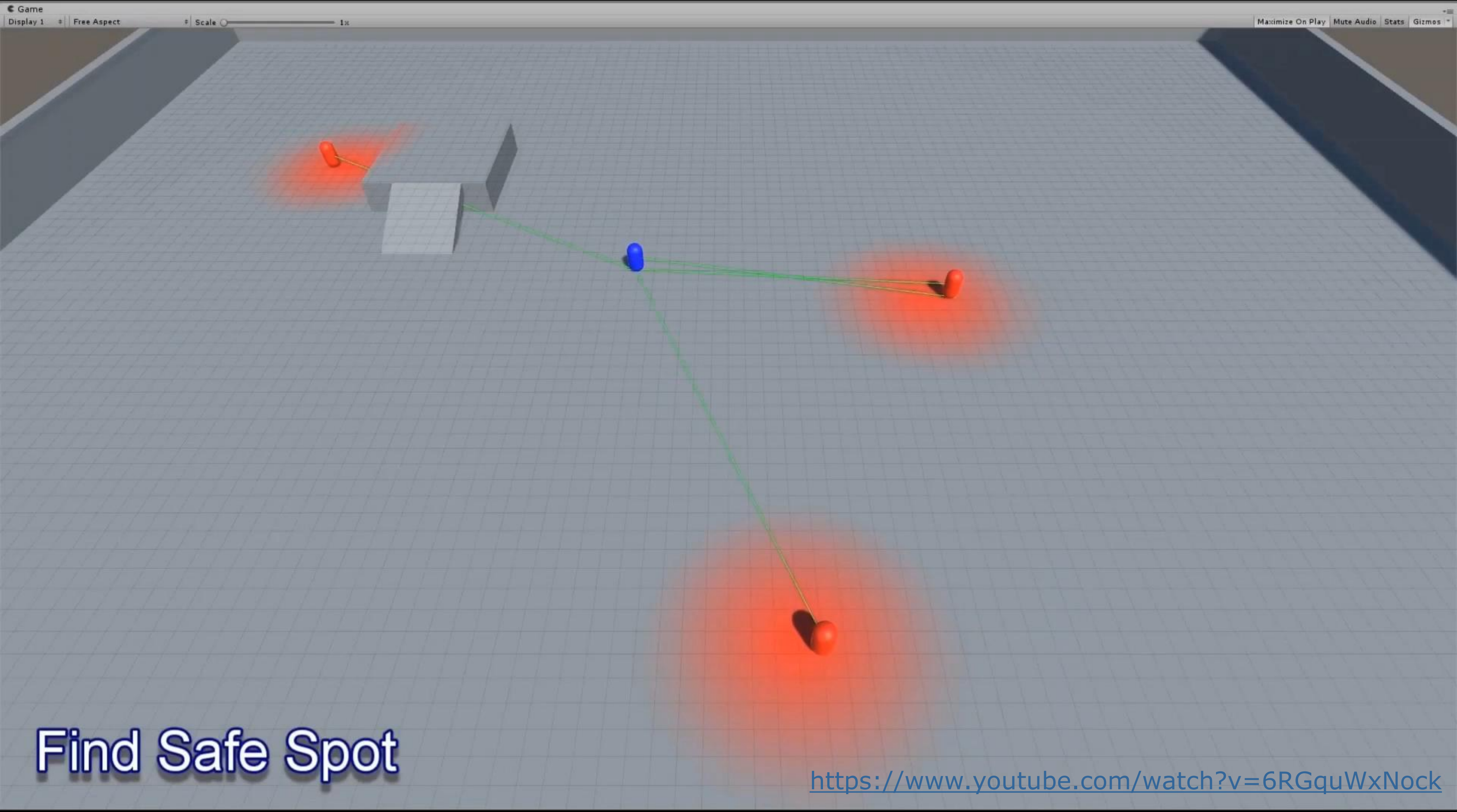
Influence Map
Multi-Influence



Influence Map Multi-Influence

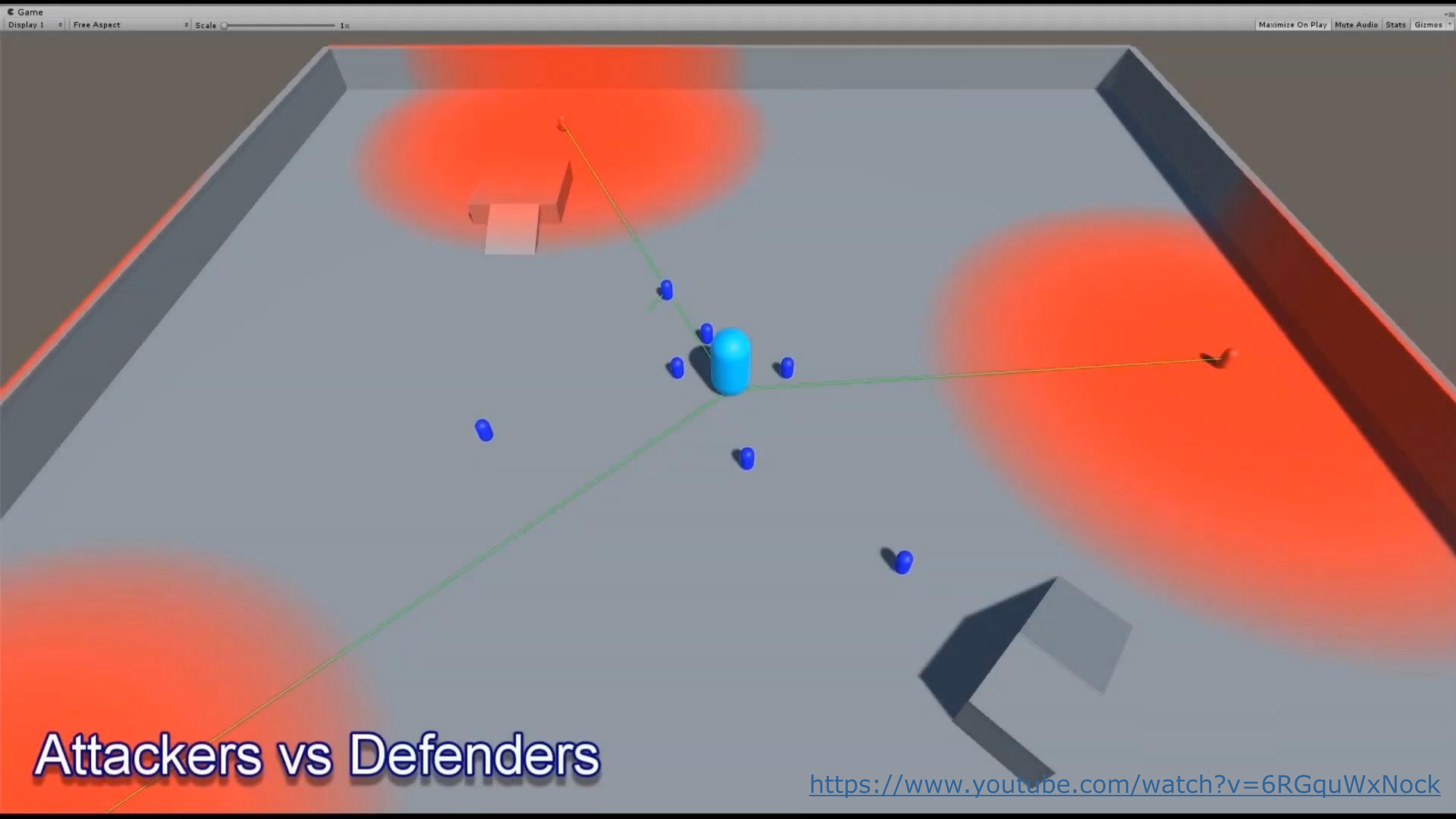


18	17	16	15	14	13	12	11	10	9	8	7	6	7	8
17	16	15	14	13	24	23	20	9	8					
18	S	16	15	24	🟩	🟩	🟩	18	7	6	5	4	5	6
19	18	17	26	🟩	🟩	🟩	🟩	🟩	16	5	4	3	4	5
20	19	18	29	🟩	🟩	🟩	🟩	🟩	65	54	3	2	3	4
21	20	19	20	31	🟩	🟩	🟩	65	54	53	2	1	2	3
22	21	20	21	22	34	68	67	E	53	52	1	G	1	2
23	22	21	22	23	24	57	56	55	54	53	2	1	2	3



Find Safe Spot

https://www.youtube.com/watch?v=6RGquWxNock



Attackers vs Defenders

https://www.youtube.com/watch?v=6RGquWxNock

Vector Field Demo