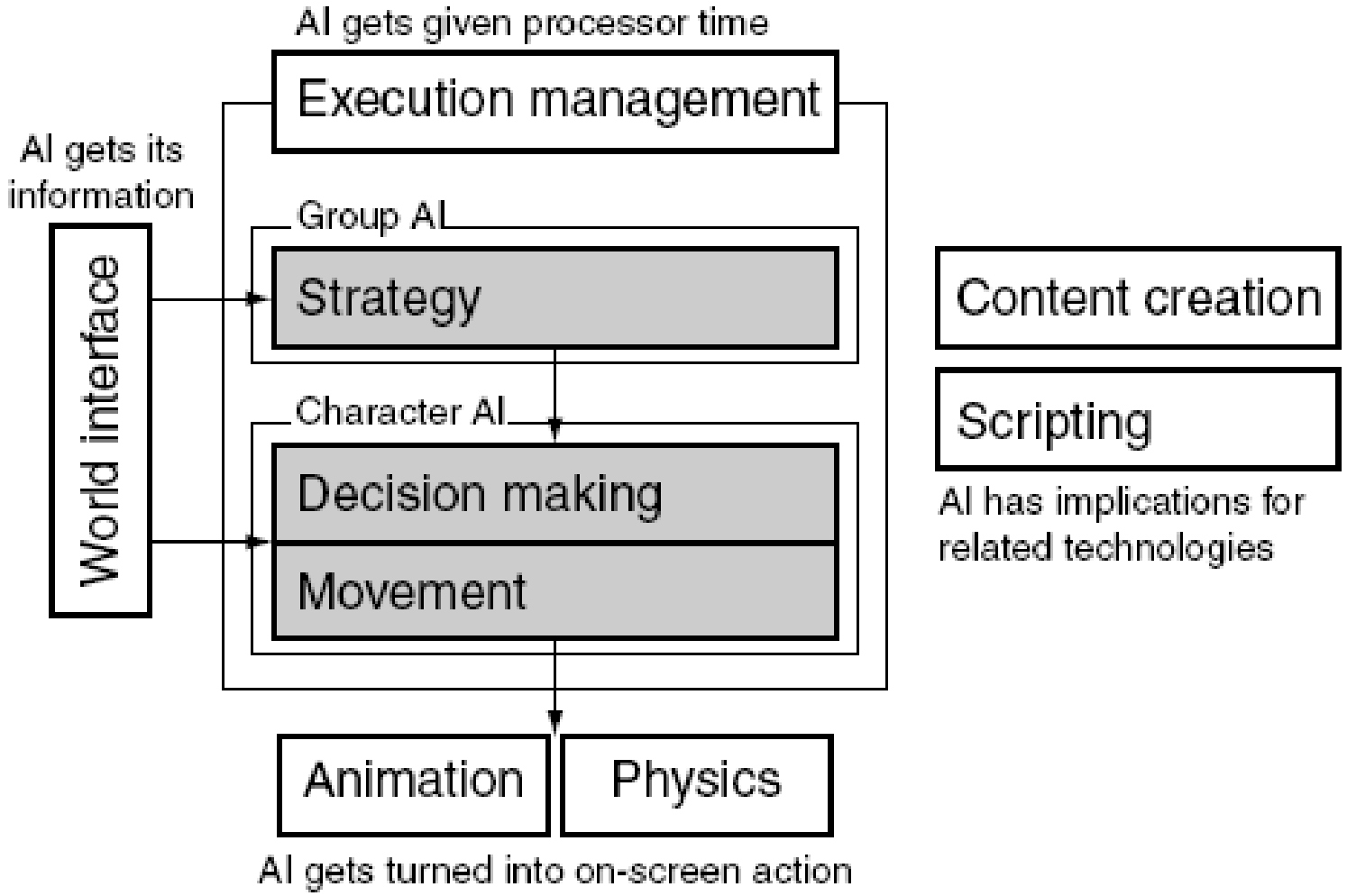
Strategy:
Strategic Locations and Paths, Coordination of Characters

Tactical and strategic AI in Games

- "One of the key fields for next five y ear$[*from "Artificial Intelligence for Games" by I. Millington & J. Funge*]

Tactical and strategic AI in Games

- Overall diagram



from "Artificial Intelligence for Games" by I. Millington & J. Funge

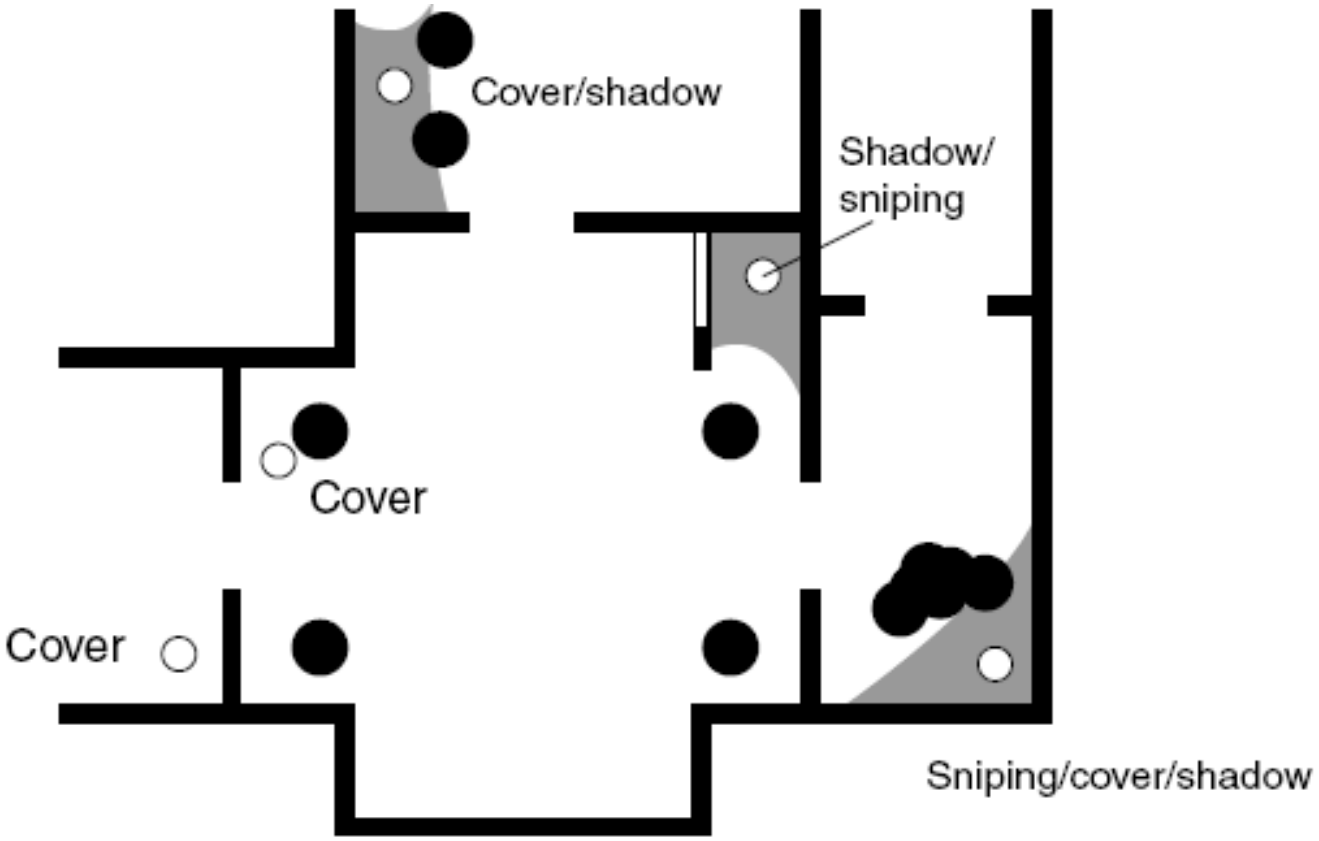
Tactical Locations (“rally points”)

- Points that have important tactical features
  - safe locations for troops to retreat to in case of defeat (commonly used in real-world military planning)
  - cover points (e.g., hiding place behind barrels, etc.)
  - sniper points
  - avoid points (e.g., exposed areas, etc.)
  - shadow points (e.g., out-of-light points)

What would be good points?

Tactical Locations (“rally points”)

- Sniping/cover/shadow points

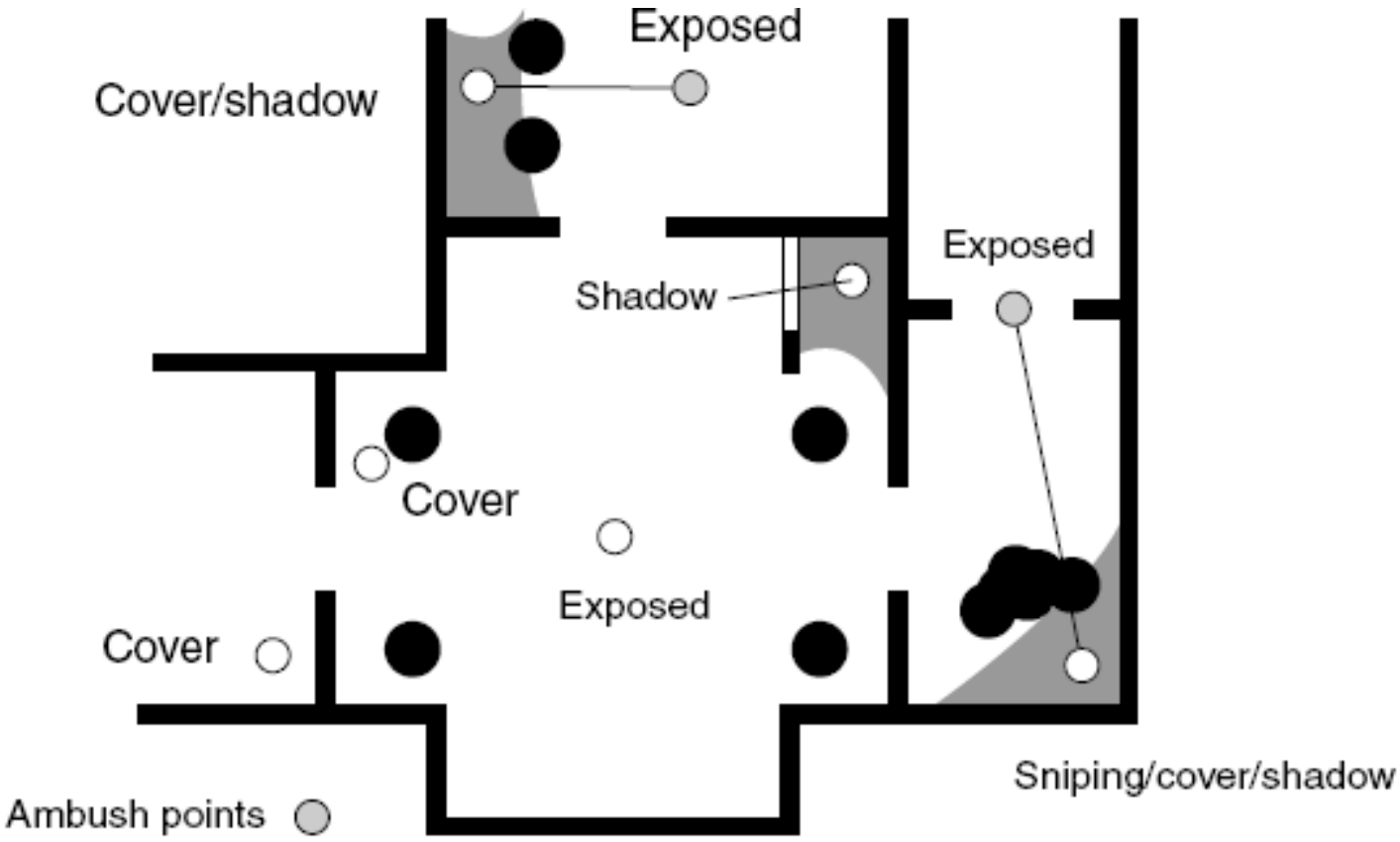


from “Artificial Intelligence for Games” by I. Millington & J. Funge

Tactical Locations ("rally points")

- Ambush points

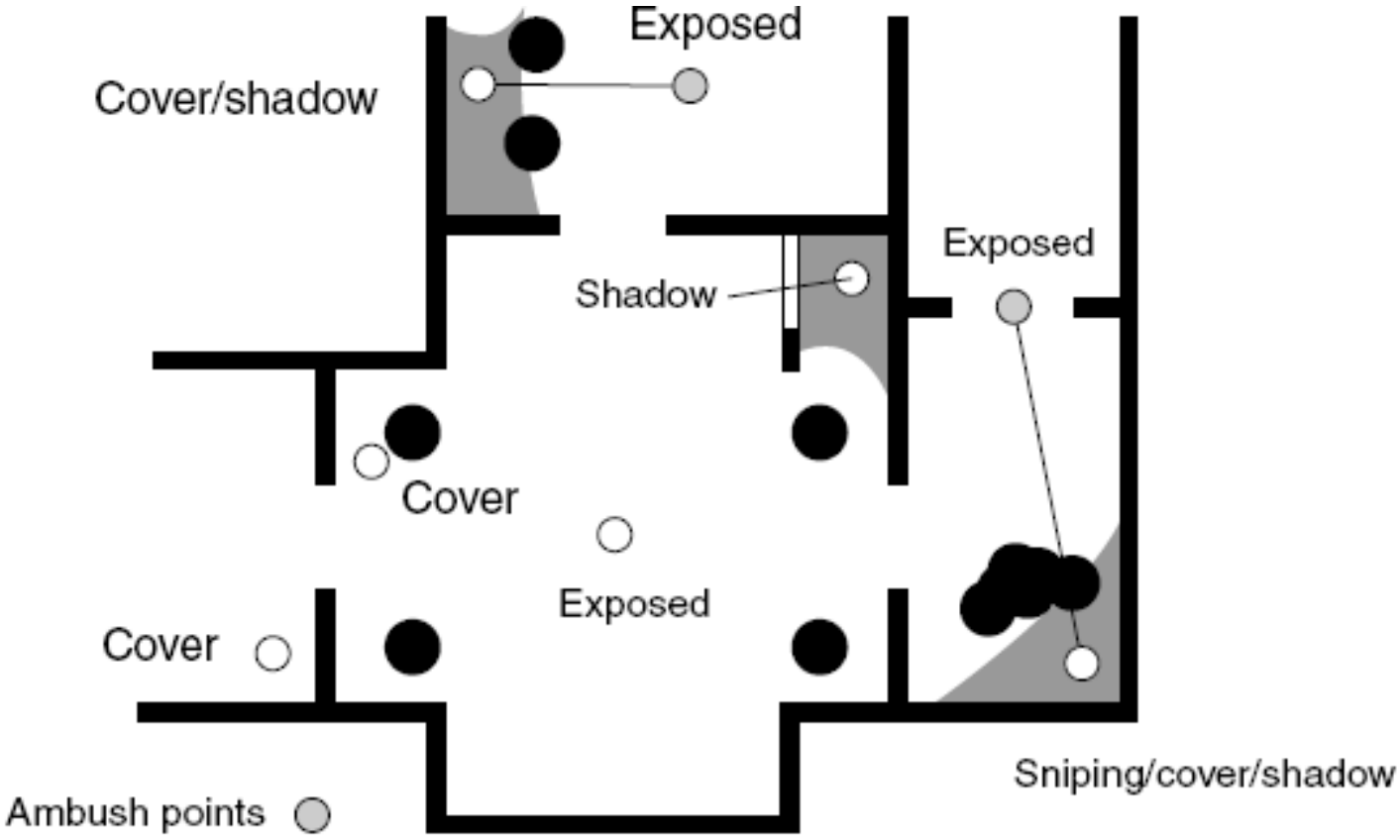
What are those points?



from "Artificial Intelligence for Games" by I. Millington & J. Funge

Tactical Locations ("rally points")

- Ambush points
  - good cover point in shadow that is close to and sees one or more exposed points

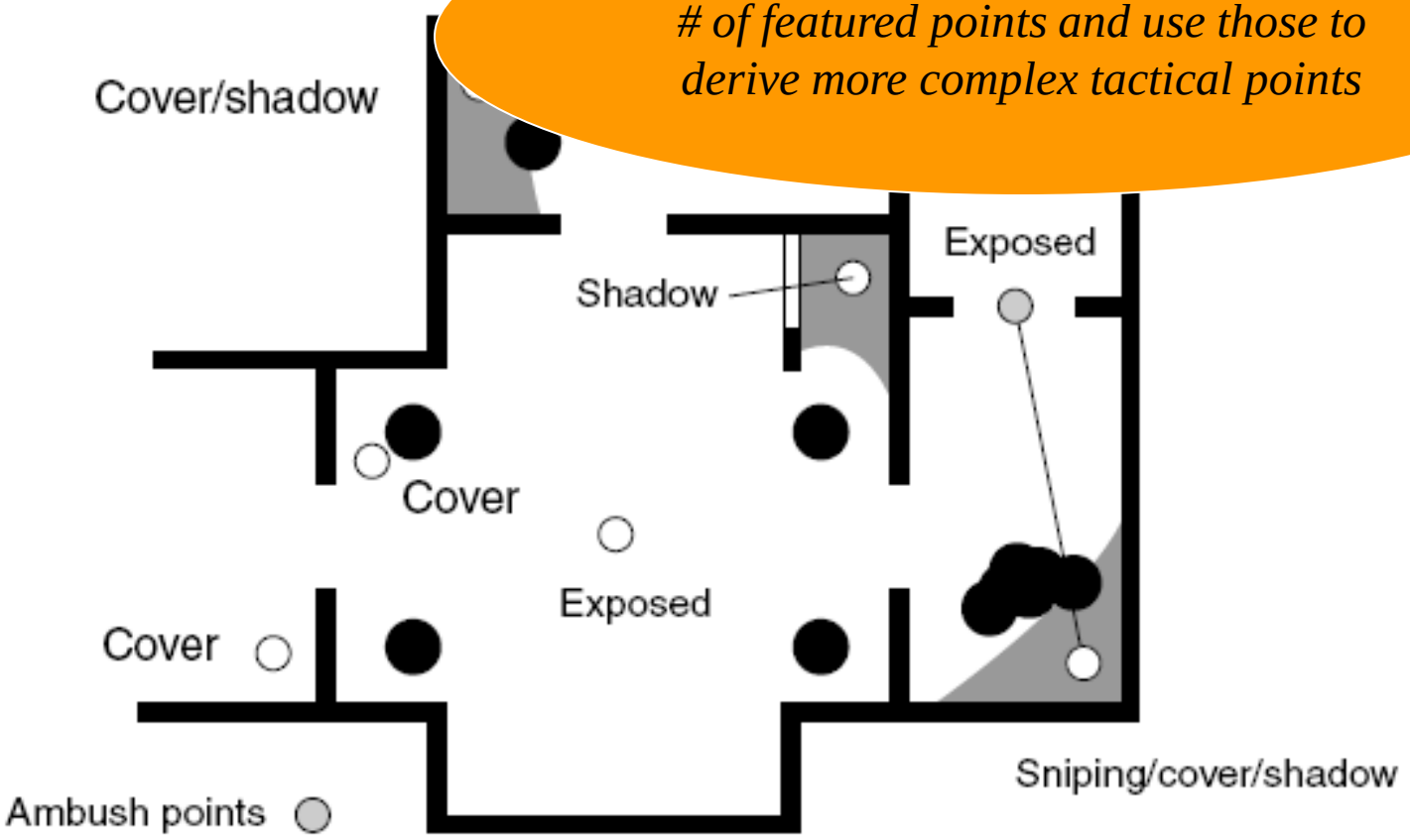


from "Artificial Intelligence for Games" by I. Millington & J. Funge

Tactical Locations ("rally points")

- Ambush points
  - good cover point in shadow that is close to and sees one or more exposed points

More generally: want to compute limited # of featured points and use those to derive more complex tactical points



from "Artificial Intelligence for Games" by I. Millington & J. Funge

Tactical Locations (“rally points”)

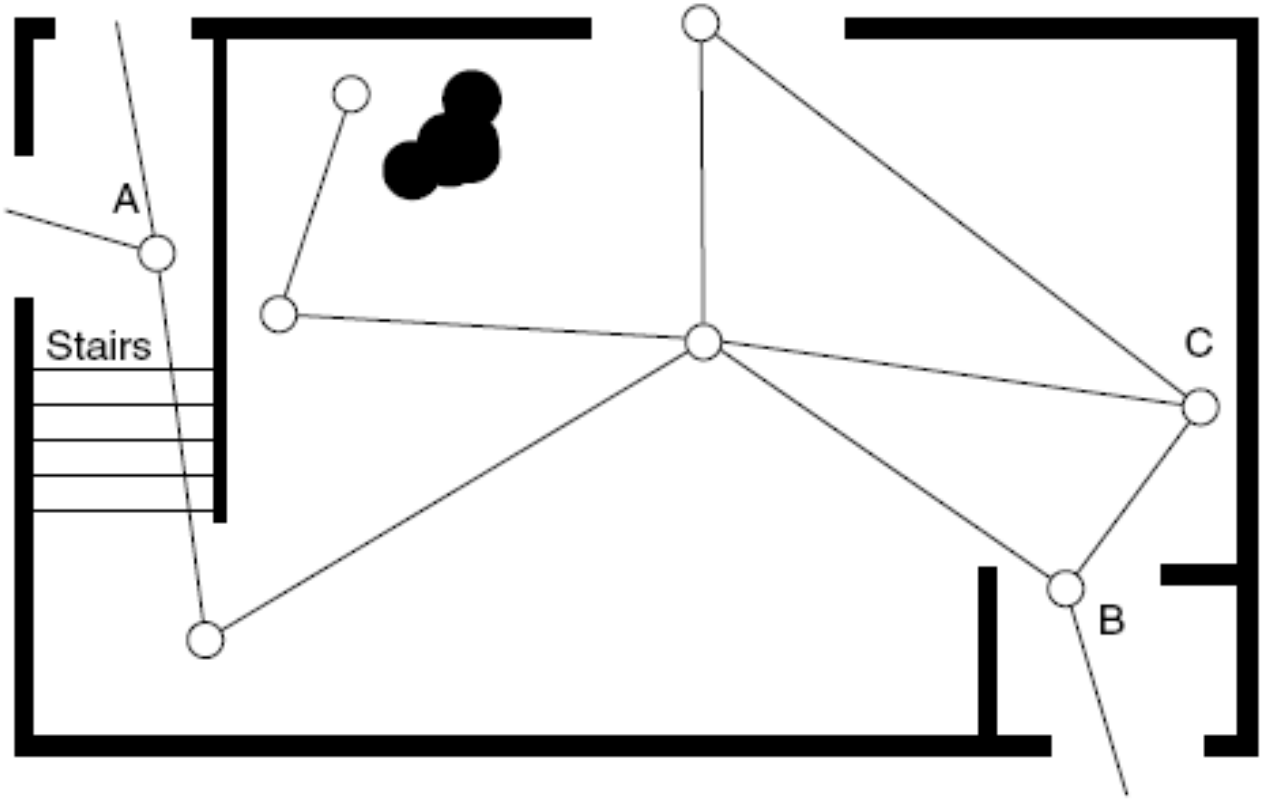
- Can be selected manually by the designer
  - very time consuming
  - requires large number of points to be stored in memory
  - can provide high-quality (interesting) behaviors

- Can be selected automatically
  - requires less time
  - lower memory requirements
  - provides less control over the resulting behavior

Tactical Locations (“rally points”)

- Waypoint graph with states defined by **Boolean** attributes

A state is given by true/false for each attribute (cover, shadow, exposure)
Two states are connected by transition if they are quickly reachable (and visible)

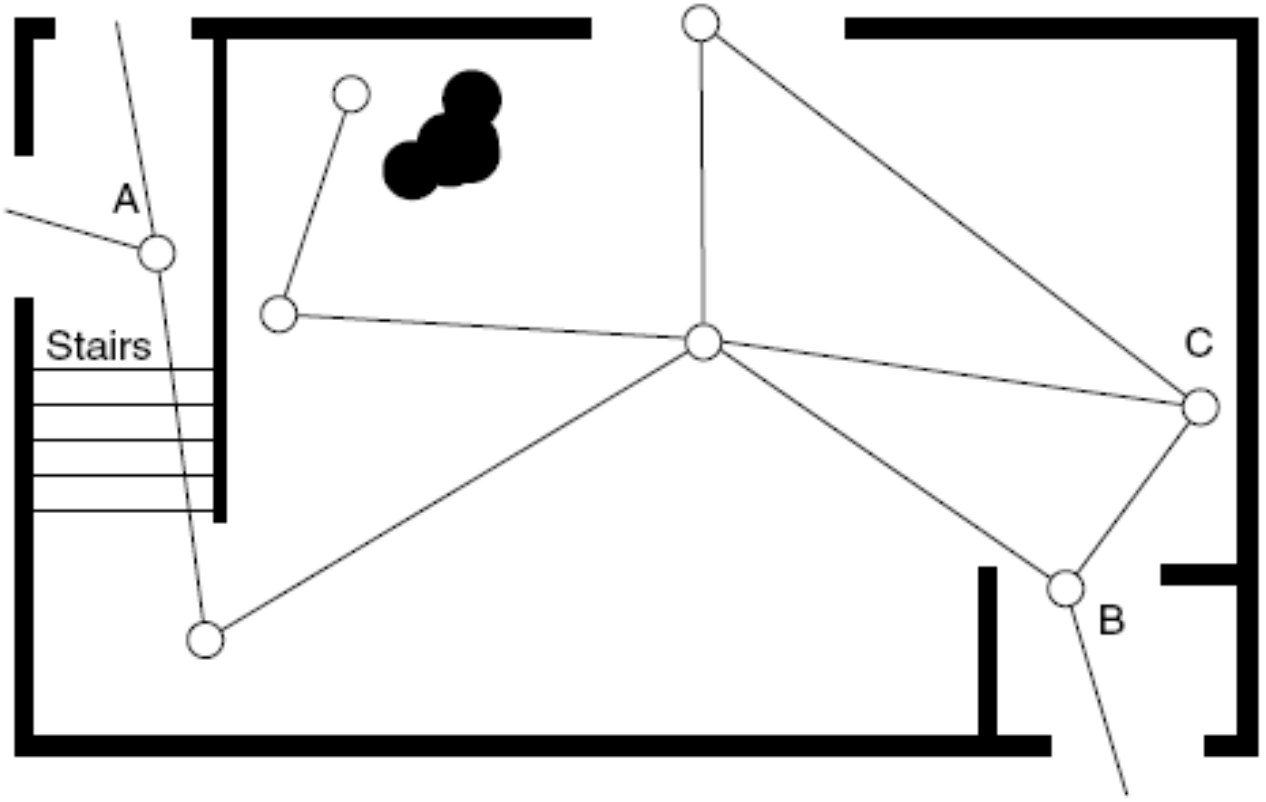


from “Artificial Intelligence for Games” by I. Millington & J. Funge

Tactical Locations (“rally points”)

- Waypoint graph with states defined by **continuous** attributes

A state is given by [0,1] for each attribute (cover, shadow, exposure)
Two states are connected by transition if they are quickly reachable (and visible)

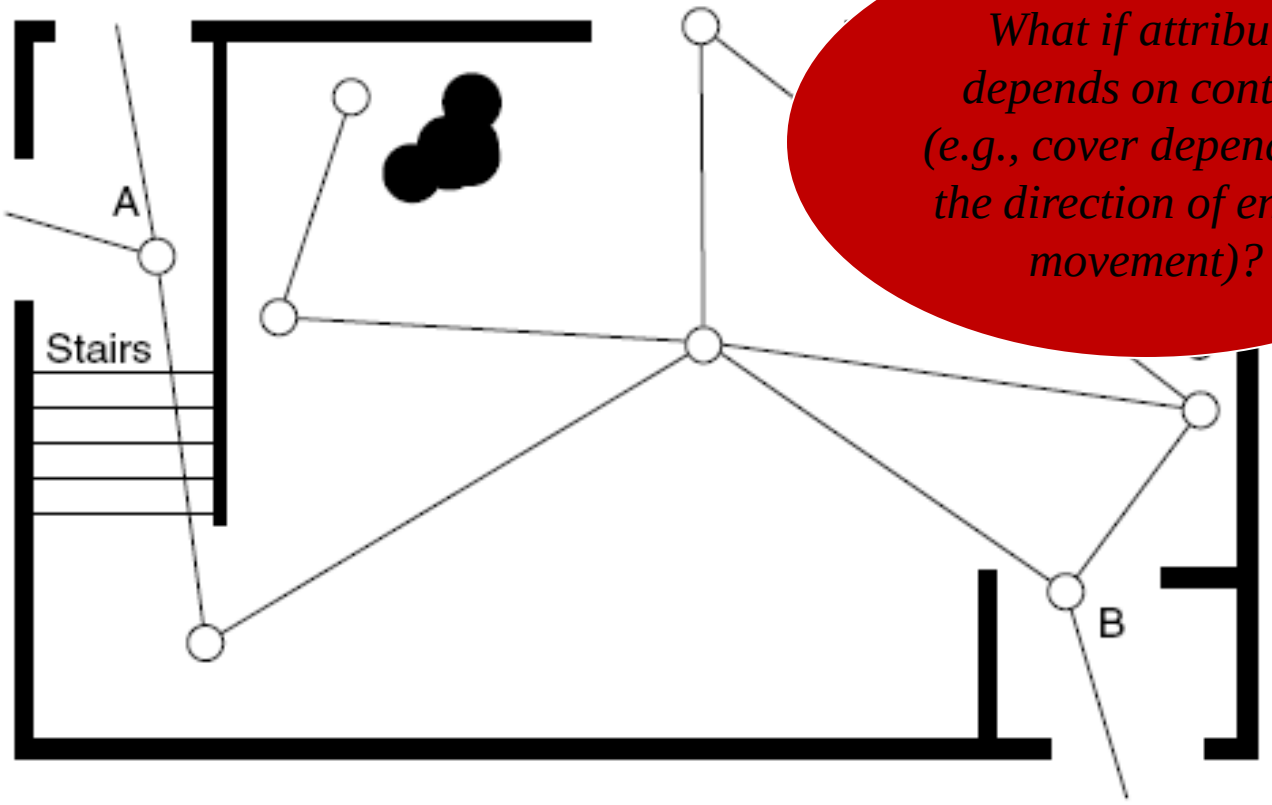


from “Artificial Intelligence for Games” by I. Millington & J. Funge

Tactical Locations ("rally points")

- Waypoint graph with states defined by **continuous** attributes

A state is given by [0,1] for each attribute (cover, shadow, exposure)
Two states are connected by transition if they are quickly reachable (and visible)



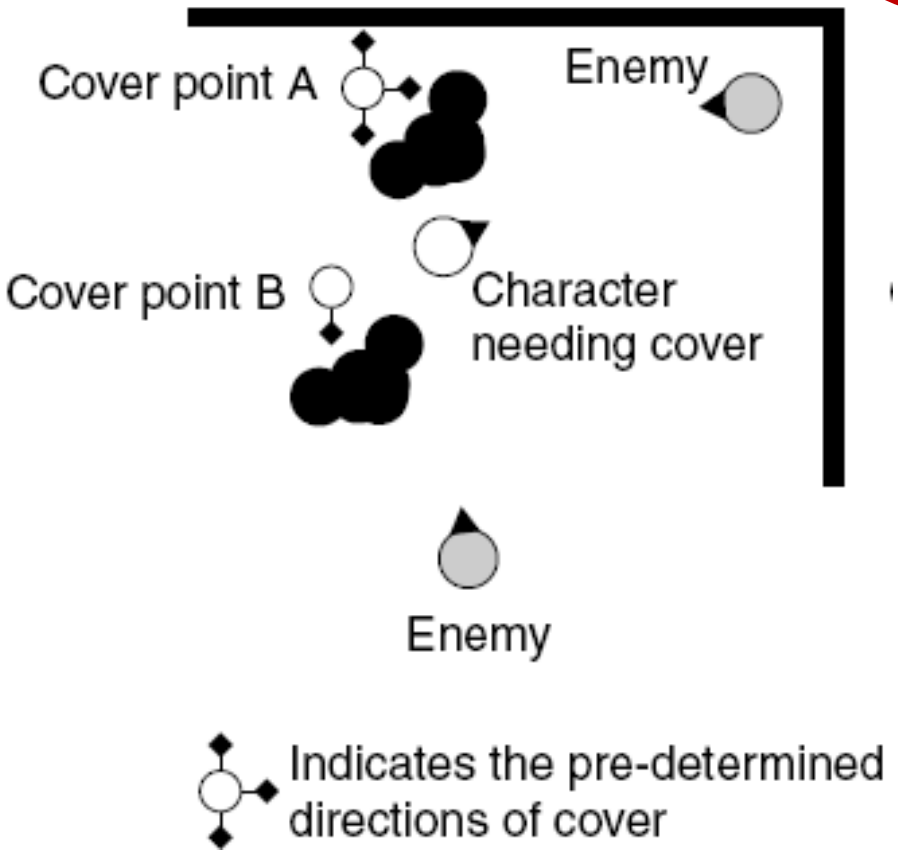
What if attribute depends on context (e.g., cover depends on the direction of enemy movement)?

from "Artificial Intelligence for Games" by I. Millington & J. Funge

Tactical Locations ("rally points")

- Attributes that are dependent on the context
  - can be pre-computed for all possible contexts (for all directions, all kinds of weapons, etc.)

Disadvantages?

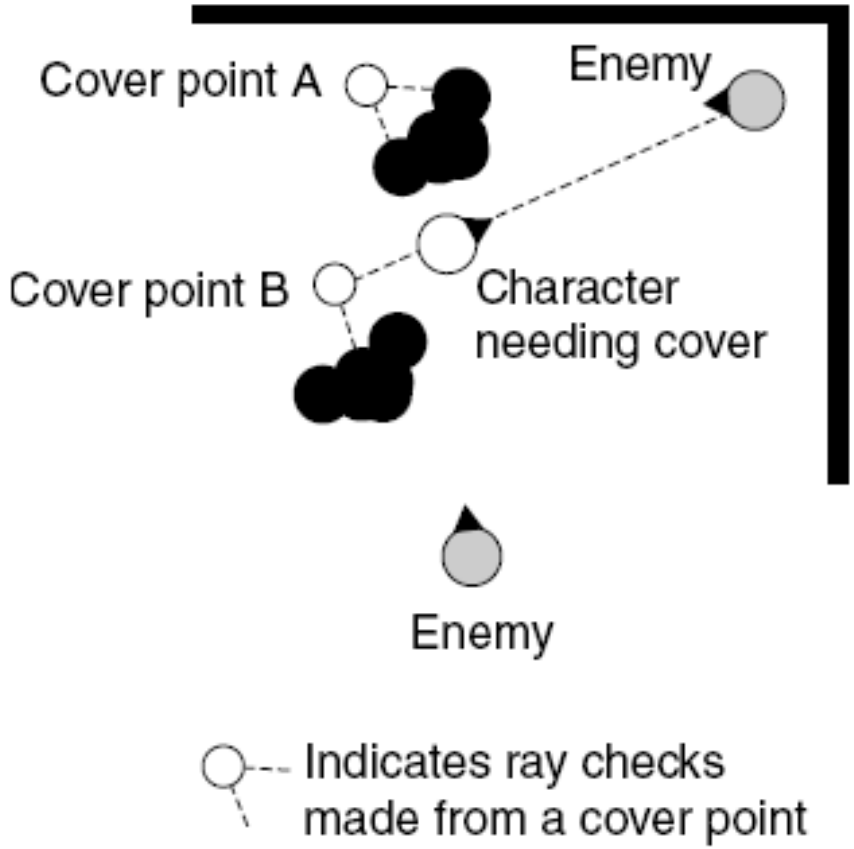


from "Artificial Intelligence for Games" by I. Millington & J. Funge

Tactical Locations ("rally points")

- Attributes that are dependent on the context
  - can be computed online for the given context

Disadvantages?



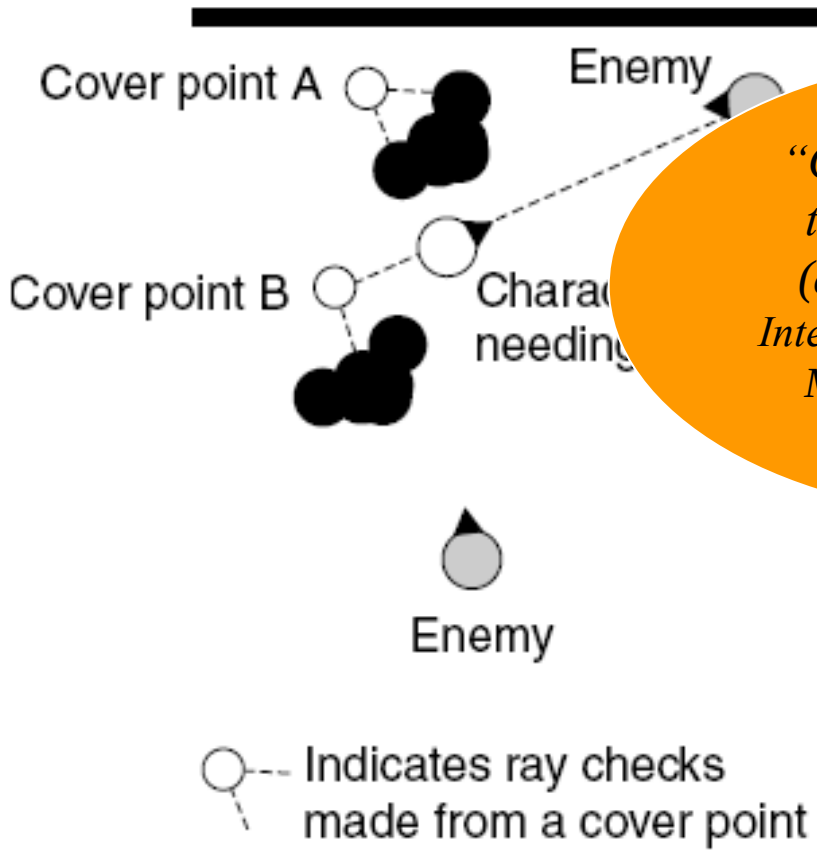
from "Artificial Intelligence for Games" by I. Millington & J. Funge

Tactical Locations (“rally points”)

- Attributes that are dependent on the context
  - can be computed online for the given context

Disadvantages?

“Can take up to 30% of total processor time” (according to “Artificial Intelligence for Games” by I. Millington & J. Funge)



from “Artificial Intelligence for Games” by I. Millington & J. Funge

Using Tactical Locations

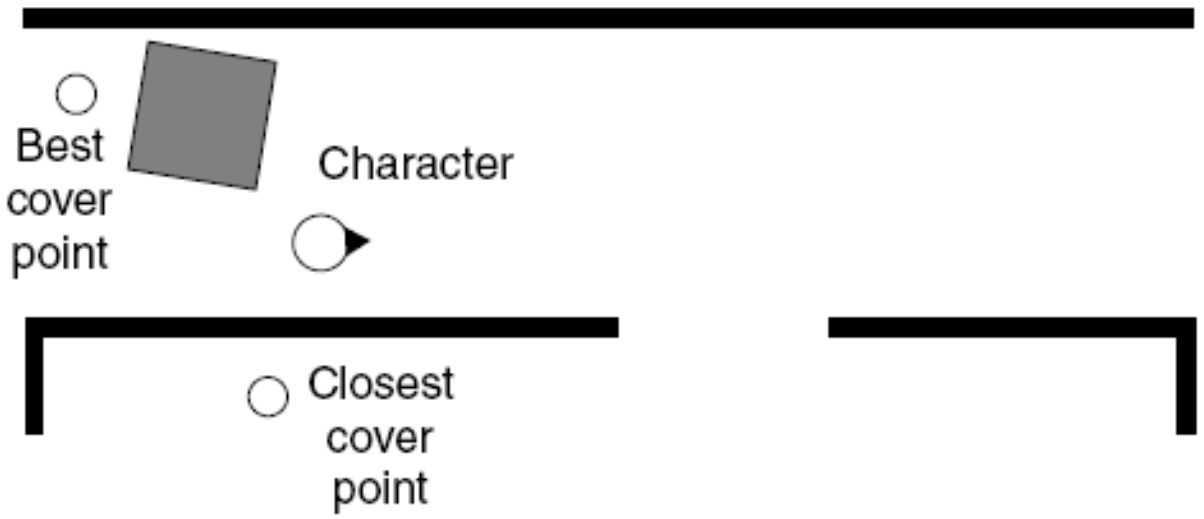
- Three uses
  - tactical movement (deciding where to go)

  - incorporating tactical information into the decision-making (deciding what to do)

  - incorporating tactical information into the pathfinding

Using Tactical Locations

- Three uses
  - tactical movement (deciding where to go)
    - e.g., once a decision to "reload" is made, the "nearest suitable tactical location" is looked up and the character moves there to "reload"



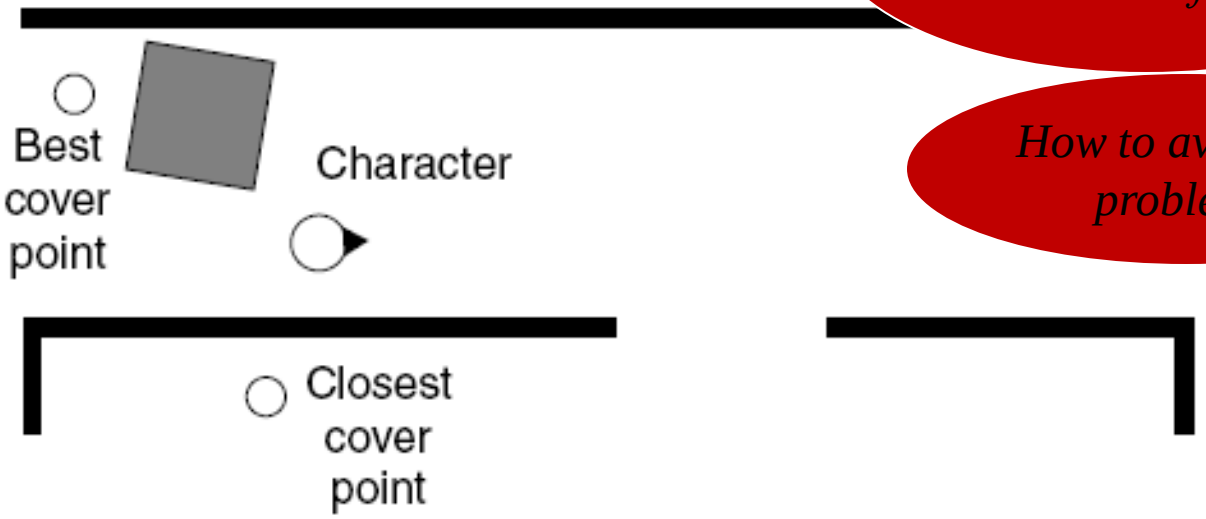
from "Artificial Intelligence for Games" by I. Millington & J. Funge

Using Tactical Locations

- Three uses
  - tactical movement (deciding where to go)
    - e.g., once a decision to "reload" is made, the "nearest suitable tactical location" is looked up and the character moves there to "reload"

What happens if no good location for "reload" is found?

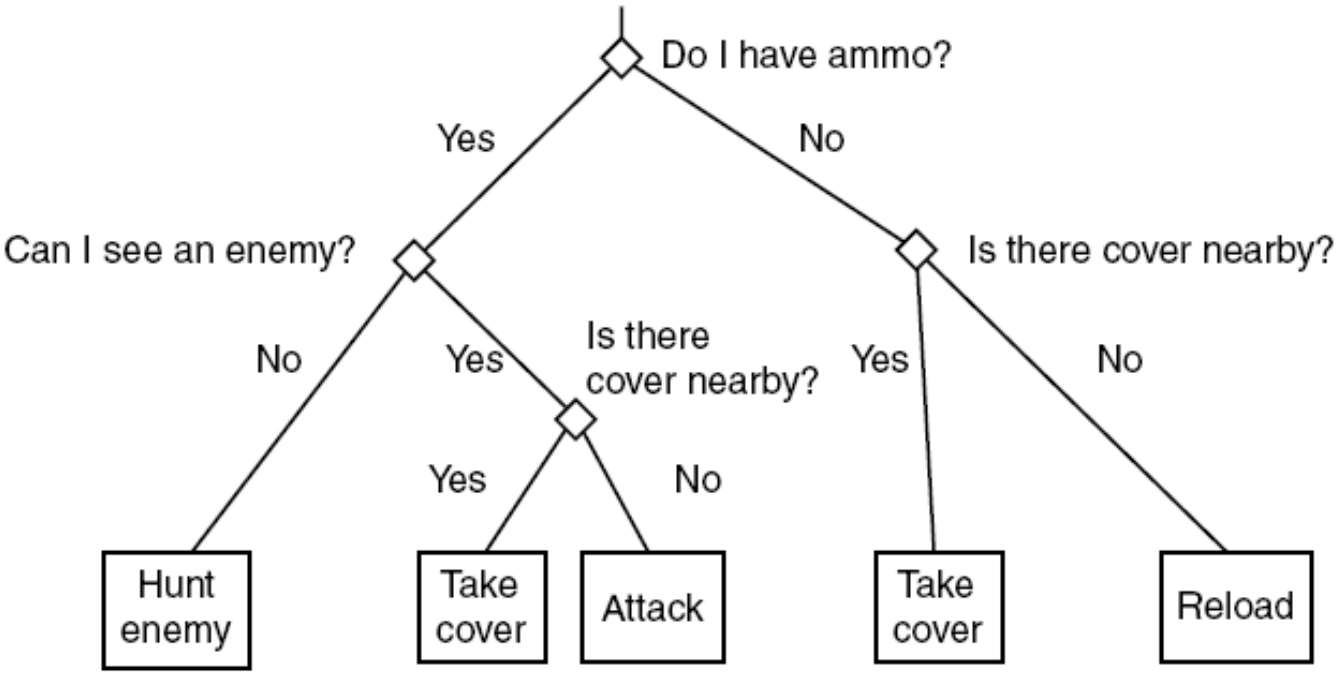
How to avoid this problem?



from "Artificial Intelligence for Games" by I. Millington & J. Funge

Using Tactical Locations

- Three uses
  - incorporating tactical information into the decision-making (deciding what to do)



from "Artificial Intelligence for Games" by I. Millington & J. Funge

Using Tactical Locations

- Three uses
  - incorporating tactical information into the pathfinding

  for example: modify the costs to penalize going through high exposure points (more on this later)

Computing Attributes of Tactical Locations

• Manual setting of attribute value can be tedious but allows to select favor “interesting” points (used in a large number of shooters)

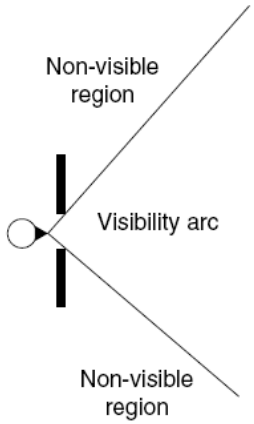
How to compute them automatically?

Computing Attributes of Tactical Locations

- Automatic setting of attribute value
  - typically based on ray-casting

Is point X a good visibility (can see much) point for a character:

How to compute it?



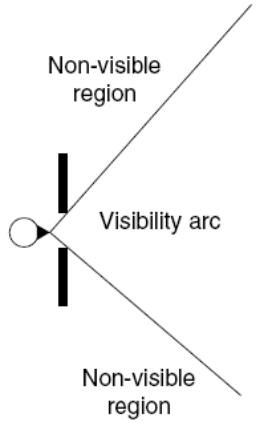
from "Artificial Intelligence for Games" by I. Millington & J. Funge

Computing Attributes of Tactical Locations

- Automatic setting of attribute value
  - typically based on ray-casting

Is point X a good visibility (can see much) point for a character:

select random (or at all angles around X) locations Y
for each location Y
cast a ray from Y to the position of eyes of the character situated at point X
set the visibility attribute to the proportion of collision-free rays



from "Artificial Intelligence for Games" by I. Millington & J. Funge

Computing Attributes of Tactical Locations

- Automatic setting of attribute value
  - typically based on ray-casting

Is point X a good cover point for a character:

How to compute it?

Computing Attributes of Tactical Locations

- Automatic setting of attribute value
  - typically based on ray-casting

Is point X a good cover point for a character:

select random (or at all angles around X) locations Y
for each location Y
cast a ray from Y to a random point on the surface of the character's body situated at point X
set the cover attribute to the proportion of collision-free rays

Automatic Computation of Tactical Locations

- The Condensation Algorithm

Construct a dense grid of possible locations with attribute values
For each pair of locations that are close and have line-of-sight
discard the location that has significantly lower attribute value
(can also be based on the weighted sum of attribute value difference and distance in between)
Do the above for every attribute and select points remaining for each attribute

Automatic Computation of Tactical Locations

- The Condensation Algorithm

Construct a dense grid of possible locations with att

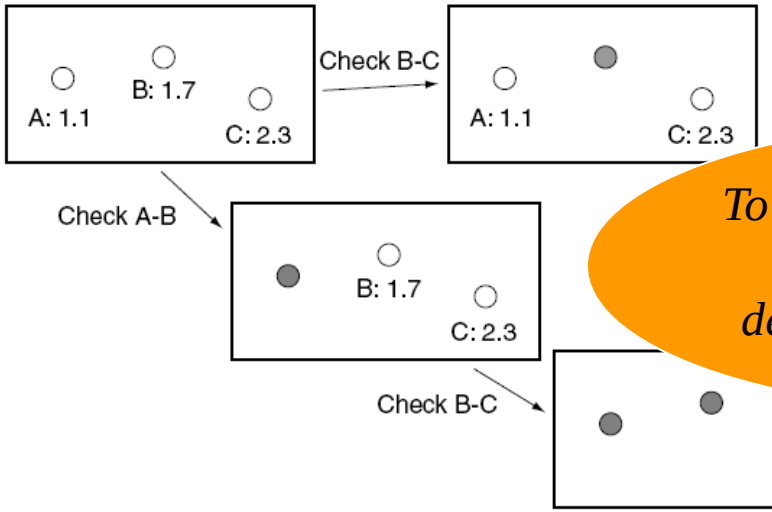
For each pair of locations that are close and have line-of-sight

discard the location that has significantly lower attribute value

(can also be based on the weighted sum of attribute value difference and distance in between)

Do the above for every attribute and select points remaining for each attribute

The order matters



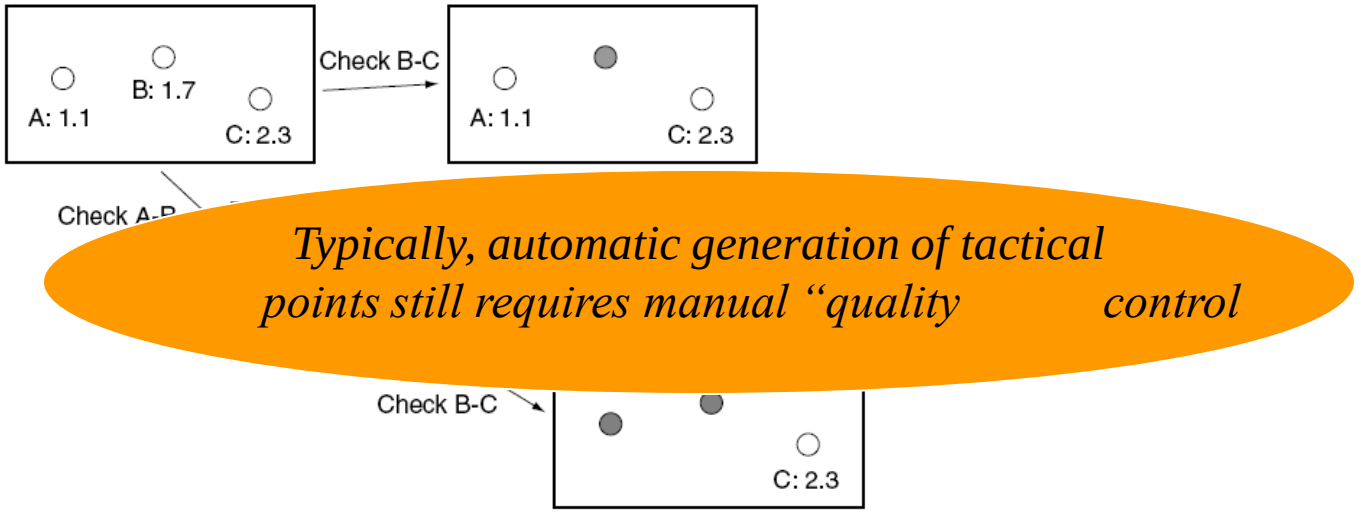
To help somewhat: process pairs in the order of decreasing differences in attribute values

from "Artificial Intelligence for Games" by I. Millington & J. Funge

Automatic Computation of Tactical Locations

- The Condensation Algorithm

Construct a dense grid of possible locations with attribute values
For each pair of locations that are close and have line-of-sight
discard the location that has significantly lower attribute value
(can also be based on the weighted sum of attribute value difference and distance in between)
Do the above for every attribute and select points remaining for each attribute



Typically, automatic generation of tactical points still requires manual "quality control

from "Artificial Intelligence for Games" by I. Millington & J. Funge

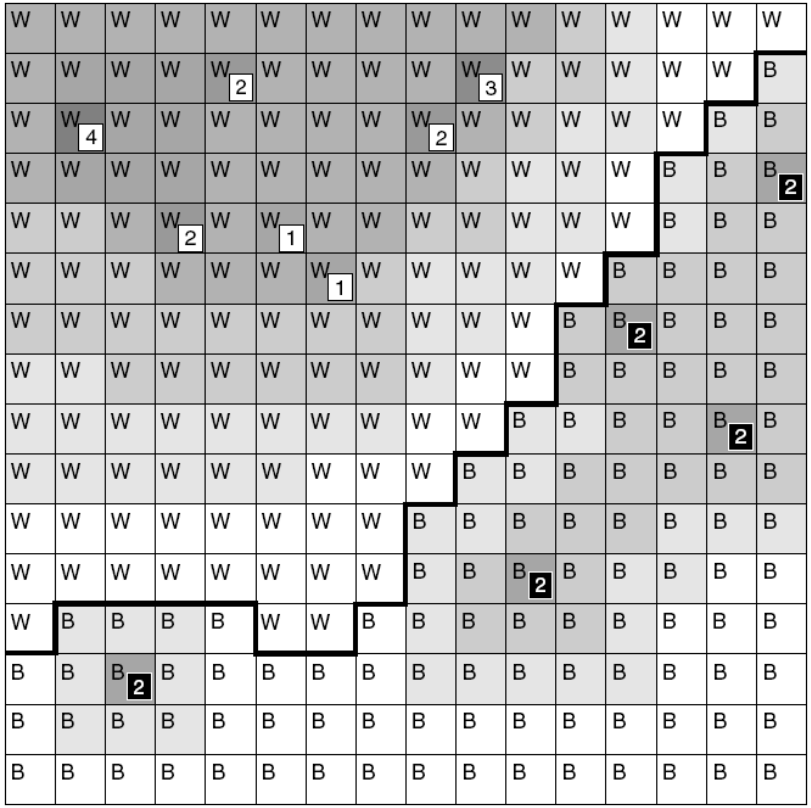
Tactical Analysis

• Simple Influence Maps: represent the balance of military influence at each location in the level based on

- the proximity and strength of close military units
- the duration since the last time the location was occupied
- the surrounding terrain
- …

Tactical Analysis

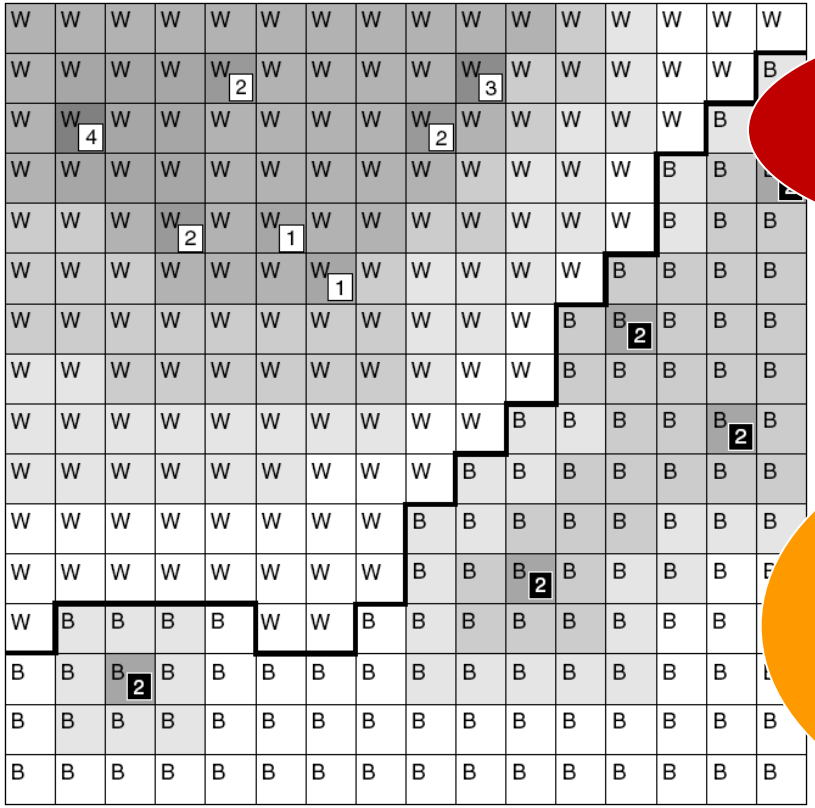
- Calculating influence maps assuming
  - limited radius of effect from each unit
  - influence is a function of distance ($I = f(d)$)
  - additive influence ($I_{total} = I_1 + I_2 + \ldots$)



from "Artificial Intelligence for Games" by I. Millington & F. Funge

Tactical Analysis

- Calculating influence maps assuming
  - limited radius of effect from each unit
  - influence is a function of distance ($I = f(d)$)
  - additive influence ($I_{total} = I_1 + I_2 + \ldots$)



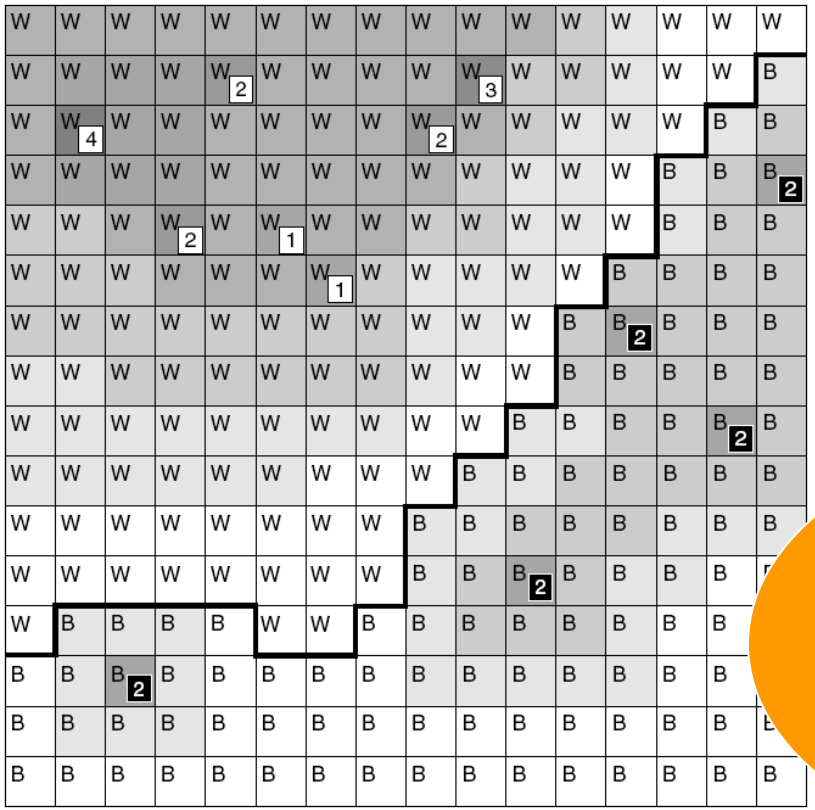
Efficient way to compute it for small # of units?

For each cell in their respective circles of influences, add the influence of the unit to the total influence at the cell

from "Artificial Intelligence for Games" by I. Millington & J. Funge

Tactical Analysis

- Calculating influence maps assuming
  - distance-based and total strength-based decaying effect



from "Artificial Intelligence for Games" by I. Millington

Compute using convolution filters (more expensive as it requires iteration over all cells in the map)

2D Convolution Filters

- Highly popular in games, graphics, computer science, engineering

Given a matrix of coefficients M and the original input matrix (map) X, any i,j element in the output matrix (map) Y is given by weighted summation of elements in X around $X_{i,j}$ with weights given by M

2D Convolution Filters

- Highly popular in games, graphics, computer science, engineering

Single Pass: given a matrix of coefficients M and the original input matrix (map) X, any i,j element in the output matrix (map) Y is given by weighted summation of elements in X around $X_{i,j}$ with weights given by M

$$M: \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$$X: \begin{matrix} 5 & 6 & 2 \\ 1 & 4 & 2 \\ 6 & 3 & 3 \end{matrix}$$

$$Y_{2,2} = \begin{pmatrix} 5 \times \frac{1}{16} & + & 6 \times \frac{2}{16} & + & 2 \times \frac{1}{16} & + \\ 1 \times \frac{2}{16} & + & 4 \times \frac{4}{16} & + & 2 \times \frac{2}{16} & + \\ 6 \times \frac{1}{16} & + & 3 \times \frac{2}{16} & + & 3 \times \frac{1}{16} & \end{pmatrix} = 3.5$$

2D Convolution Filters

- Highly popular in games, graphics, computer science, engineering

Single Pass: given a matrix of coefficients M and the original input matrix (map) X, any i,j element in the output matrix (map) Y is given by weighted summation of elements in X around $X_{i,j}$ with weights given by M

$$M: \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$$X: \begin{matrix} 5 & 6 & 2 \\ 1 & 4 & 2 \\ 6 & 3 & 3 \end{matrix}$$

$$Y_{2,2} = \begin{pmatrix} 5 \times \frac{1}{16} & + & 6 \times \frac{2}{16} & + & 2 \times \frac{1}{16} & + \\ 1 \times \frac{2}{16} & + & 4 \times \frac{4}{16} & + & 2 \times \frac{2}{16} & + \\ 6 \times \frac{1}{16} & + & 3 \times \frac{2}{16} & + & 3 \times \frac{1}{16} & \end{pmatrix} = 3.5$$

Set X = Y and repeat the above pass

Continue until Y doesn't change

2D Convolution Filters

- Highly popular in games, graphics, computer science, engineering

Single Pass: given a matrix of coefficients M and the original input matrix (map) X, any i,j element in the output matrix (map) Y is given by weighted summation of the neighbors of $X_{i,j}$ with weights given by M

Typically, M is normalized to have the total sum = 1 (ΣΣM=1)

$$M: \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$$Y_{2,2} = \begin{pmatrix} 5 \times \frac{1}{16} + 6 \times \frac{2}{16} + 2 \times \frac{1}{16} + \\ 1 \times \frac{2}{16} + 4 \times \frac{4}{16} + 2 \times \frac{2}{16} + \\ 6 \times \frac{1}{16} + 3 \times \frac{2}{16} + 3 \times \frac{1}{16} \end{pmatrix} = 3.5$$

$$X: \begin{matrix} 5 & 6 & 2 \\ 1 & 4 & 2 \\ 6 & 3 & 3 \end{matrix}$$

Set X = Y and repeat the above pass
Continue until Y doesn't change

In games, one pass per frame is often done

2D Convolution Filters

- Highly popular in games, graphics, computer science, engineering

Single Pass: given a matrix of coefficients M and the original input matrix (map) X, any i,j element in the output matrix (map) Y is given by weighted summation of elements in X around X [illegible] given by M

$$M: \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$$X: \begin{matrix} 5 & 6 & 2 \\ 1 & 4 & 2 \\ 6 & 3 & 3 \end{matrix}$$

$$Y_{2,2} = \begin{pmatrix} 5 \times \frac{1}{\cdot} \\ 1 \\ 6 \times \frac{\cdot}{16} \end{pmatrix}$$

Option 2: augment X with additional zero-boundaries (faster but creates artifacts)

Option 1: use only available entries in the summation

How to deal with boundaries?

Set X = Y and repeat the above pass

Continue until Y doesn't change

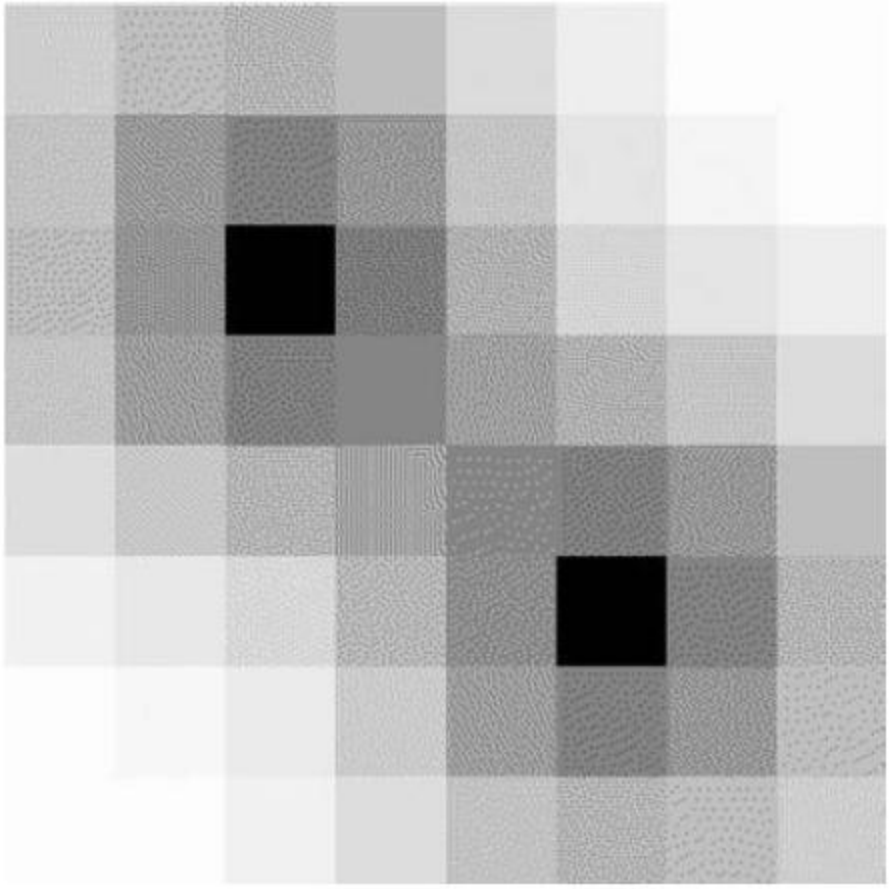
2D Convolution Filters

- Highly popular in games, graphics, computer science, engineering

Example of Gaussian (blurring) filter:

$$M = \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}.$$

$Y =$



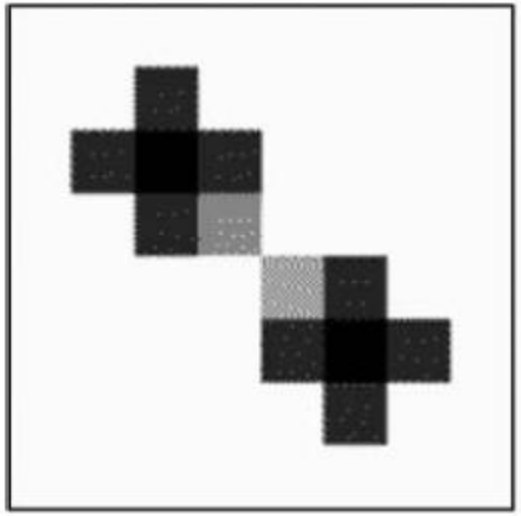
2D Convolution Filters

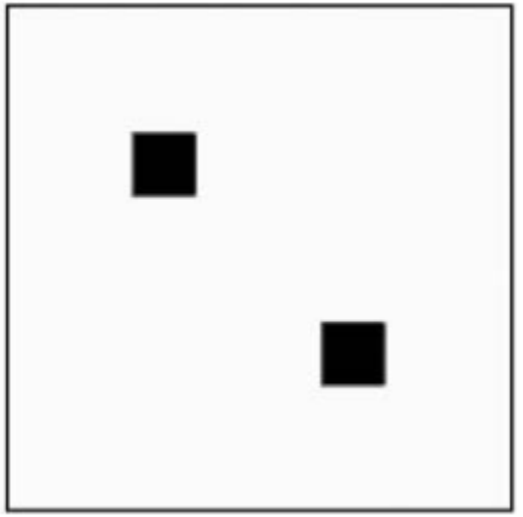
• Highly popular in games, graphics, computer science, engineering

Example of sharpening filter:

$$M: \frac{1}{a}\begin{bmatrix} -b & -c & -b \\ -c & a(4b+4c+1) & -c \\ -b & -c & -b \end{bmatrix} \quad \textit{for } a,b,c, > 0$$

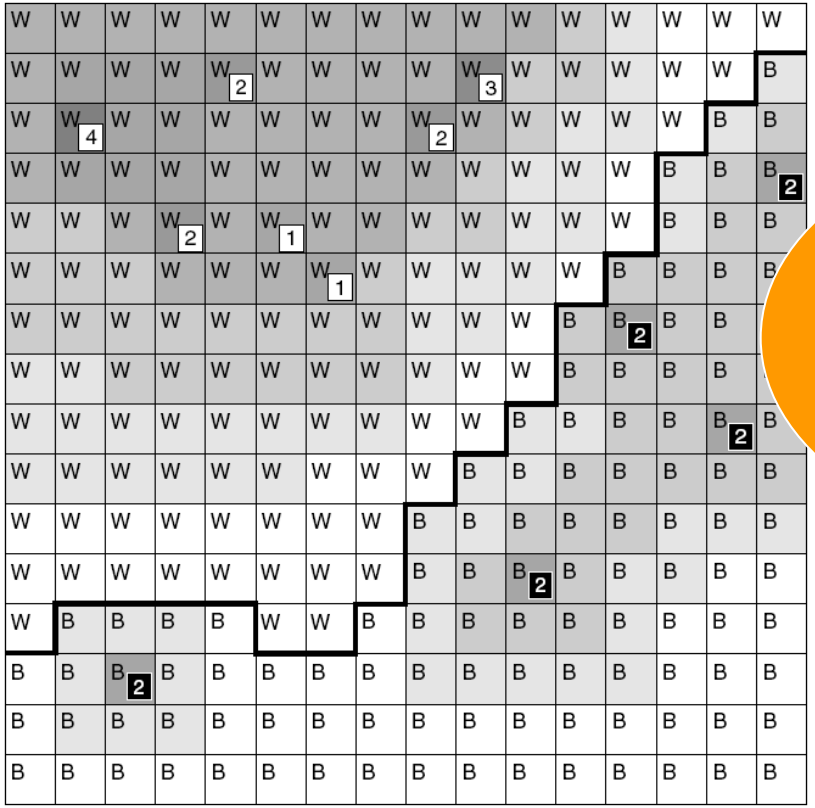
$$M: \frac{1}{2}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 18 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad X= \quad Y=$$





Tactical Analysis

- Calculating influence maps assuming
  - limited or unlimited radius of effect from each unit
  - influence is a function of distance ($I = f(d)$)
  - max influence ($I_{total} = max(I_1, I_2, \ldots)$)



from "Artificial Intelligence for Games" by I. Millington & J. Funge

Can be done by map flooding – single Dijkstra's search computing maximum influence to all unit locations

How?

Map Flooding Algorithm

- Remember A*?

Main function

$g(s_{start}) = 0$; all other g-values are infinite; $OPEN = \{s_{start}\}$;
ComputePath();
publish solution;

ComputePath function

while(s_{goal} is not expanded)
  remove s with the smallest $[f(s) = g(s)+h(s)]$ from $OPEN$;
  insert s into $CLOSED$;
  for every successor s' of s such that s' not in $CLOSED$
    if $g(s') > g(s) + c(s,s')$
      $g(s') = g(s) + c(s,s')$;
      insert s' into $OPEN$;

Map Flooding Algorithm

• Somewhat similar except for maximizing, no heuristics and multiple starts

Main function

OPEN = {};

For every s, g(s) = 0;

For every source cell s (e.g., every unit)

    g(s) = initial value of s; add s into OPEN;

while(OPEN is not empty)

    remove s with the largest g(s) from OPEN;

    for every successor s' of s

        gnew = strength of influence of s' on s given the value g(s')

        if g(s') < gnew

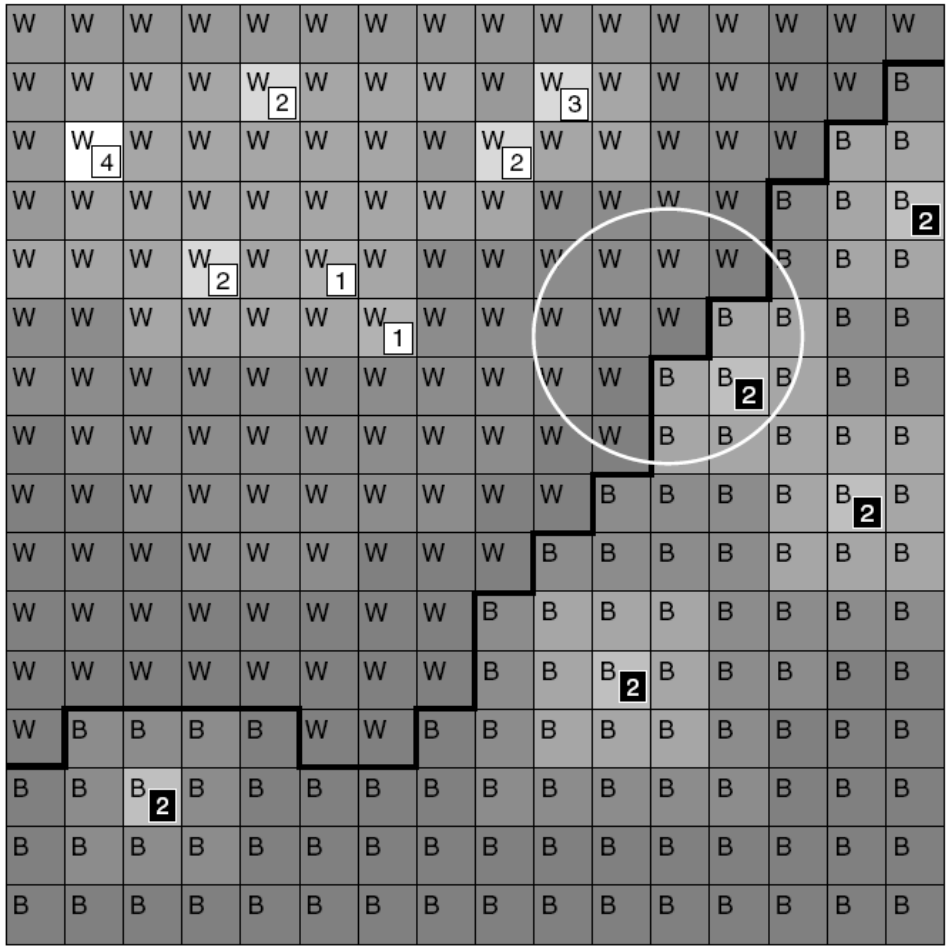
            g(s') = gnew;

            insert s' into OPy N;

Tactical Analysis

- Using influence maps

  Selecting a place to attack or to re-allocate troops:



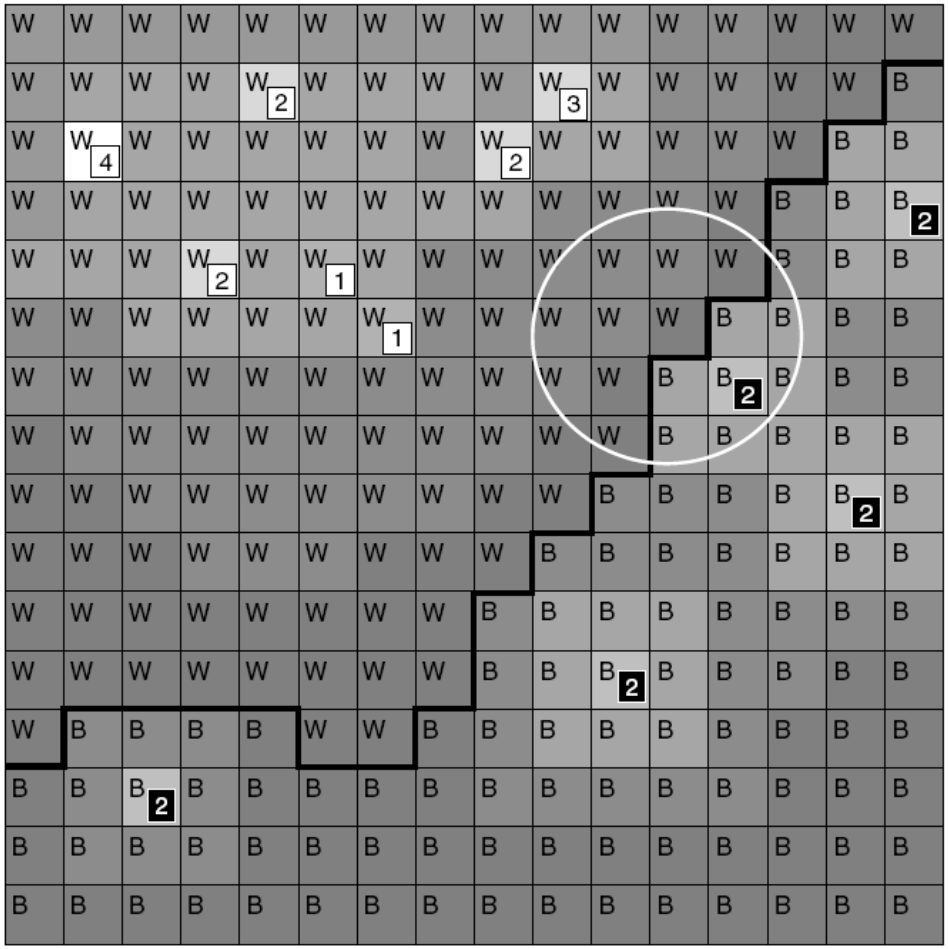
from "Artificial Intelligence for Games" by I. Millington & J. Funge

Tactical Analysis

- Using influence maps

How to deal with fog-of-war?

Selecting a place to attack or to re-allocate troops:

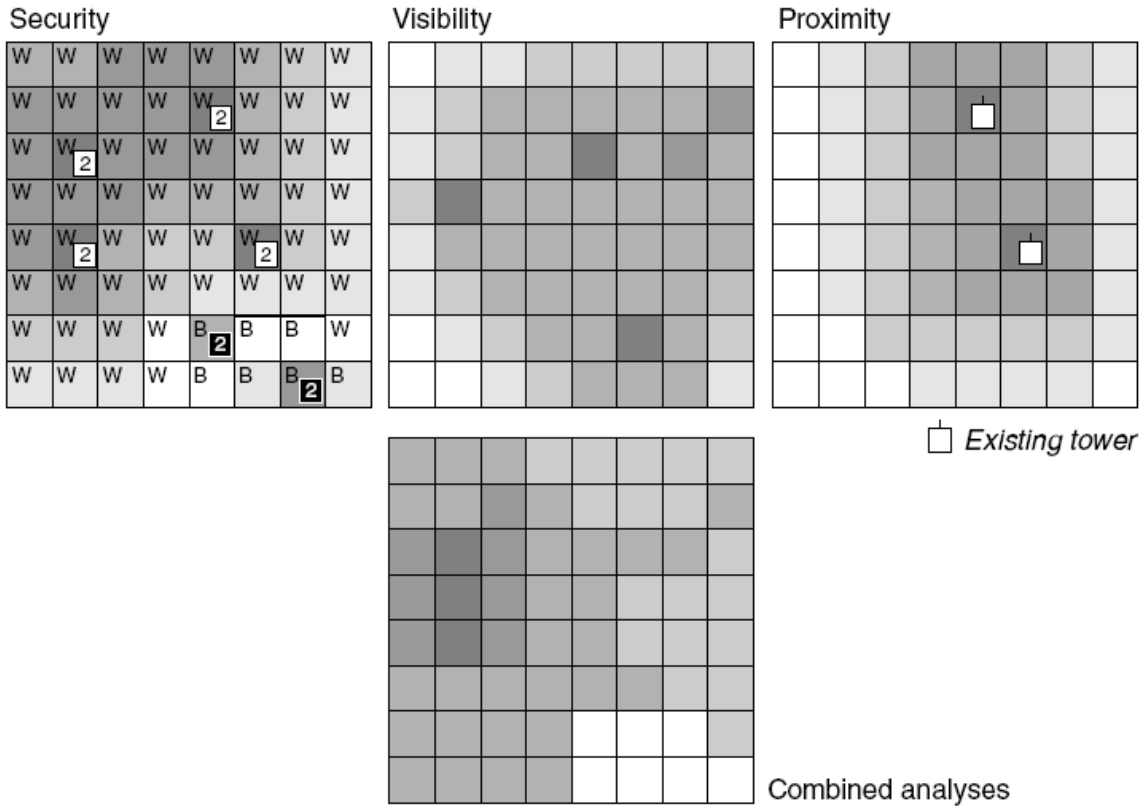


from "Artificial Intelligence for Games" by I. Millington & J. Funge

Tactical Analysis

- Terrain Analysis

Use the grid to compute such values as security, visibility and proximity to objects of interest. Use the weighted sum of these values to decide the strategically good location (e.g., position of a watch tower)



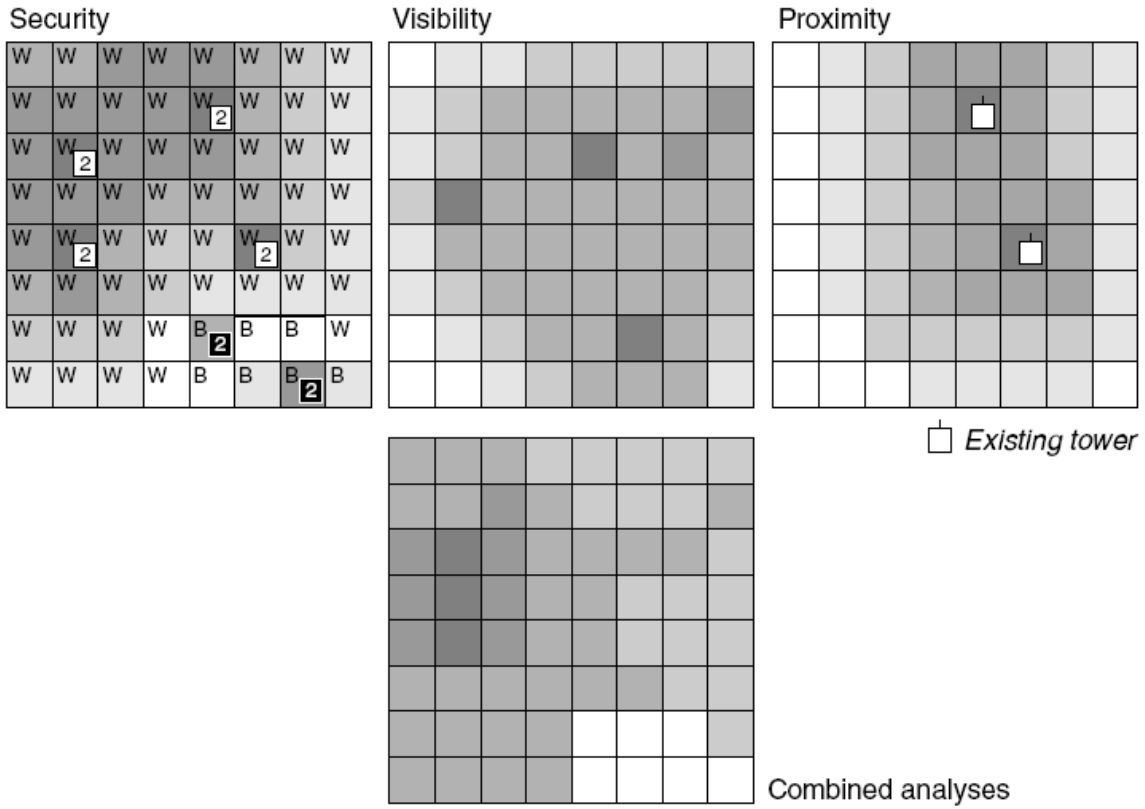
from "Artificial Intelligence for Games" by I. Millington & J. Funge

Tactical Analysis

- Terrain Analysis

How to compute security, visibility and proximity values?

Use the grid to compute such values as security, visibility and proximity to objects of interest. Use the weighted sum of these values to decide the strategically good location (e.g., position of a watch tower)



from "Artificial Intelligence for Games" by I. Millington & J. Funge

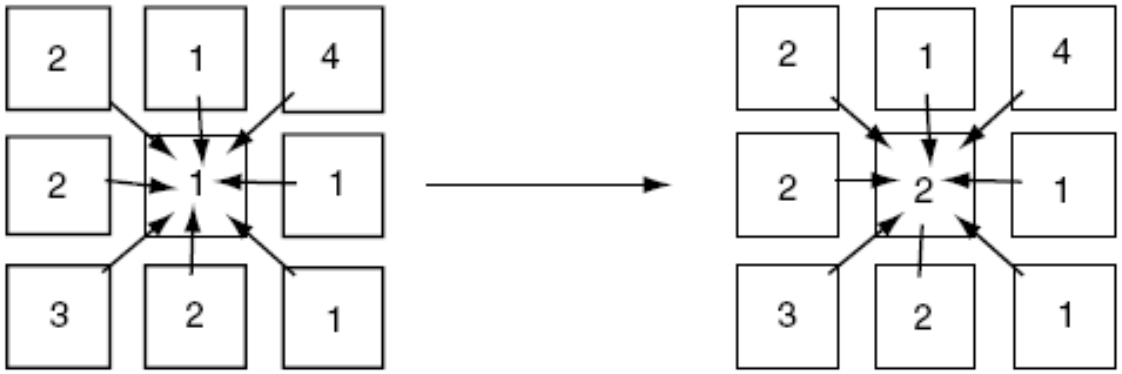
Cellular Automata

- General way to compute the values of cells based on the values of neighboring cells via complex rules

Algorithm:

Iterate through all the cells until fully converged

Using the rule, update each cell based on its neighbor values during previous iteration



IF two or more neighbors with higher values,
THEN increment

IF no neighbors with as high a value,
THEN decrement

from "Artificial Intelligence for Games" by I. Millington & J. Funge

Cellular Automata

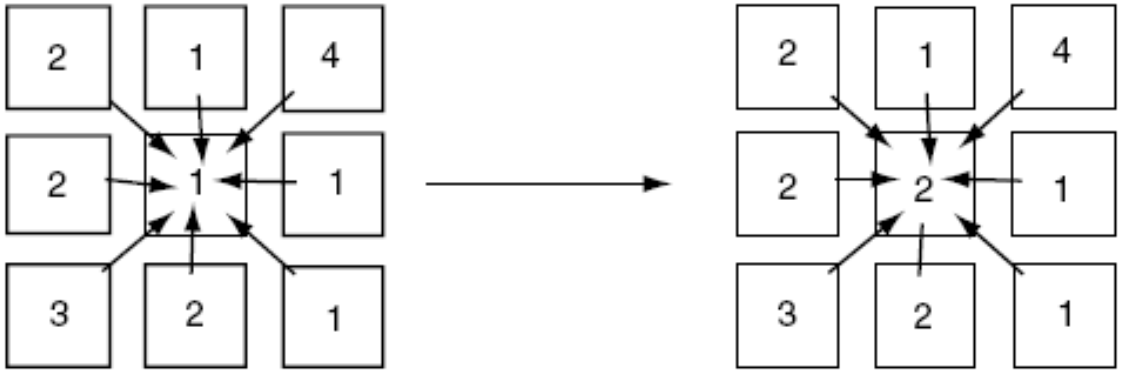
- General way to compute the values of cells based on the values of neighboring cells via complex rules

Requires two copies of the map

Algorithm:

Iterate through all the cells until fully converged

Using the rule, update each cell based on its neighbor values during previous iteration



IF two or more neighbors with higher values,
THEN increment

IF no neighbors with as high a value,
THEN decrement

from "Artificial Intelligence for Games" by I. Millington & J. Funge

Cellular Automata

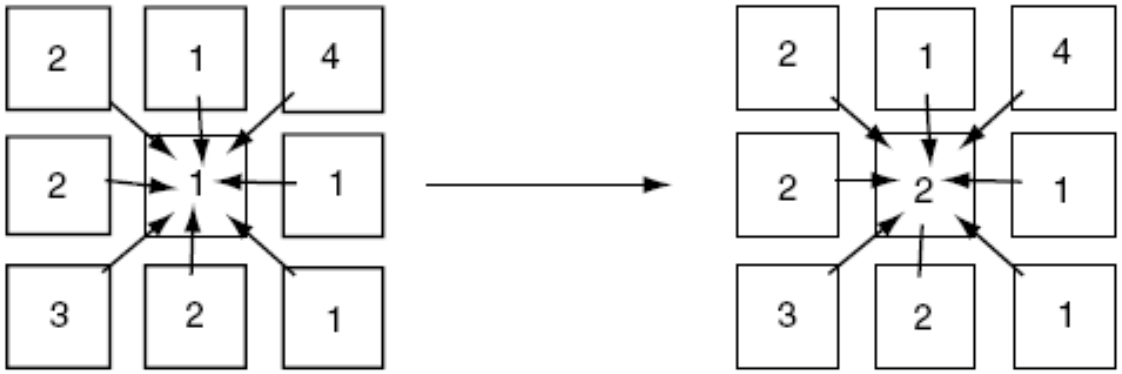
- General way to compute th
values of neighboring cells via complex rules

Anything we just learned is an example of cellular automata?

Algorithm:

Iterate through all the cells until fully converged

Using the rule, update each cell based on its neighbor values during previous iteration



IF two or more neighbors with higher values,
THEN increment

IF no neighbors with as high a value,
THEN decrement

from "Artificial Intelligence for Games" by I. Millington & J. Funge

Cellular Automata

- General way to compute the [illegible] the values of neighboring cells via [illegible]

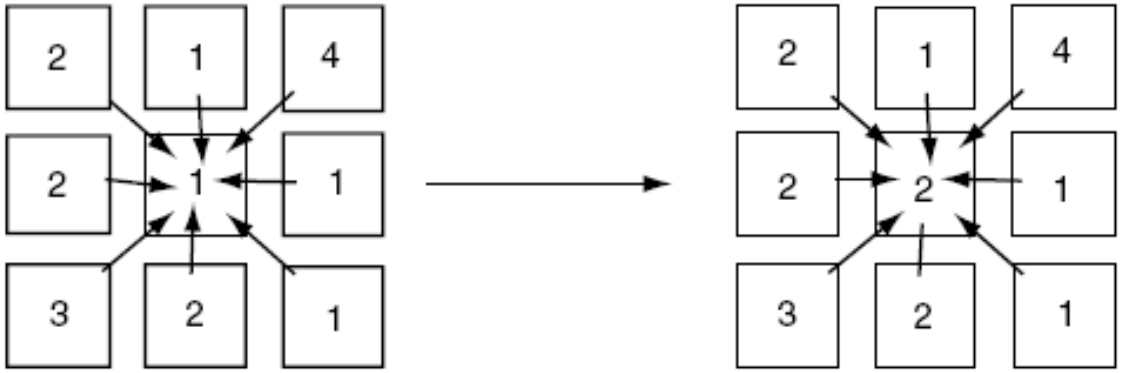
What about the convolution filter and why?

Is it guaranteed to converge?

Algorithm:

Iterate through all the cells until fully converged

Using the rule, update each cell based on its neighbor values during previous iteration



IF two or more neighbors with higher values,
THEN increment

IF no neighbors with as high a value,
THEN decrement

from "Artificial Intelligence for Games" by I. Millington & J. Funge

Cellular Automata

- Famous example of Cellular Automata

Conway Game of Life:

http://en.wikipedia.org/wiki/Conway's_Game_of_Life

Cellular Automata

- Applications of Cellular Automata in games:

Rule for Area of Security:

A location (cell) is secure if at least four of its eight neighbors (50%) are secure

Cellular Automata

- Applications of Cellular Automata in games:

Rules for Building a City (e.g., Age of Empire, SimCity, ...) :

A location is good for building a defense building if at least one neighbor has raw materials

A location is good for building a basic building if at least two of its neighbors have defensive buildings

A location is good for building valuable facilities if at least two of its neighbors have basic buildings

Tactical Pathfinding

- Takes surrounding environment (stealth, avoiding possible enemy locations, …) into account when planning a path
- Can generate impressive results (e.g., intelligently looking behavior) at little expense
- Hot topic in game development

Tactical Pathfinding

- Takes surrounding environment (stealth, avoiding possible enemy locations, …) into account when planning a path
- Can generate impressive results (e.g., intelligently looking behavior) at little expense
- Hot topic in game development

Any ideas how to do it?

Tactical Pathfinding

- Takes surrounding environment (stealth, avoiding possible enemy locations, …) into account when planning a path

Cost Function to incorporate tactical info:

Cost of moving from cell s to cell s'

$$c(s,s') = Distance(s,s') + \sum w_i \, T_i(s')$$

$T_i(s')$ – the value of ith tactical info at cell s'
w_i – importance (weight) of this tactical info for this particular character

T_i-values could come from influence maps, visibility computations on the fly, terrain analysis, etc.

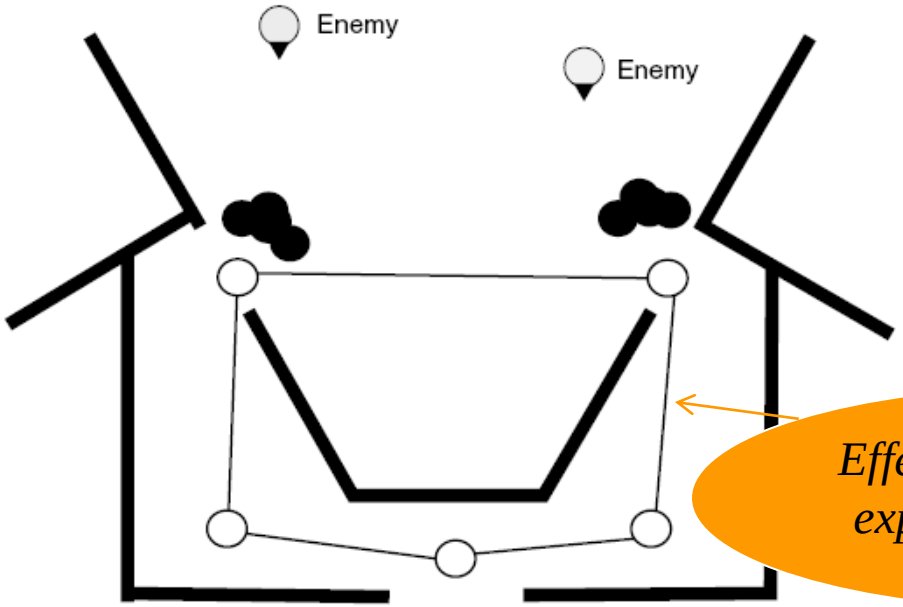
Tactical Pathfinding

- Takes surrounding environment (stealth, avoiding possible enemy locations, …) into account when planning a path

Cost Function to incorporate tactical info:

Cost of moving from cell s to cell s'

$$c(s,s') = Distance(s,s') + \sum w_i \, T_i(s')$$



Effect of incorporating exposure information into pathfinding

from "Artificial Intelligence for Games" by I. Millington & J. Funge

Tactical Pathfinding

• Takes surrounding environment (stealth, avoiding possible enemy locations, …) into account when planning a path

Cost Function to incorporate tactical info:

Cost of moving from cell s to cell s'

$$c(s,s') = Distance(s,s') + \sum w_i \, T_i(s')$$

$T_i(s')$ can even be negative (e.g., prefer to go through friendly areas)

Potential problems?

Better (if possible) to raise the costs of all other cells and assign smaller costs to cells we favor

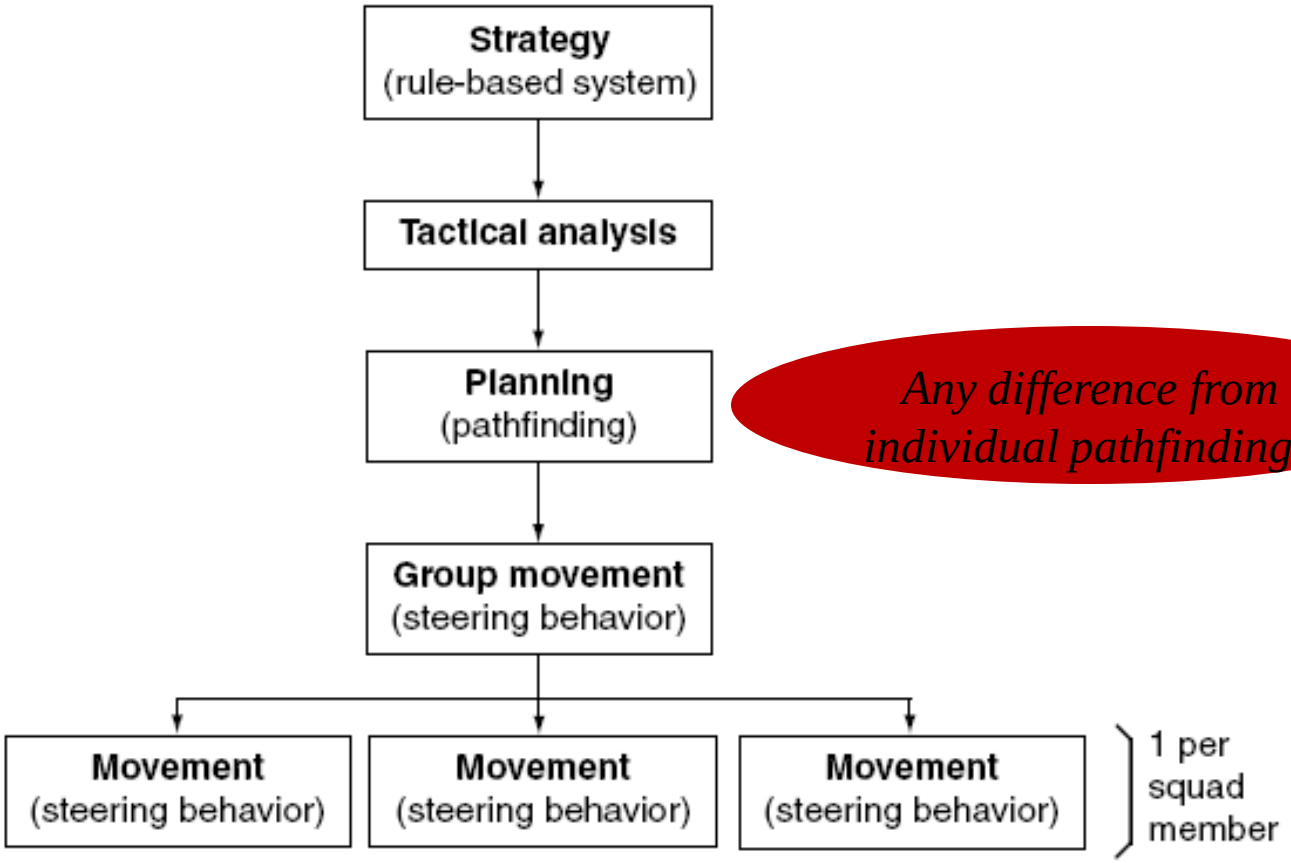
Coordinated Actions

- Cooperation among multiple characters (e.g., in RTS)
- Cooperation between player and characters (e.g., in Shooter)

Coordinated Actions

- Cooperation among multiple characters (e.g., in RTS)

Multi-tier AI for cooperating characters:



Any difference from individual pathfinding?

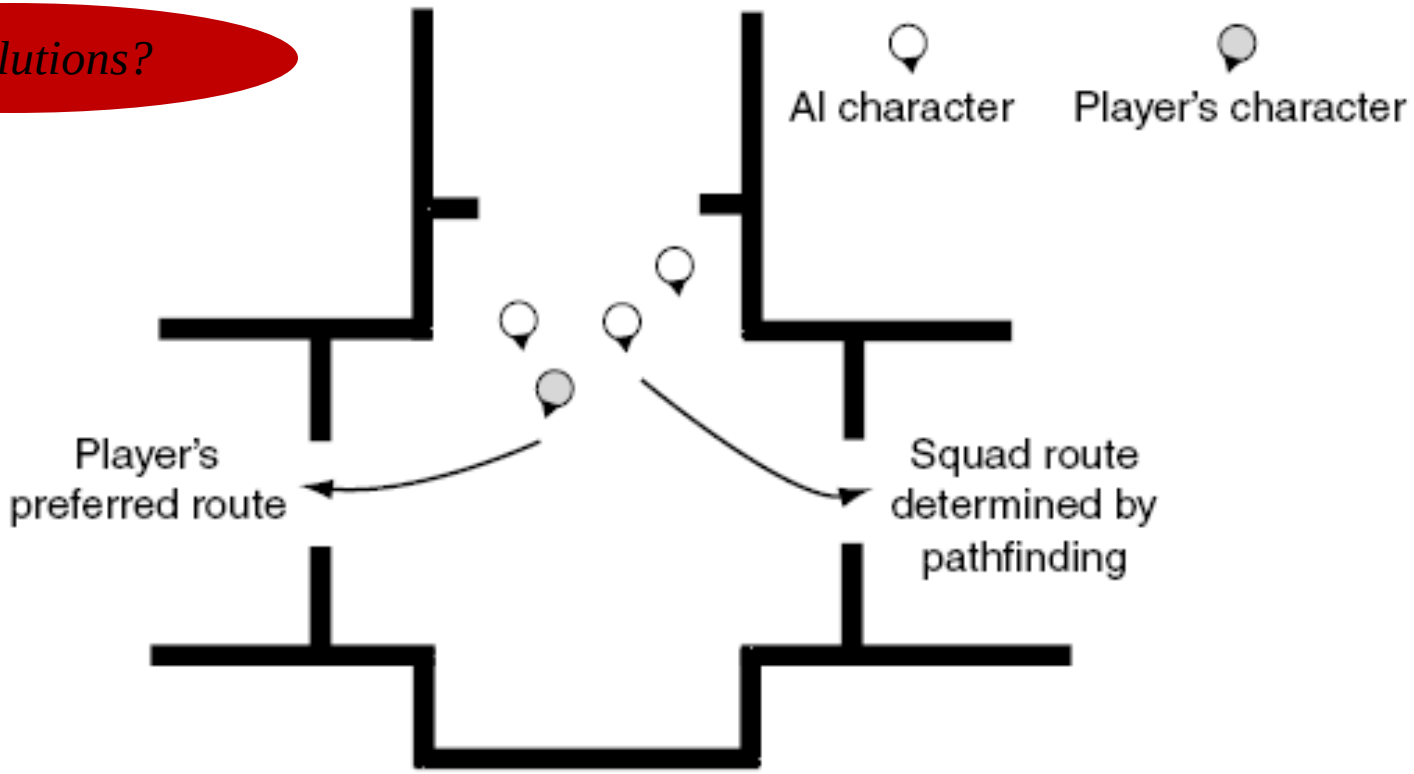
from "Artificial Intelligence for Games" by I. Millington & J. Funge

Coordinated Actions

- Cooperation between player and characters (e.g., in Shooter)

Why cooperation between player and characters requires different model:

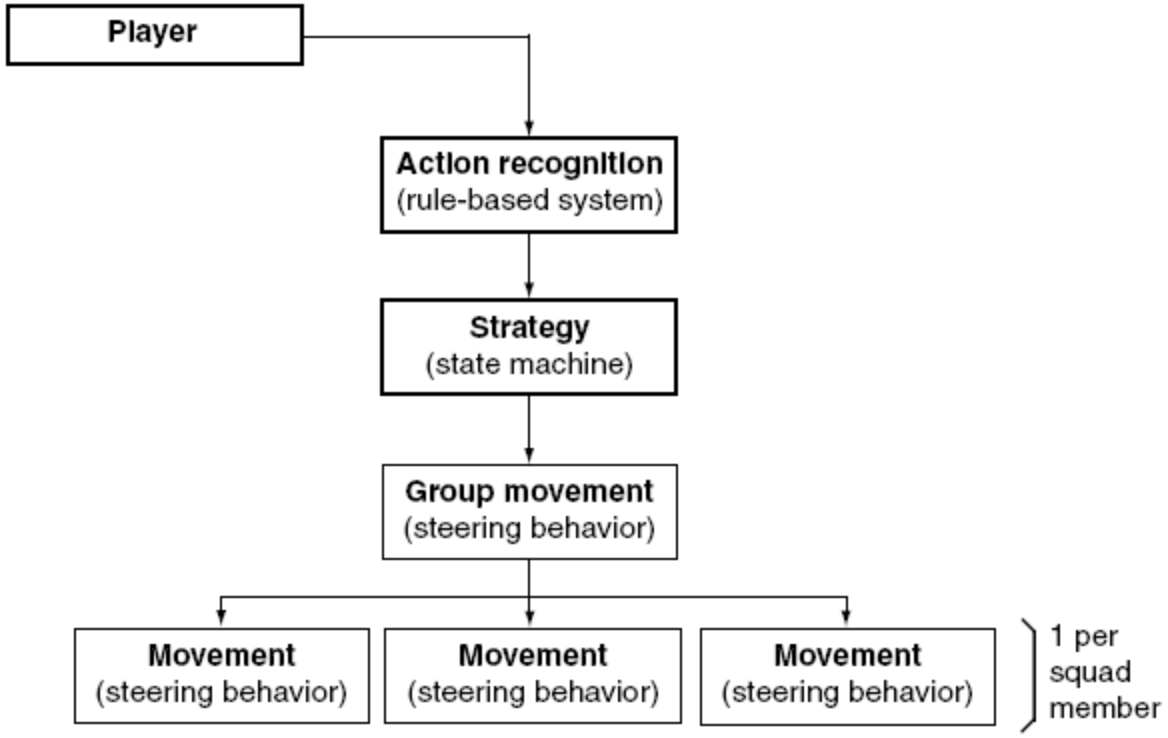
Any solutions?



from "Artificial Intelligence for Games" by I. Millington & J. Funge

Coordinated Actions

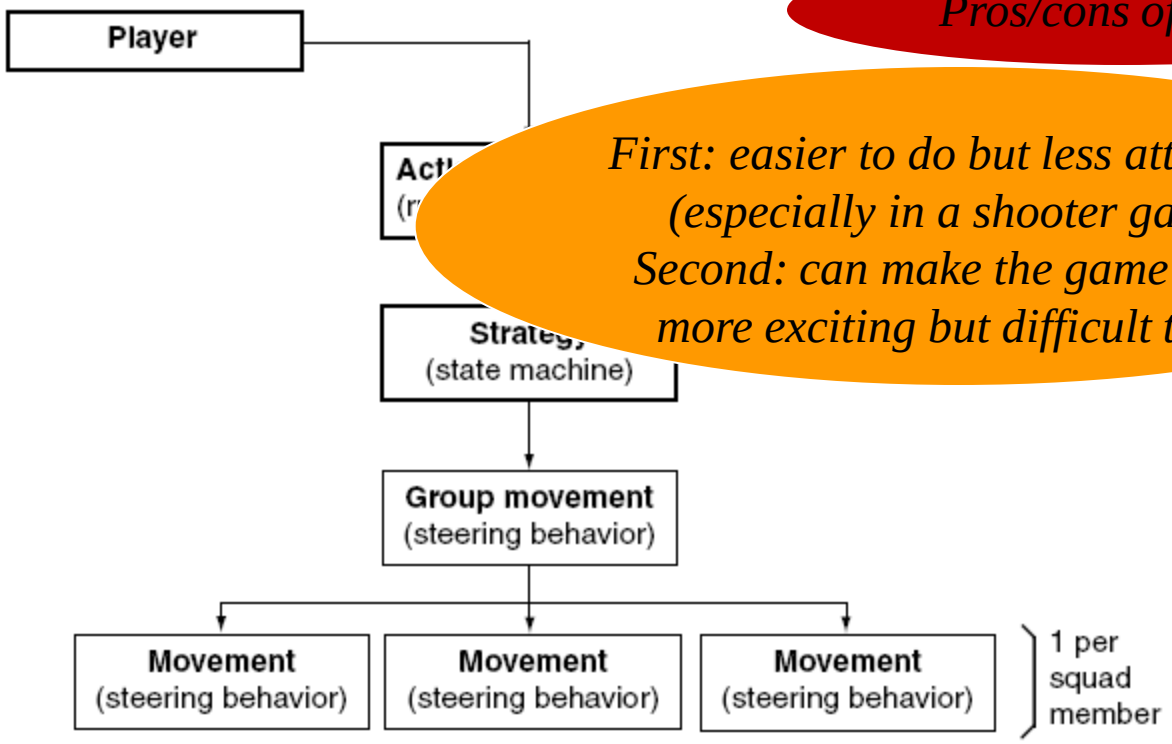
- Cooperation between player and characters (e.g., in Shooter)
  -player either gives direct orders to characters (teammates)
  -or characters (teammates) try to recognize play er intentions through his actions and decide how to assist



from "Artificial Intelligence for Games" by I. Millington & J. Funge

Coordinated Actions

- Cooperation between player and characters (e.g., in Shooter)
  -player either gives direct orders to characters (teammates)
  -or characters (teammates) try to recognize play er intentions through his actions and decide how to assist



Pros/cons of each?

First: easier to do but less attractive (especially in a shooter game)
Second: can make the game much more exciting but difficult to do

from "Artificial Intelligence for Games" by I. Millington & F.unge

Emergent Cooperation

- Decentralized control:
  -each character runs its own decision-making based on its information about other characters
  -the hope is that the behavior of the overall team (group) will emerge to be intelligent

  Perfect example: Rey nolds flocking algorithm

Emergent Cooperation

- Decentralized control:
  -each character runs its own decision-making based on its information about other characters
  -the hope is that the behavior of the overall team (group) will emerge to be intelligent

One of the holy grails of multi-agent AI

Perfect example: Rey nolds flocking algorithm

For now, very often the emergent behavior will be Unintelligent and annoying

Lack of control makes it difficult to use extensively in games