Lecture Overview

- ❑ Waypoint Tactics
- ❑ Tactical Analysis
- ❑ Tactical Pathfinding
- ❑ Coordinated Action

Tactical Analyses

- Sometimes known as <u>influence maps</u> – a technique pioneered and widely used in RTS games where the AI keeps track of areas of military influence in game
- Can also be used in <u>simulation/evolution</u> games, <u>FPSs</u> or <u>massively multi-player games</u>
- Overwhelming majority of current implementations are based on tile-based grid worlds. Even for non-tile-based worlds, a grid can be imposed over the geometry for tactical analyses.

Influence Maps

- Keeps track of current balance of military influence at each location in level
- E.g. Factors: proximity of military units, proximity of base, duration since a unit last occupied a location, terrain, current financial situation, weather, *etc.*

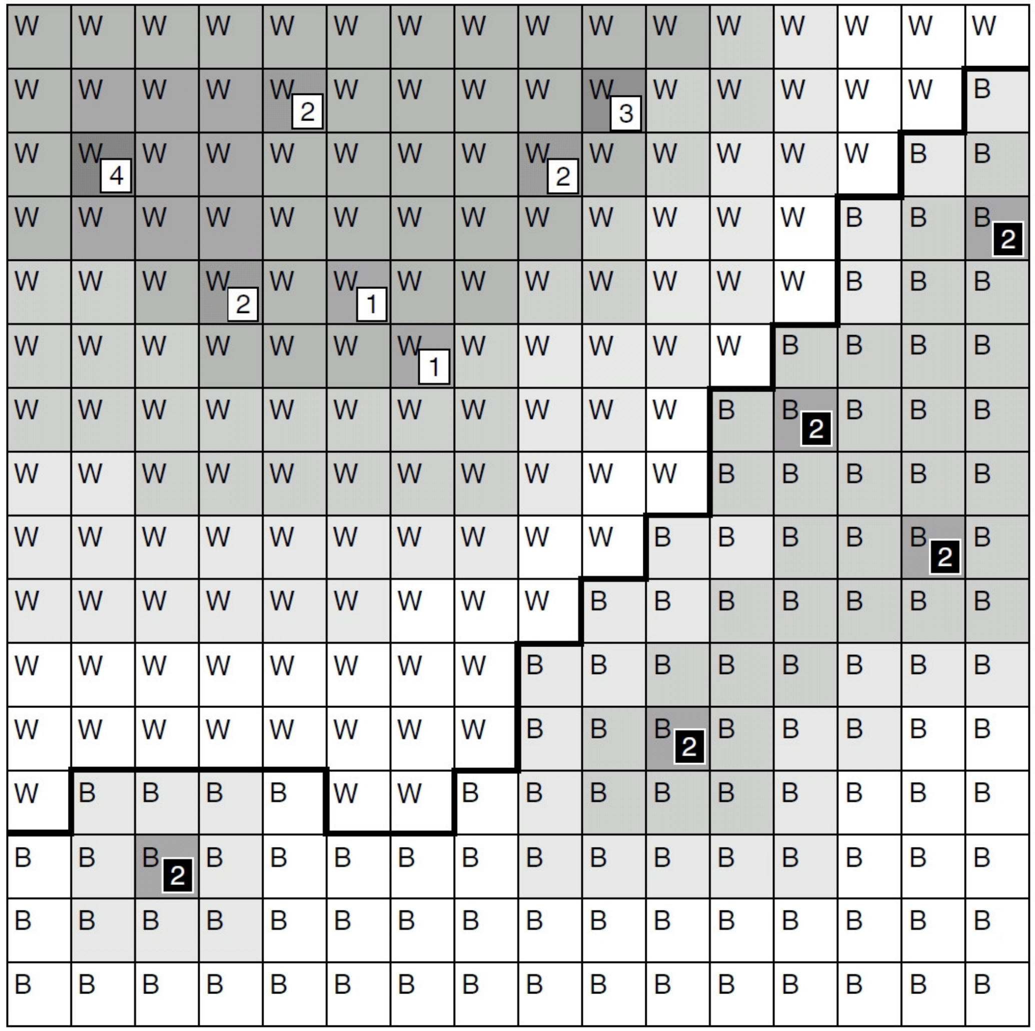
- In many games, simple influence maps are constructed based on these popular factors:
  - Proximity of enemy units and bases, and
  - Their relative military power

Influence Calculations

- **Concept: Influence is taken to drop off with distance d. The farther away from a unit/base/ *etc.*, the lesser the value of their influence, Id**
- **Linear model:** $I_d = \frac{I_0}{1+d}$

  where I_0 is the influence at distance 0
- **Non-linear models:** $I_d = \frac{I_0}{\sqrt{1+d}}$ $\quad I_d = \frac{I_0}{(1+d)^2}$
- **In practice, *linear drop off is perfectly reasonable*, and is also faster to compute**
- **Use additive influence at a point (cell) in the influence map:** $I_{\text{total}} = I_1 + I_2 + \cdots$

Influence Map Example



W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
W	W	W	W	W 2	W	W	W	W	W 3	W	W	W	W	W	B
W	W 4	W	W	W	W	W	W	W 2	W	W	W	W	W	B	B
W	W	W	W	W	W	W	W	W	W	W	W	W	B	B	B 2
W	W	W	W 2	W	W 1	W	W	W	W	W	W	W	B	B	B
W	W	W	W	W	W	W 1	W	W	W	W	W	B	B	B	B
W	W	W	W	W	W	W	W	W	W	W	B	B 2	B	B	B
W	W	W	W	W	W	W	W	W	W	W	B	B	B	B	B
W	W	W	W	W	W	W	W	W	W	B	B	B	B	B 2	B
W	W	W	W	W	W	W	W	W	B	B	B	B	B	B	B
W	W	W	W	W	W	W	W	B	B	B	B	B	B	B	B
W	W	W	W	W	W	W	W	B	B	B 2	B	B	B	B	B
W	B	B	B	B	W	W	B	B	B	B	B	B	B	B	B
B	B	B 2	B	B	B	B	B	B	B	B	B	B	B	B	B
B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B
B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B

- **An influence map calculated for all locations on a tiny RTS map. There are <u>two sides</u>, white and black, with a few units on each side.**
- **The side with the greatest influence on a location has control over it.**

Influence Calculations

- To calculate the map, *need to consider each unit* in the game *for each location* in the level.
- Up to billions of calculations may be needed! Execution time: O(*nm*), Memory: O(*m*), where *m* number of locations, *n* number of units.
- 3 approaches:
  - Limited Radius of Effect
  - Convolution Filters
  - Map Flooding

Limited Radius of Effect

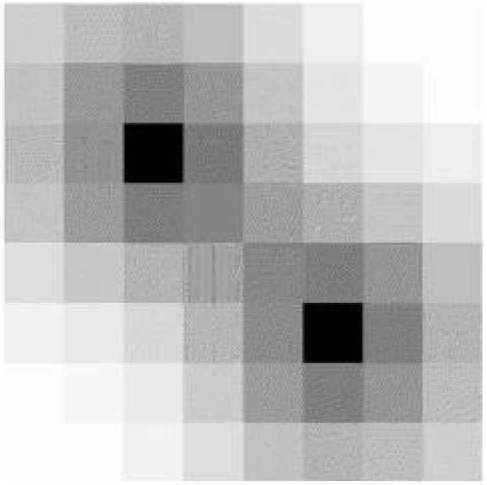
- Limit the radius of influence effect for each unit
- Each unit has a maximum radius of influence; *beyond that radius*, no computation is required
- Use a threshold influence, I_t (*beyond which influence is zero*), the radius is given by

$$R = \frac{I_0}{I_t - 1}$$

- This approach results in O(*nr*) in time, where *r* is number of locations within the average radius of a unit. $r << m$: much faster!
- Any disadvantages?

Other Calculations

Convolution Filters



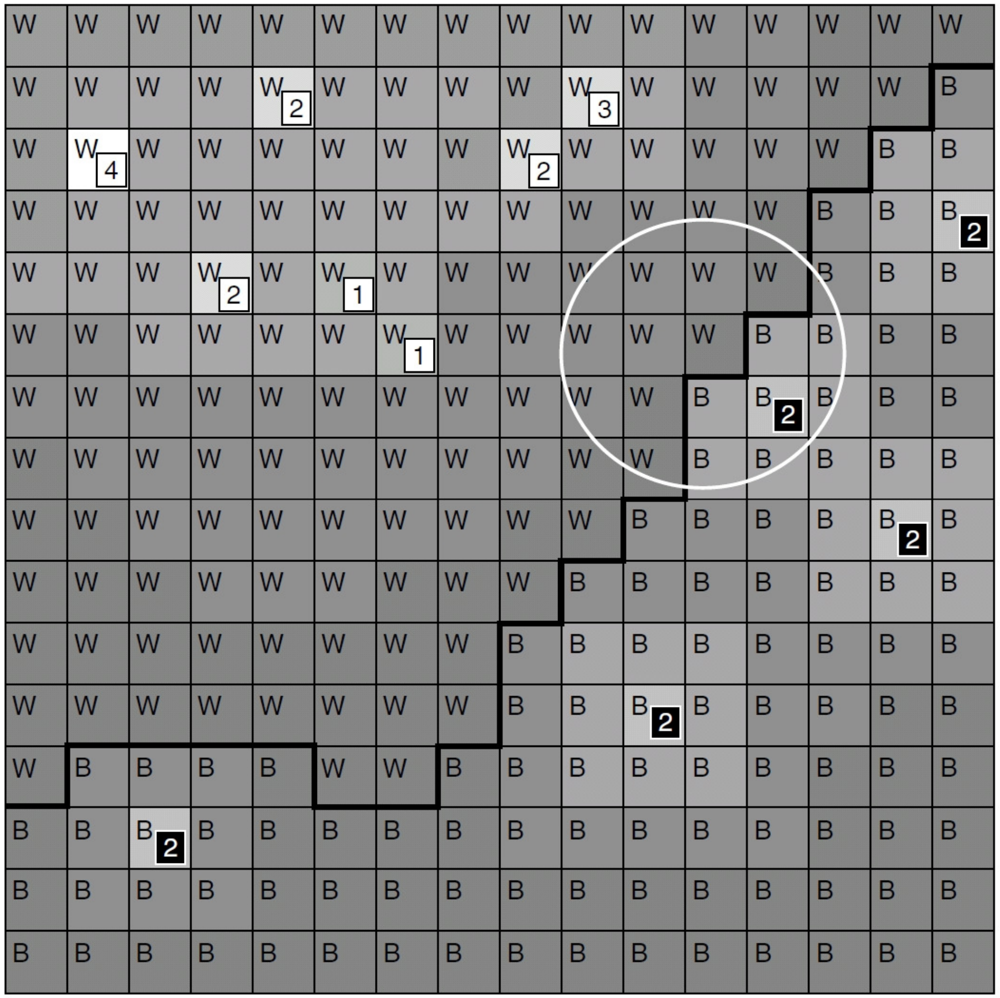
Gaussian filter

- Takes advantage of graphical techniques to blur the information of the spots of influence on the map *by applying a filter*.

Map Flooding

- Based on the assumption that the *influence of each location* is equal to the largest influence contributed by any unit (*e.g.*, $I = \max_i\left(I_{0,i}/(1+d_i)\right)$).
- The algorithm floods the map with values, starting from each unit in the game and *propagating its influence out*.

Influence Map: Examples



W W W W W W W W W W W W W W W W
W W W W W 2 W W W W W 3 W W W W W B
W W 4 W W W W W W W 2 W W W W W B B
W W W W W W W W W W W W W B B B 2
W W W W 2 W W 1 W W W W W W W B B B
W W W W W W W 1 W W W W W B B B B
W W W W W W W W W W W B B 2 B B B
W W W W W W W W W W W B B B B B
W W W W W W W W W W B B B B B 2 B
W W W W W W W W W B B B B B B B
W W W W W W W W B B B B B B B B
W W W W W W W W B B B 2 B B B B B
W B B B B W W B B B B B B B B B
B B B 2 B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B

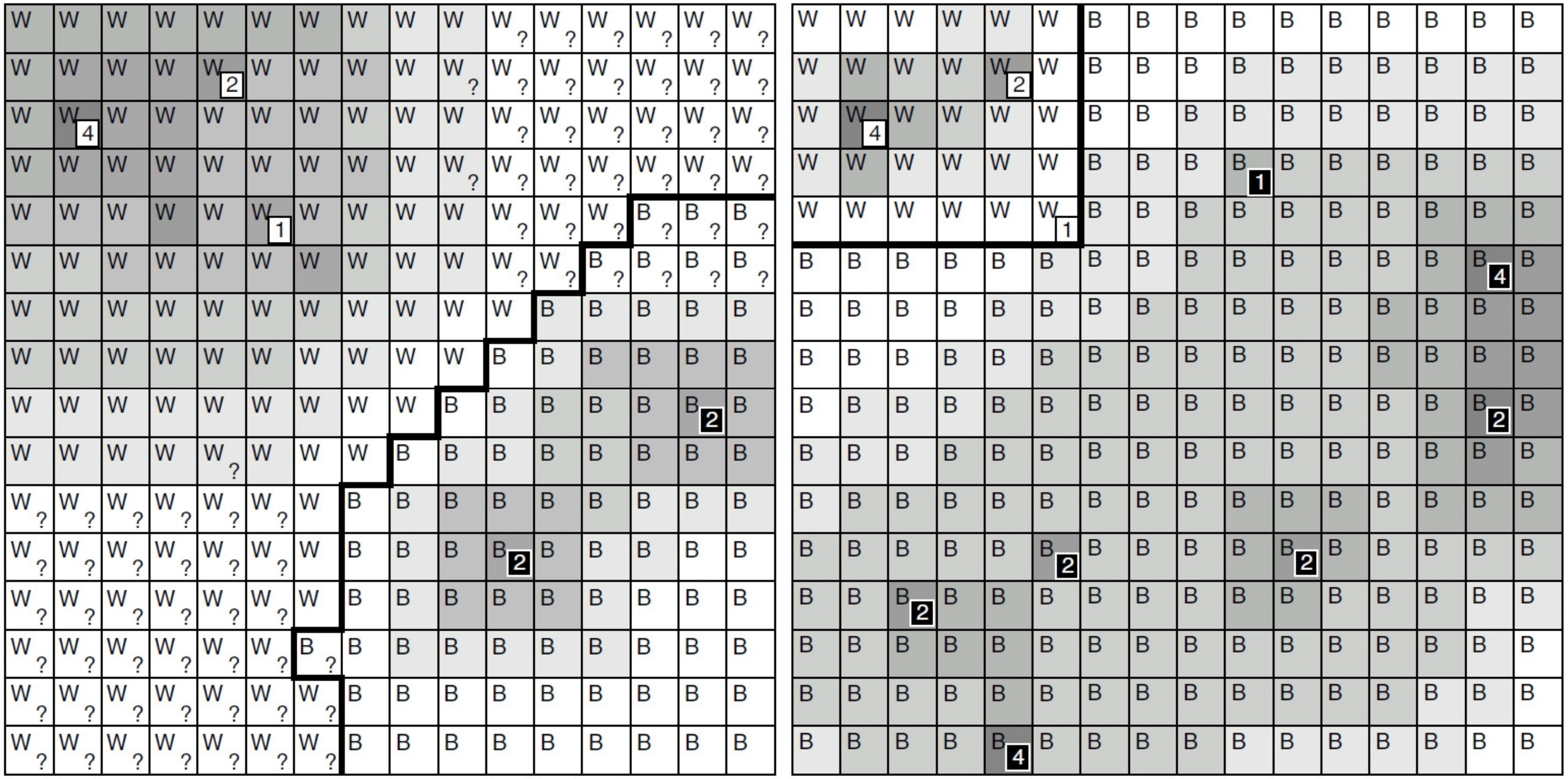
- Influence maps allows AI to see which areas of the game are safe, which to avoid, where the border between teams are weakest
- Example: Security influence map
  - <u>A location is secure if at least four of its eight neighbors are secure</u>.

Dealing with Unknowns

- Typically, *games don't allow players to see all the units in the game*. Vision can be additionally limited by hills and other terrain features – often known as "fog-of-war" (not as in the military-speak term)
- Should AIs have full knowledge of the entire map? Or should they be subjected to "fog-of-war" as well?
- With the partial knowledge, one set of tactical analyses is required per side in the game (incl. all AI sides).

Dealing with Unknowns

- Influence map problems with lack of knowledge



Combining Tactical Analyses

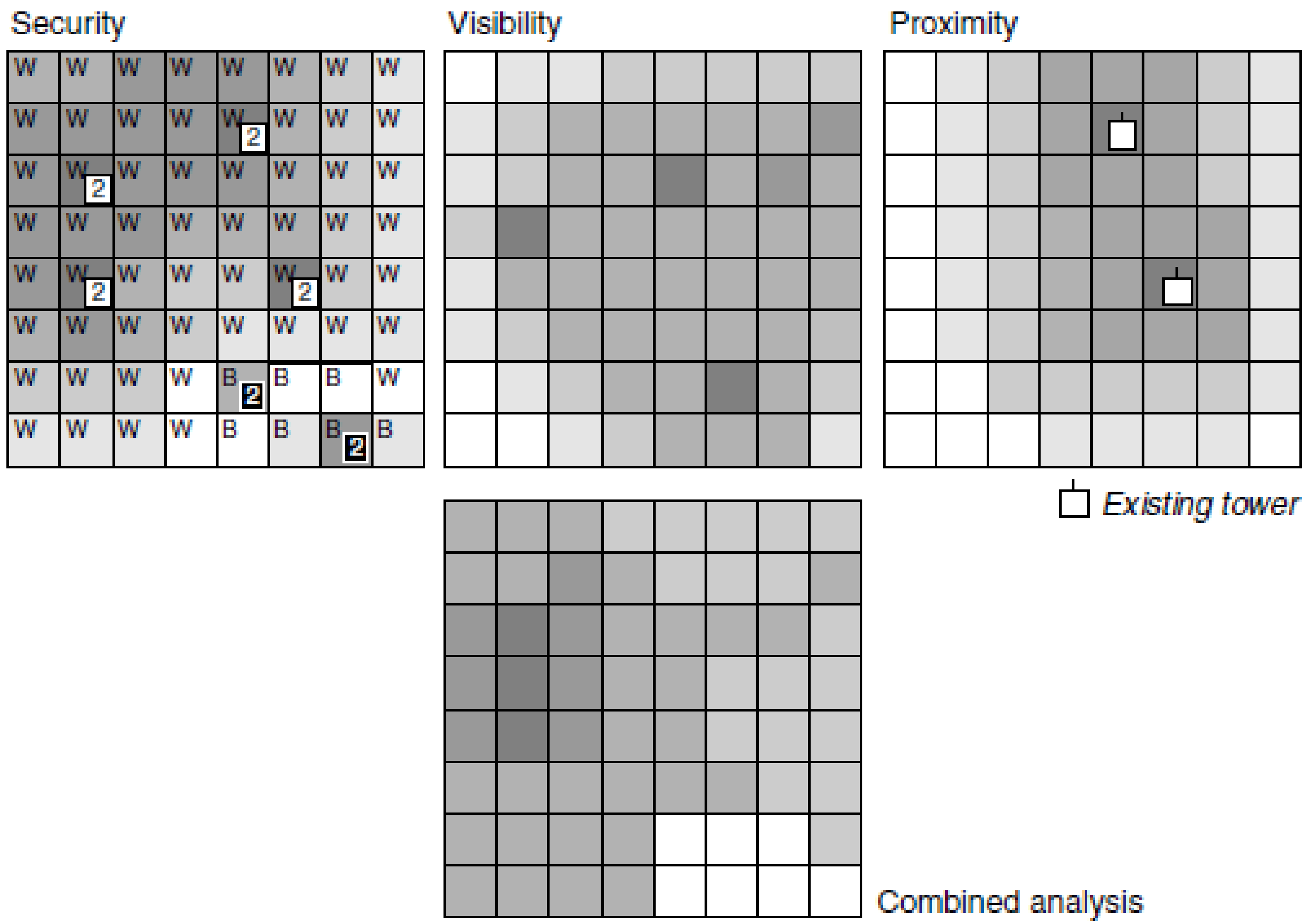
- **Multi-layer analyses** involves combining a few influence maps into a composite influence map.
- Example: To find the best location to build radar tower, consider: wide range of visibility, secured location, far from other towers to avoid redundancy (3 maps)
- To get a single influence value, the **3 base tactical analyses can be combined** by multiplication (or addition, etc.)

$$\text{Quality} = \text{Security} \times \text{Visibility} \times \text{Distance}$$

(or if tower influence is used instead of distance)

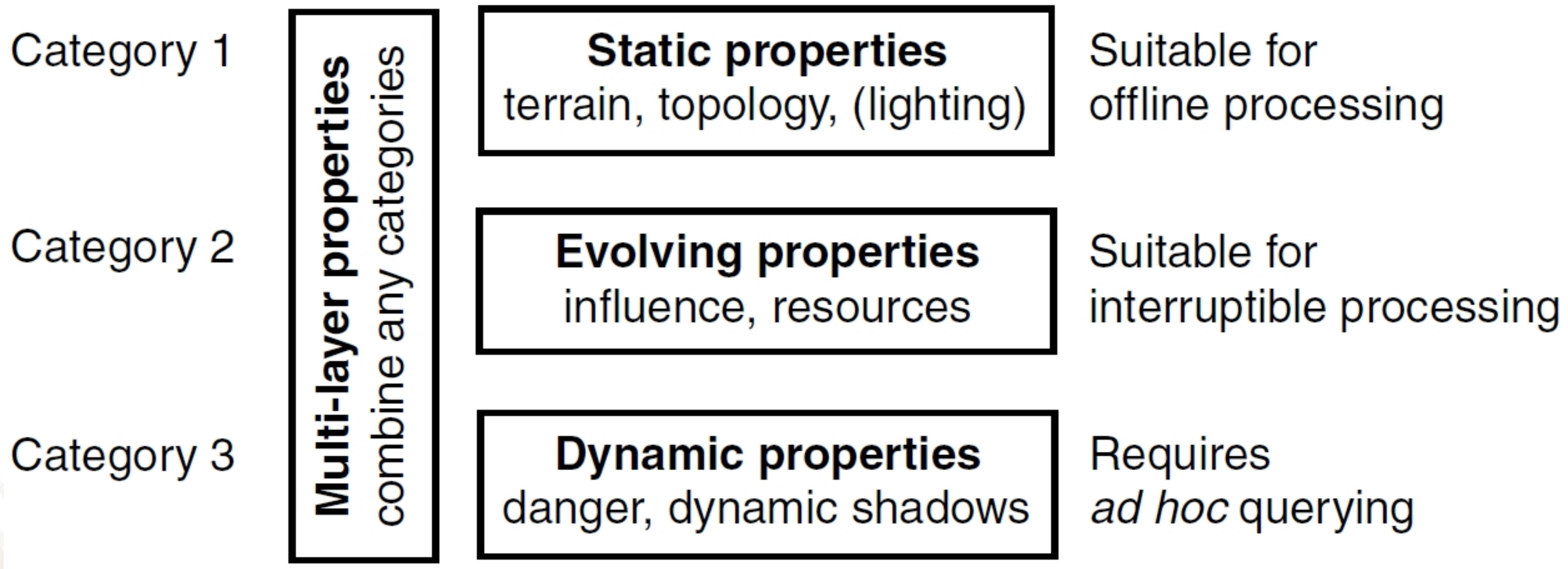
$$\text{Quality} = \frac{\text{Security} \times \text{Visibility}}{\text{Tower Influence}}$$

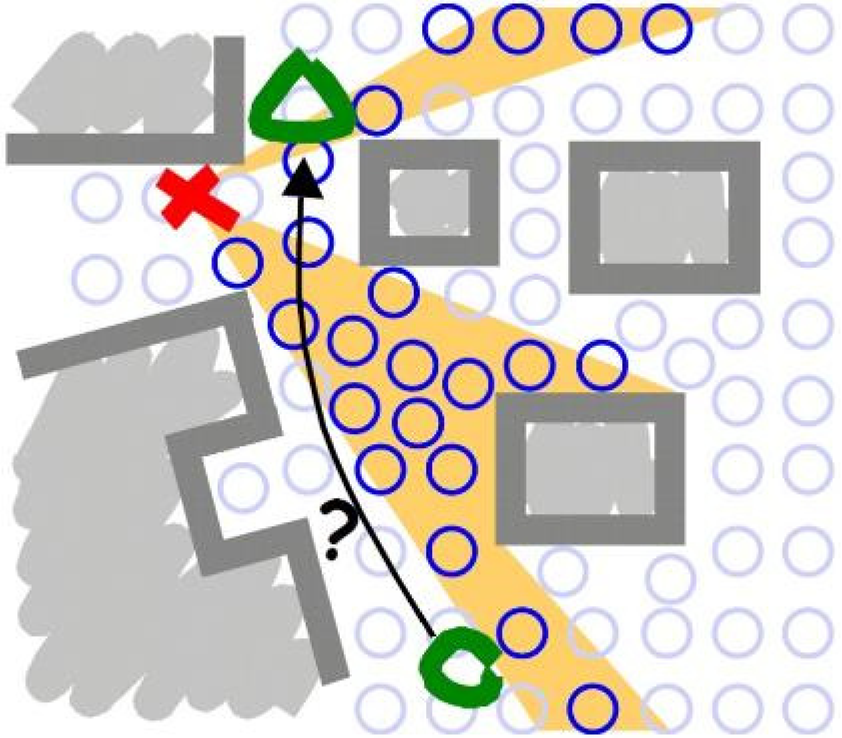
Combining Tactical Analyses

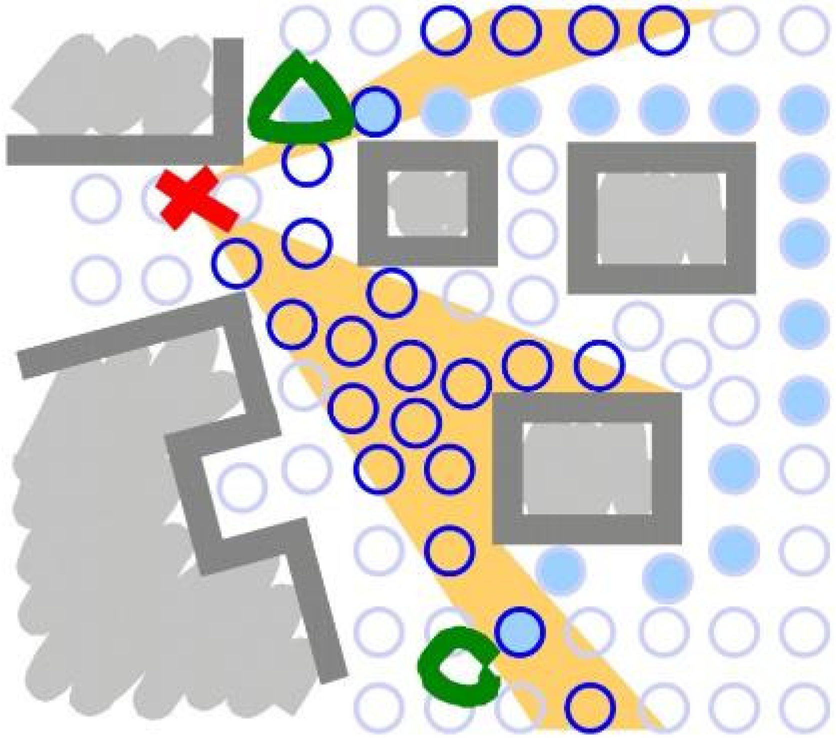


Structure for Tactical Analyses

- Different types of tactical analyses can be distinguished by ***its properties and frequency of updating needed***







(§6.3)

Tactical Pathfinding

Tactical Pathfinding

- **Similar to regular pathfinding (same graph/ techniques/algorithms), *only modification is the cost function used* – extended to tactical info**
- **Cost function influenced by two criteria:**
  - **Distance/time (as before)**
  - **Tactical Information (included now)**
- **Cost of a connection given by a formula**

$$C = D + \sum_i w_i T_i$$

where *D* is the distance/time of connection, w_i is the weighing factor for each tactic T_i and *i* is the number of tactics supported.

Tactical Pathfinding

Example

```
HP: 1273/1300 (Regeneration:
EP: 500/500 (Regeneration: 5)
Panzer: 50 (Organic)
Sicht: 10
Einheiten Zerstört: 0 (Expert: 5
Stop
```

```
MouseXY: 655,366
Render FPS: 241
Update FPS: 40
GameCamera pos: 58.16,20.00,71.00
Time: 11.72
Triangle count: 3301
Vertex count: 4667
Player 0 res: 111 0 500 400 400
Player 1 res: 0 50 500 500 500
```

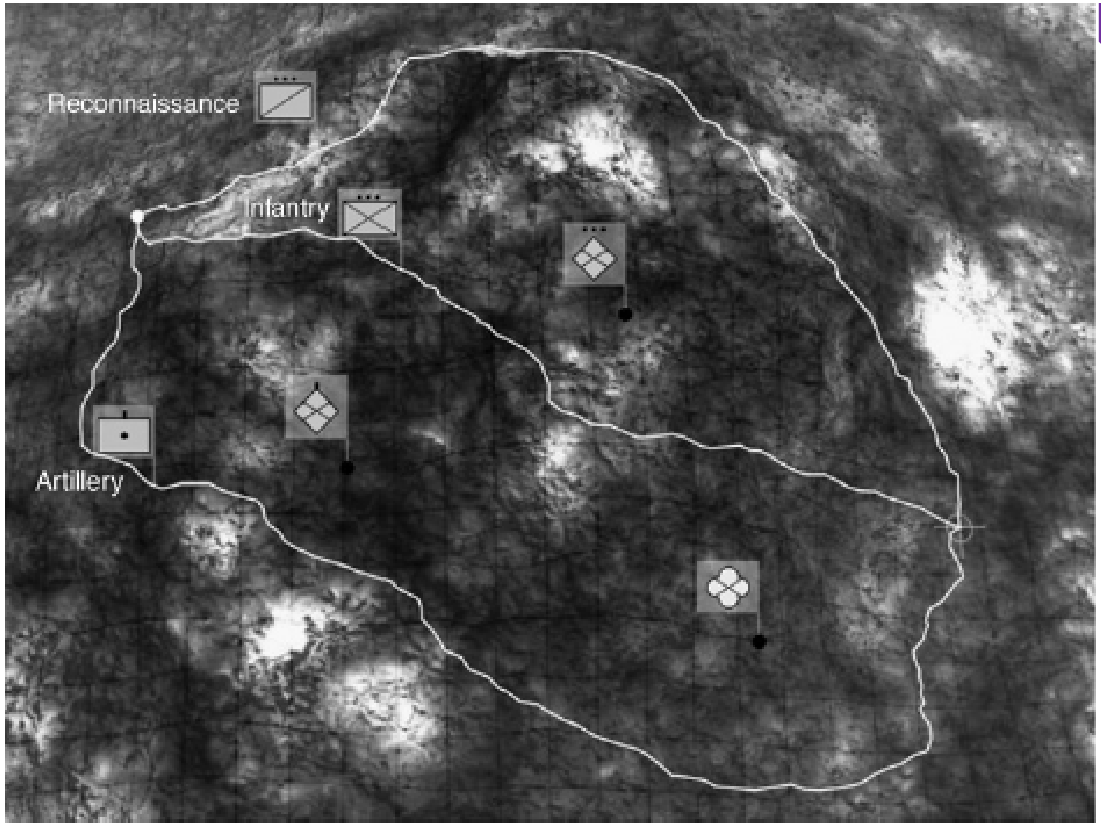
[Demo Link](#)

Tactic Weights & Concern Blending

- In $C = D + \sum_i w_i T_i$, the real-valued quality for each tactic T_i is multiplied by a weighting factor w_i before summing into the final cost value C.
- Locations with high tactics weight will be avoided
- Locations with low tactics weight will be favoured
- Weights can be negative, but careful not to have negative overall weight, which *may result in negative overall cost* of the connection!
- Tactical costs can be pre-calculated if they are static (terrain, visibility). If they are dynamic (military power, number of units), they must be updated from time-to-time.

Customizing Weights

- In certain games, different units can have different sets of tactical weights (w_i) based on their characteristics.



- Example: Reconnaissance units, light infantry, heavy artillery. Tactical info: terrain difficulty, visibility, proximity of enemy units

Customizing Weights

- **Possible weights for previous example:**

	Heavy Artillery	***Reconnaissance***	***Light Infantry***
Terrain	*1.5*	*0.1*	*0.3*
Visibility	*1.0*	*-1.0*	*0.4*
Proximity	*0.5*	*1.0*	*0.5*

- **Weights can also be *dynamically* customized according to a unit's aggression**
- **E.g. Healthy units finds paths in normal way. When it is injured, the weight for proximity to enemy can be increased to make the unit choose a more conservative route back to base.**

Group Pathfinding

- **Pathfinding for a group is typically no more difficult than for an individual character. Game *levels are typically wide enough for a squad to move through*, *etc.***
- **When using tactical pathfinding, it is common to have *a range of different units in a squad*. As a whole they will need to have a different blend of tactical concerns for pathfinding than any individual would have alone.**
  - **Heuristic of the weakest character: the whole squad should use the tactical concerns of their weakest member.**

Group Pathfinding

Example

Terrain Multiplier	*Recon Unit*	*Heavy Weapon*	*Infantry*	*Squad*
Gradient	*0.1*	*1.4*	*0.3*	*1.4*
Proximity	*1.0*	*0.6*	*0.5*	*1.0*

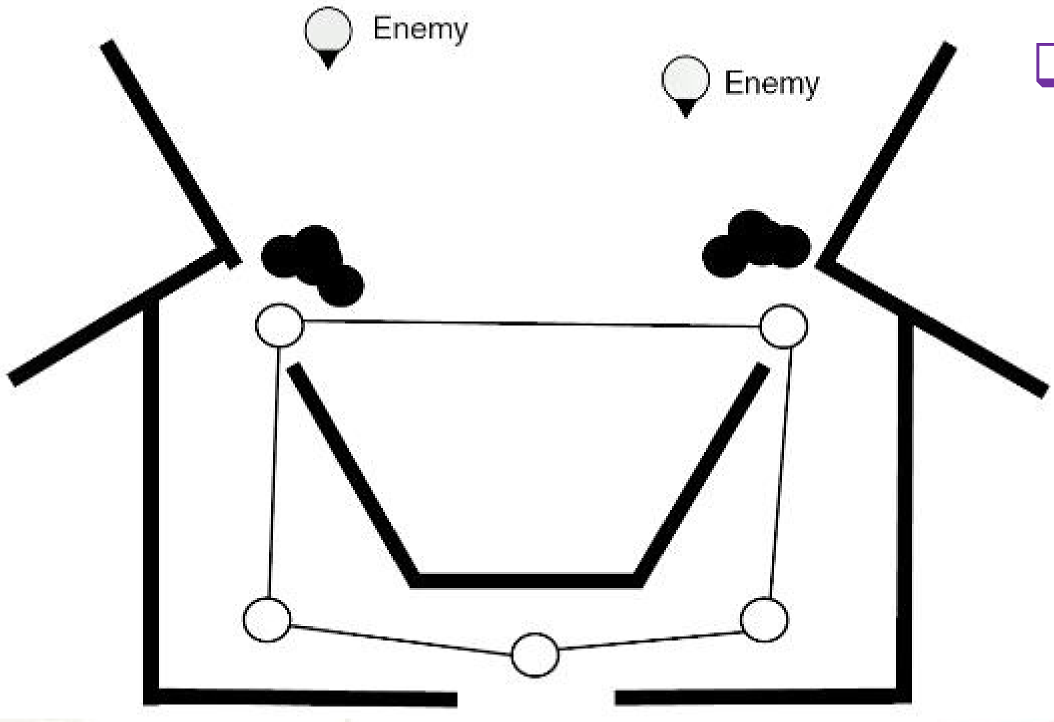
- **We have a recon unit, a heavy weapon unit, and a regular soldier unit in a squad.**
  - **The recon unit tries to avoid enemy contact, but can move over any terrain.**
  - **The heavy weapon unit tries to avoid rough terrain, but doesn't try to avoid engagement.**
- **To make sure the whole squad is safe, we *try to find routes that avoid both enemies and rough terrain*.**

Tactical Graphs for Pathfinding

- Influence maps (or any other kind of tactical analysis such as waypoint tactics) are ideal for guiding tactical pathfinding.
- But the locations alone are not sufficient for pathfinding. We also need a record of the *connections between them*.
- We can generate connections by running movement checks or line-of-sight checks between waypoints or map locations.
- The most common graph for tactical pathfinding is the grid-based graph used in RTS games.

Connection Tactical Costs

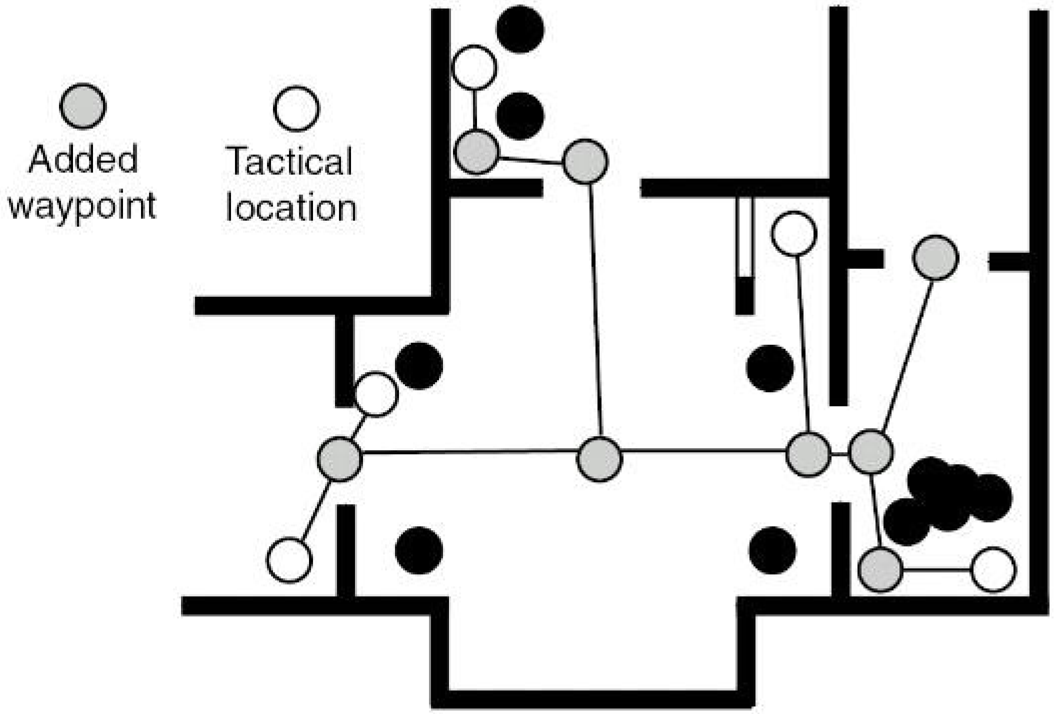
- Tactical information is **normally stored on a per-location basis**. To convert location-based information into connection-based costs, we **normally average the tactical quality of each of the locations** that the connection connects.



- This assumption is good enough for most games, although it ***sometimes produces quite poor results***.

Using Tactical Waypoints

- **Tactical waypoints**, **unlike tactical analysis maps,** ***have tactical properties that refer to a very small area of the game level*****. A** ***small movement from a waypoint may produce a dramatic change*** **in the tactical quality of the location.**



- **To make sensible pathfinding graphs it is almost always necessary to add additional waypoints at locations that do not have particular tactical properties.**

Waypoints and Pathfinding Graph

- The simplest way: **superimpose the tactical waypoints** onto a **regular pathfinding graph**.
  - The tactical locations need to be linked into their adjacent pathfinding nodes, but the **basic graph provides the ability to move easily between different areas of the level**.
- Typically developers will ***include the placement of tactical waypoints into the same level design process*** used to place nodes for the pathfinding (e.g., using Dirichlet domains for quantization).
  - **Graph** can be used for both ***simple tactical decision making*** and for ***full-blown tactical pathfinding***

(§6.4)

Coordinated Action

Coordinated Action

- **To coordinate multiple characters to cooperate together to get their job done, some structure need to be in place. Two categories:**
  - **Team/Group AI (a group of AI NPCs, fully AI)**
  - **Cooperative AI (AI cooperates with a human player in a team)**
- **Common Qs:**
  - **Should individual AIs "speak" to each other, and make collective decisions?**
  - **Should a central "command center/brain" give orders and instructions to each individual AI?**
  - **Can we have a bit of both?**

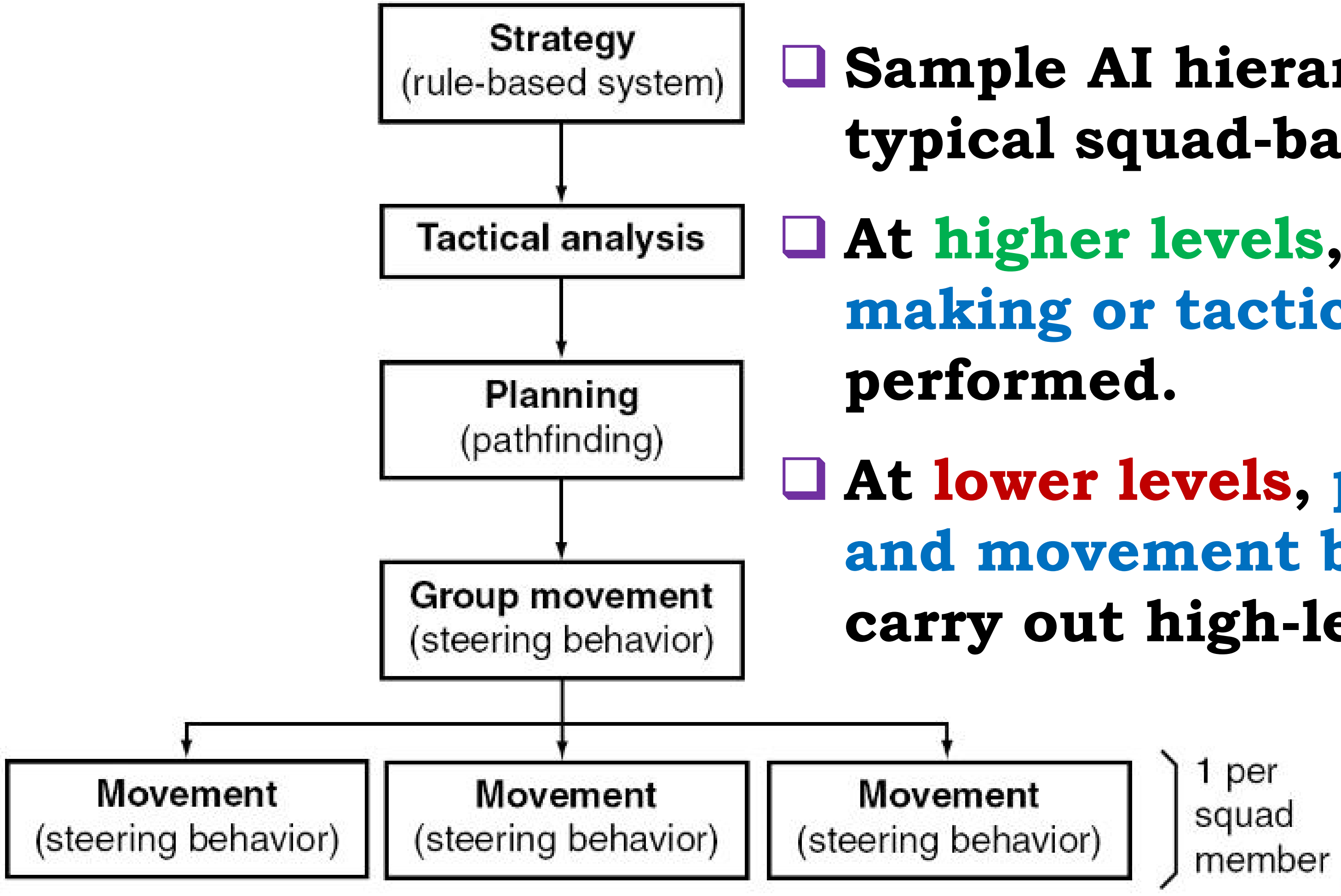
Multi-Tier AI – Top-Down Approach

- **Highest level AI makes a decision, passes it down to next level, which uses its instruction to make its decision, and pass again down to the lowest level**

- **Example: Military Hierarchy**
- **<u>Most often used</u>.**

Multi-Tier AI

Structural Example



Strategy (rule-based system)
Tactical analysis
Planning (pathfinding)
Group movement (steering behavior)
Movement (steering behavior)
Movement (steering behavior)
Movement (steering behavior)
1 per squad member

- **Sample AI hierarchy for a typical squad-based shooter.**
- **At higher levels, decision making or tactics are performed.**
- **At lower levels, pathfinding and movement behaviors carry out high-level orders**

Multi-Tier AI – Bottom-Up Approach

- ***Lowest level AI algorithms take their own initiative to make decisions*****, then use higher level algorithms to provide information on which they can base their actions**
- **Example: Autonomous decision making by individual characters that can influence the overall game, Squad-based Strategy games, Evolution-based games**
- **"Emergent cooperation"**

Group Decisions

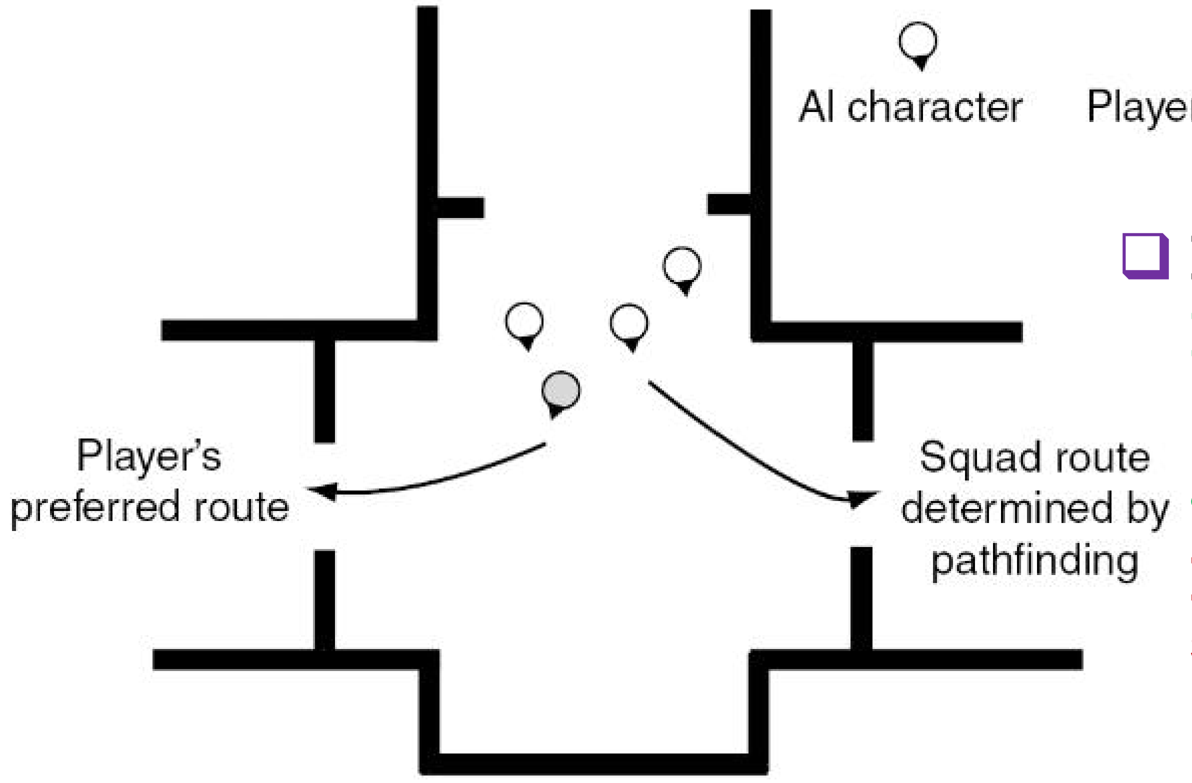
- The decision making tools used are just the same as those we saw in Week 4. There are *no special need for a group decision making algorithm*. It takes input about the world and comes up with an action.
- *At the highest level it is often some kind of strategic reasoning system* such as decision tools (that include tactical analysis to determine best places to move, apply cover, stay undetected, *etc*.)
- The difference is in the way its actions are carried out. Rather than being scheduled for execution by the character, they typically take the form of orders that *are passed down to lower levels in the hierarchy*.

Group Movement

- **The formation steering system we looked at in Week 2 is multi-tiered.**
  - ***At the higher levels the system steers the whole squad or even groups of squads*.**
  - ***At the lowest level individual NPCs move in order to stay with their formation*, while avoiding local obstacles /taking into account their environment.**
- **It is worth having some feedback from the movement algorithm that the decision making system can take account of, in case, *e.g.*, *the movement algorithms cannot reach a particular location* the high-level AI decides to attack.**

Including the Player

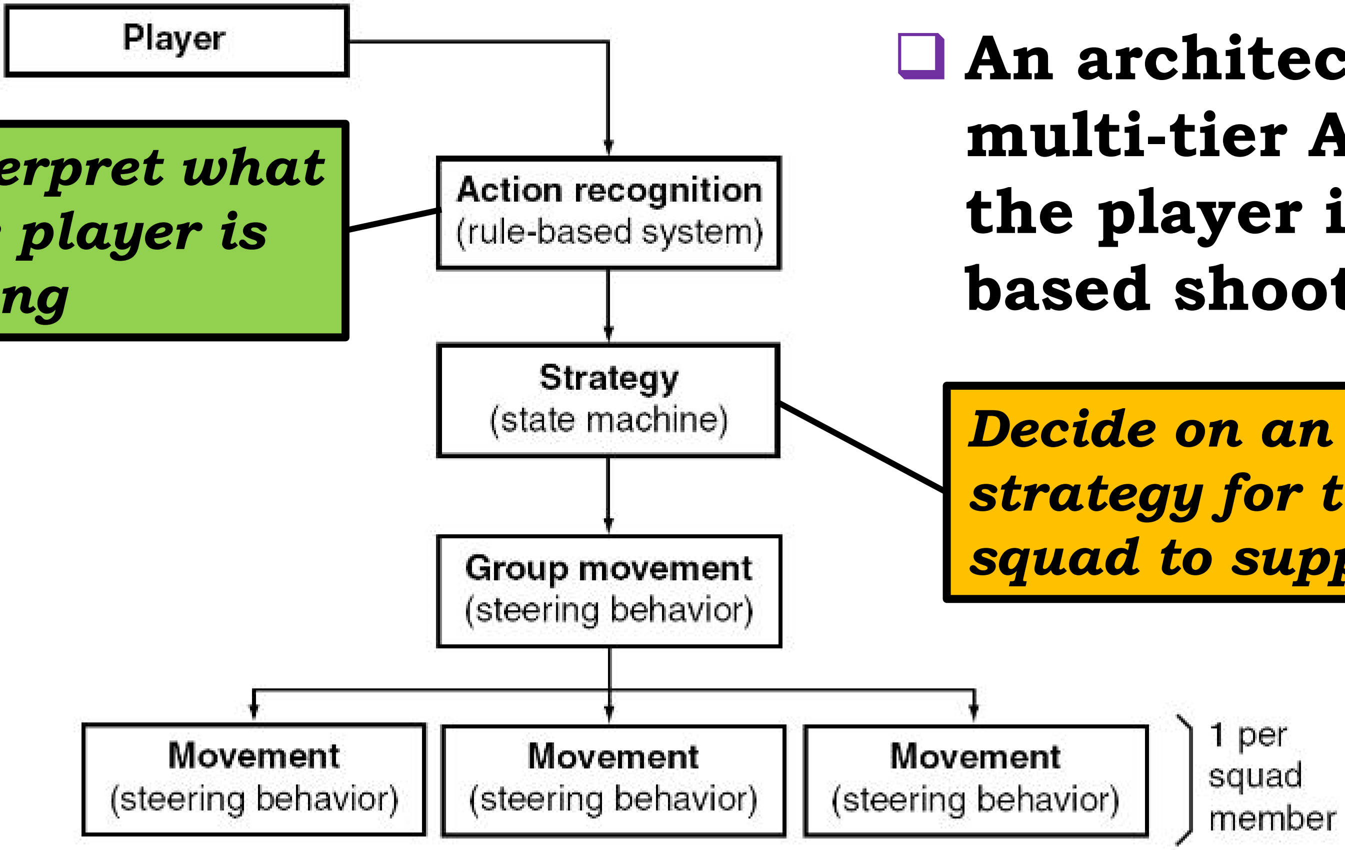
- **While multi-tier AI designs are excellent for most squad- and team-based games, they *do not cope well when the player is part of the team*.**



- **E.g., here the high-level decision making has made a decision that the player accidentally subverts**

Including the Player

- In general, the *player will always make the decisions for the whole team*.



Player
Action recognition (rule-based system)
Strategy (state machine)
Group movement (steering behavior)
Movement (steering behavior)
Movement (steering behavior)
Movement (steering behavior)
1 per squad member

Interpret what the player is doing

- An architecture for a multi-tier AI involving the player in a squad-based shooter:

Decide on an overall strategy for the whole squad to support player

Explicit Player Orders

- A different approach to including the player in a multi-tiered AI is to give them the ability to schedule *specific orders*. This is the way that an RTS game (*e.g.*, StarCraft) works.
- On the player's side, the player is the top level of AI. They get to decide the orders that each character will carry out.
- Lower levels of AI then take this order and work out how best to achieve it.
- The intent-identification problem is so difficult that *incorporating some kind of explicit player orders* into squad-based games is a good alternative.

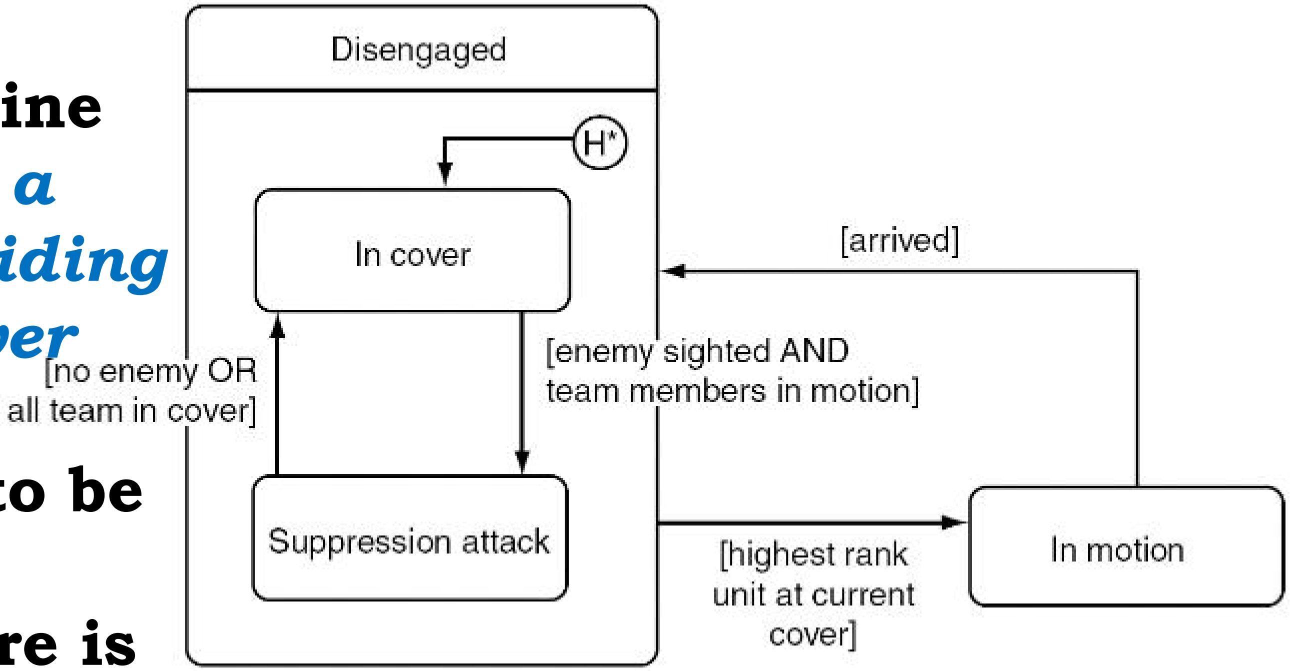
Emergent Cooperation

- There is a weakness on relying on the quality of the high-level decision: If a character cannot obey the higher level decision for some reason, then it is left without any ability to make progress.
- Instead use less centralized techniques: have each character take into account what each other is doing, they *can appear to act as a coherent whole*.
  - the hope is that the *behavior of the overall team (group) will emerge to be intelligent*
- This is the approach taken in most squad-based games.

Emergent Cooperation

Example

- Below is an example finite state machine for four characters in a fire team. Four characters with this finite state machine *will act as a team, providing mutual cover* and appearing to be a coherent whole. There is no higher level guidance being provided.


Disengaged
H*
In cover
[no enemy OR all team in cover]
[enemy sighted AND team members in motion]
Suppression attack
[arrived]
[highest rank unit at current cover]
In motion

Emergent Cooperation

Scalability

- As you add more characters to an emergently cooperating group, you will reach a threshold of complexity. Beyond this point it will be difficult to control the behavior of the group. *The exact point* where this occurs depends on the complexity of the behaviors of each individual.
- Solutions:
  - simplify the rules that each character is following;
  - set up a multi-tiered AI with different levels of emergent behavior.

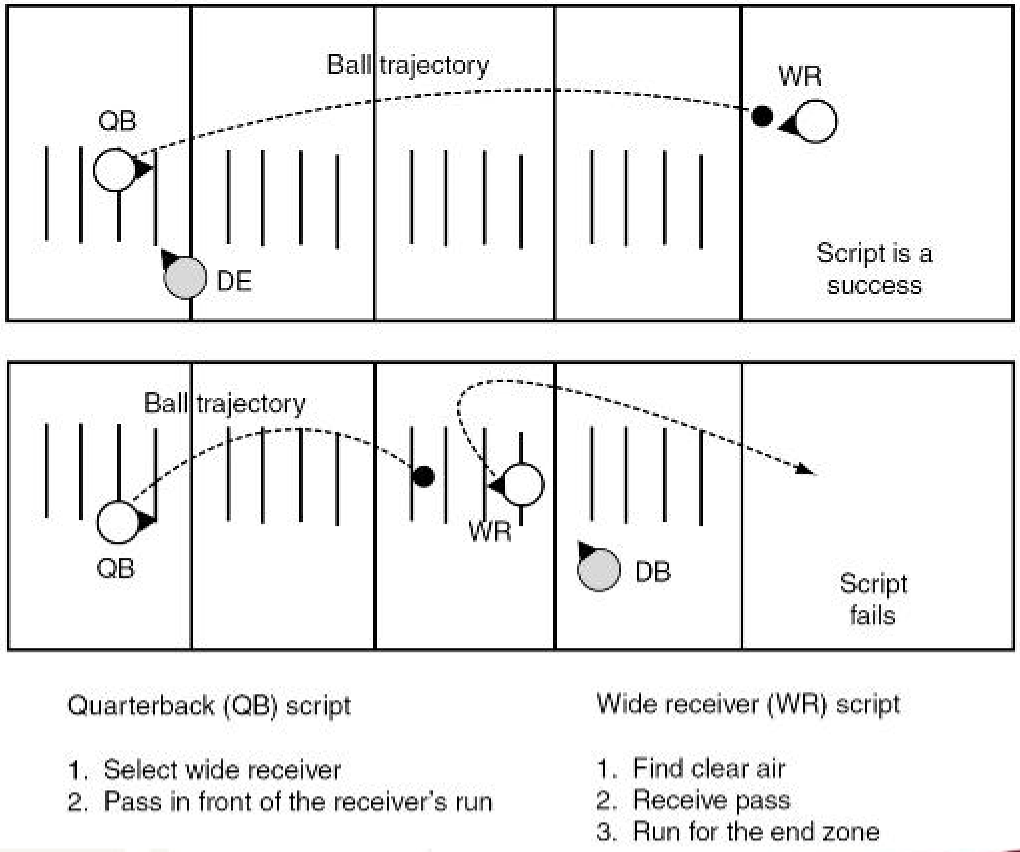
Emergent Cooperation

Predictability

- A side effect of this kind of emergent behavior is that you *often get group dynamics that you didn't explicitly design*.
  - Good: see emergent intelligence in the group, but this doesn't happen very often
  - Bad: the group starts to do something really annoying that looks unintelligent.
- If you are looking for highly intelligent high-level behavior, then you will always end up implementing (e.g., scripting) it explicitly.

Scripting Group Actions

- To make sure that all the members of a group work together, use a script that shows what actions need to be applied in what order and by which character.
- With a script per character, there are timing complications that make it difficult to keep the illusion of cooperation among several characters.



Quarterback (QB) script

1. Select wide receiver
2. Pass in front of the receiver's run

Wide receiver (WR) script

1. Find clear air
2. Receive pass
3. Run for the end zone

Scripting Group Actions

- To make cooperative scripts workable, we need to add the notion of interdependence of scripts.
- The actions that one character is carrying out need to be synchronized with the actions of other characters.
- We can achieve this most simply by using signals
- In place of an action in the sequence, we allow two new kinds of entity:
  - Signal - a message sent to anyone else who is interested.
  - Wait - stops the script until it receives a matching signal.

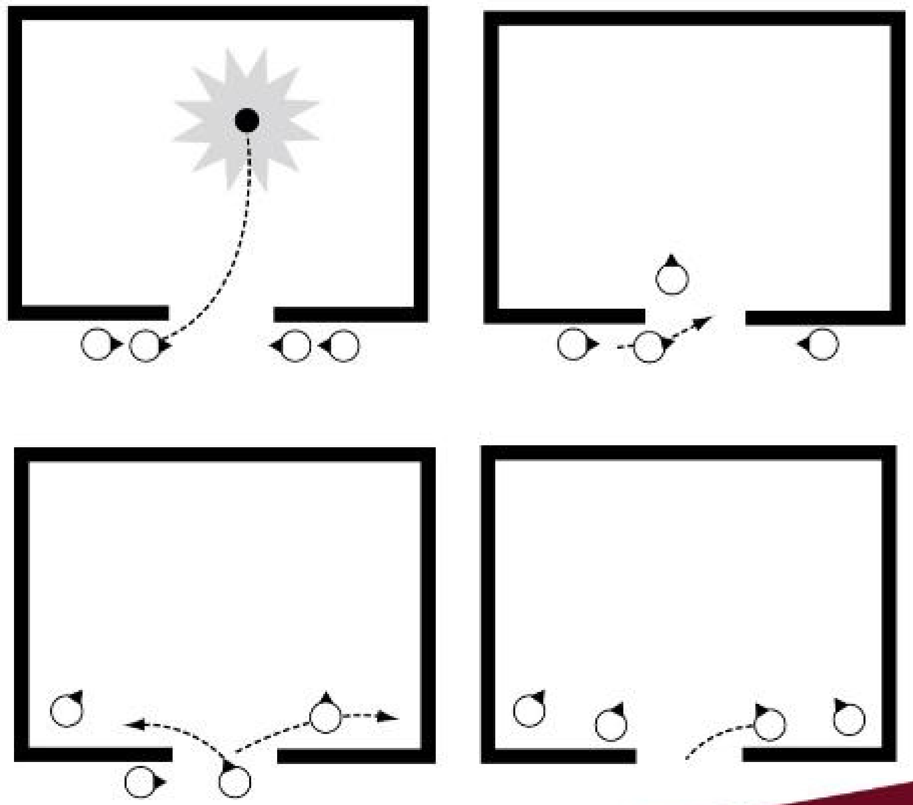
Military Tactics

- The implementation of military-style tactics is most suited to a *cooperation script approach*, *rather than open-ended multi-tier or emergent AI*
- A set of scripts can be created that represents the individual stages of the operation, and these can then be *made into a higher level script that coordinates the lower level events*.
- As in all scripted behaviors, some feedback is needed *to make sure the behaviors remain sensible* throughout the script execution.

Military Tactics

Case Study

- The following military tactical situation (securing a room) uses four scripts:
  - Move into position outside the door.
  - Throw in a grenade.
  - Move into a corner of the room.
  - Flank the inside of the doorway.
- The ***interactions between the scripts are controlled using Signal and Wait*** instances



Military Tactics

Case Study

- By using a waypoint tactics system to predetermine all the corners in all the rooms, the <u>**same script to be used on buildings of all different shapes**</u>