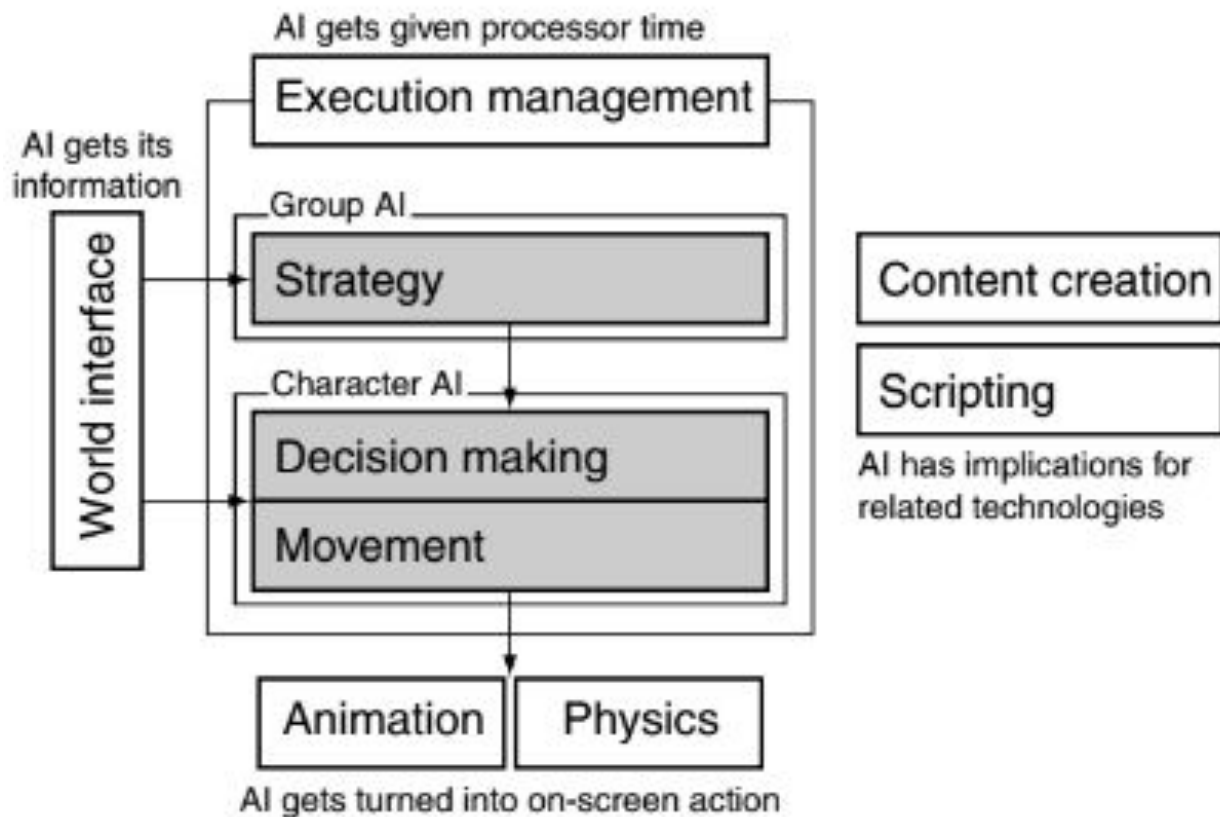




# Artificial Intelligence

Tactics and Strategy



## Game AI: The Model

# Strategic AI



# Strategic AI

We will consider Strategic AI...

1. Taking decision as a group (could the AI sacrifice an individual ?)
2. Terrain analysis (where the AI sends its sniper ?)
3. Player analysis (Where the player is deploying its units ?)
4. Tactical Pathfinding



# Group Tactics

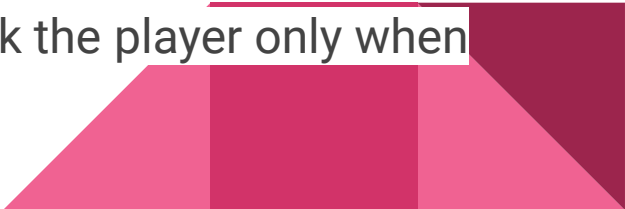
There are many ways of doing it! It always come down to the specific game:

**Coordination:** *Synchronized behavior change between 2 or more agents while the game context makes sense.*

- How would you have two AI shake hands ?

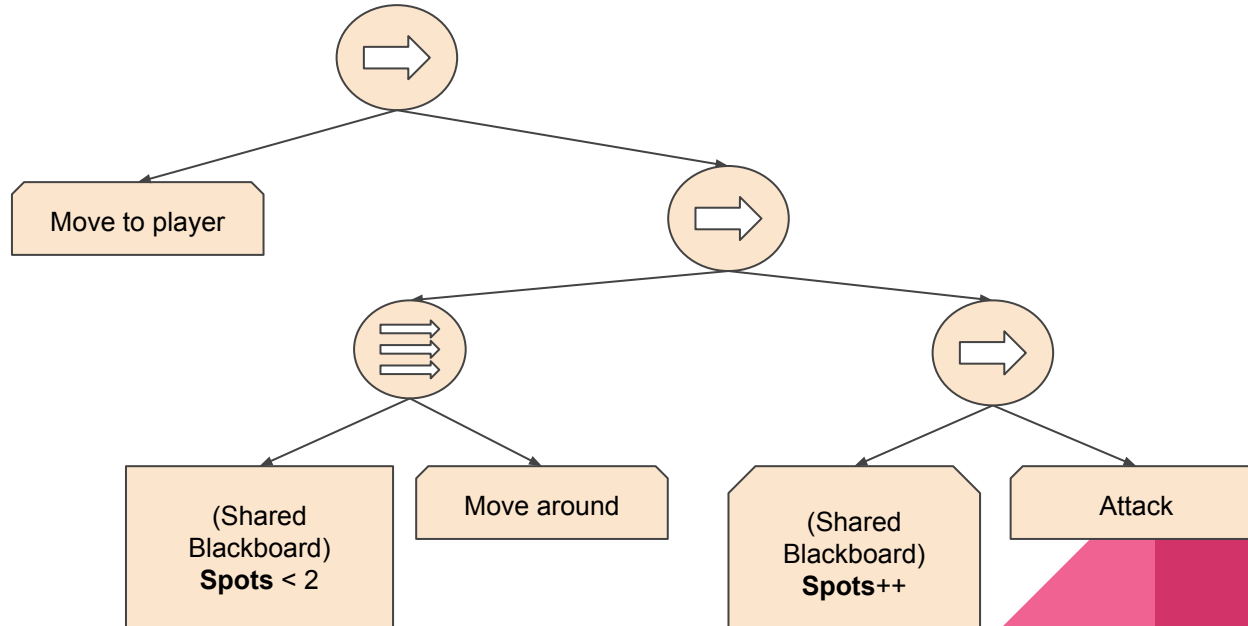
**Reactive Coordination:** How would you limit the amount of AIs attacking the player at once ?

**Candidate Accumulation:** How would you decide to attack the player only when you have enough units?



# Group Tactics: Reactive Coordination

How would you limit the amount of AIs attacking the player at once with BTs



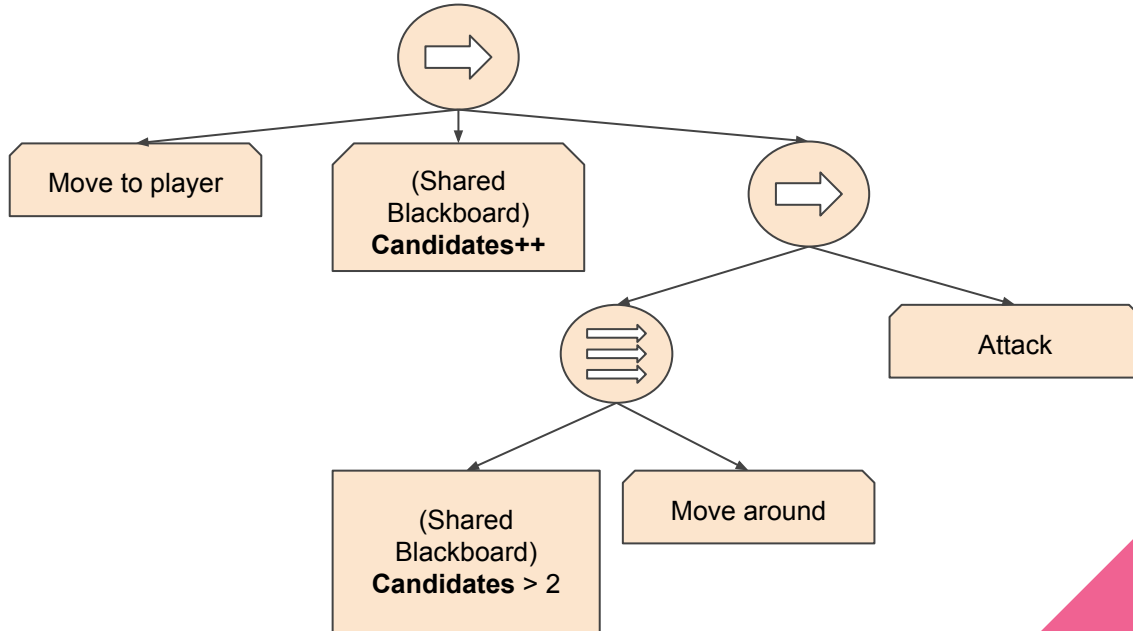
# Group Tactics: Reactive Coordination

- A lot can be achieved within the Shared Backboard
- Famous Half Life AI based its group AI on that
- Be careful to get into race conditions
- This limitation can be achieved easily with Smart Objects (Utility AI):
  - Each available spot to attack player is a *SmartObject*
  - It provides the action to attack the player to fulfill a specific goal
  - Once in use the *SmartObject* is gone until freed



# Group Tactics: Candidate Accumulation

How would you decide to attack the player only when you have enough units?





# Group Tactics: Candidate Accumulation

- This could create another set of problems
  - We might want to have only the first two to attack
  - Even in this case, which agents should be picked up ?
- This could be made to work inside a BT but could create a lot of complexity
- Easier to have a “Tactics Manager”:

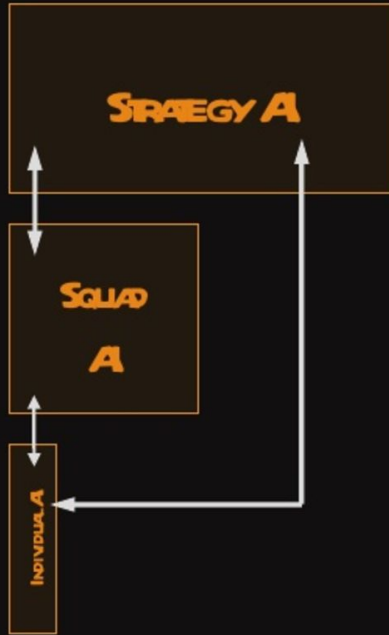
Tactic	Min Agents	Max Agents	Pick criteria	Instances	Current Candidates
Swarm	1	3	Shortest dist	2	{...}
Flank	2	2	Farthest dist	1	{...}

# Group Tactics: Advanced

- If group tactics are core to the gameplay we must separate the strategic decision taking from the individual decision taking
- Some of the best approach mix different decision taking algorithms:
  - Strategic AI: planner doing terrain analysis with influence maps
  - Tactic AI: BT or planner
  - Individual AI: BT or FSM



# Group Tactics: Killzone 2 Multiplayer bots



- Strategy to Squad: Orders: Defend, Advance, ...
- Squad to Strategy: Feedback: Order failed
- Squad to Individual: Orders: MoveTo
- Individual to Squad: Combat information
- Strategy to Individual: Orders: Assassination target
- Individual to Strategy: Request reassignment

# Terrain Analysis

- It allows to drop markup in the level to find out where to:
  - Take Cover, sniper positions, gather a resource, higher / defensive ground, etc ...
- There are mainly three ways of doing it:
  - Level Designer placed (manual markup)
  - Automatic placement as a precalculation
  - Automatic placement while playing (dynamic)
- We end up doing all of them, just to solve different problems



# Terrain Analysis: Manual Placement



# Terrain Analysis: Automatic Placement



# Terrain Analysis: Dynamic Placement



# Terrain Analysis

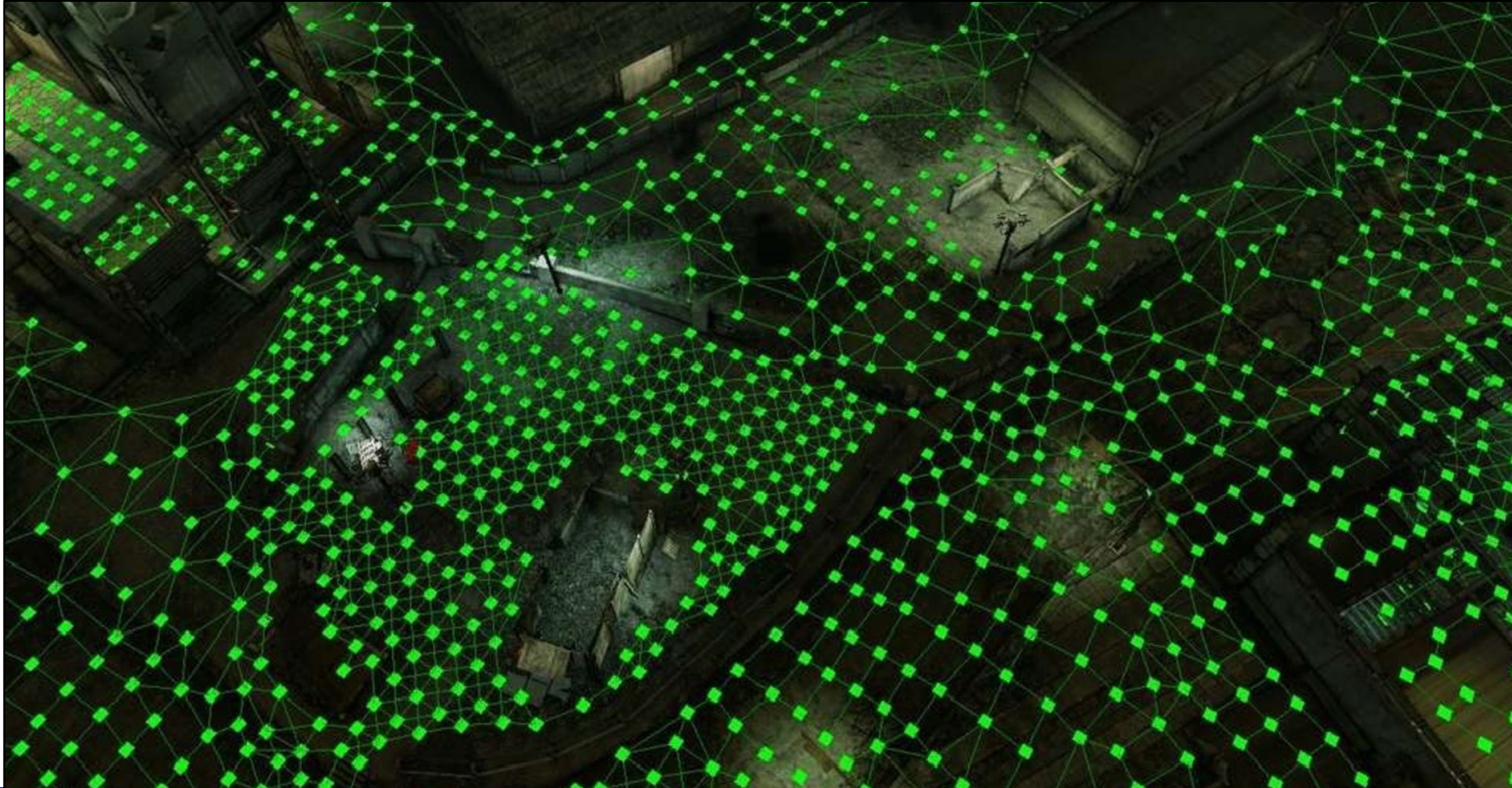
Best solution end up being:

- Automatic generation of elements in the level
- Allow level designers to change them in any way they see fit
- Have a limited subset of GameObjects capable of producing dynamic markup
- What other element of the terrain could be interesting besides cover ?

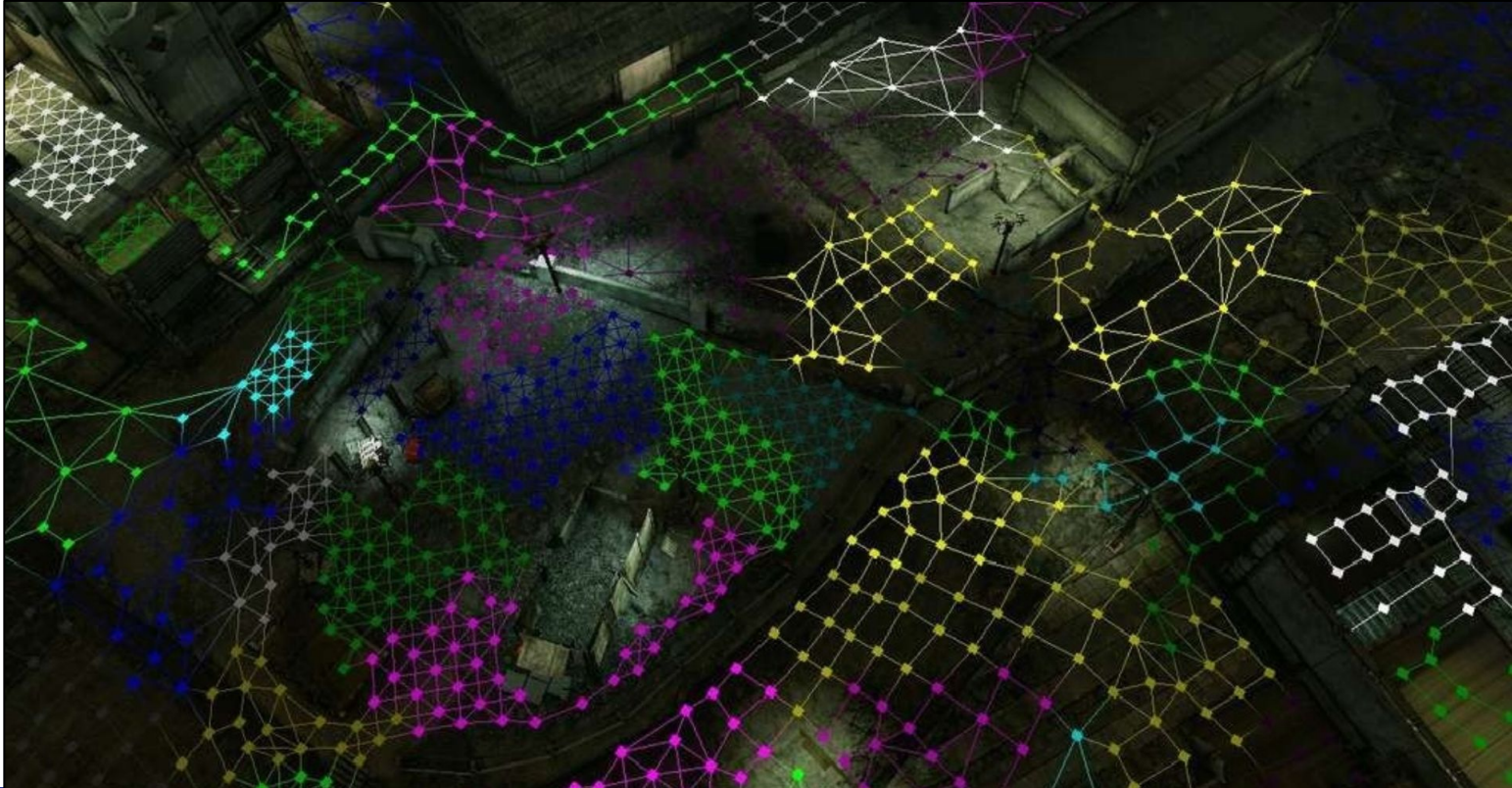
*A good example of this system is the **Recast** library for pathfinding*



# Automatic Terrain Analysis in Killzone 2



# Automatic Terrain Analysis in Killzone 2





# Automatic Terrain Analysis in Killzone 2

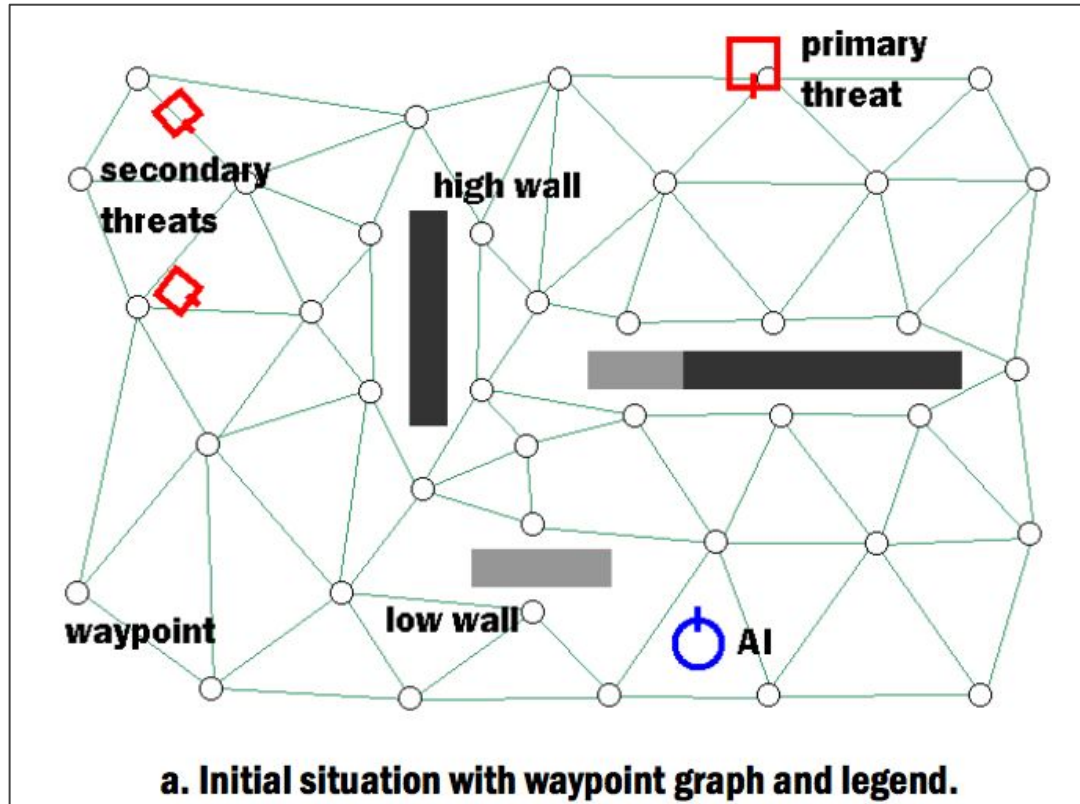


# Dynamic Terrain Analysis

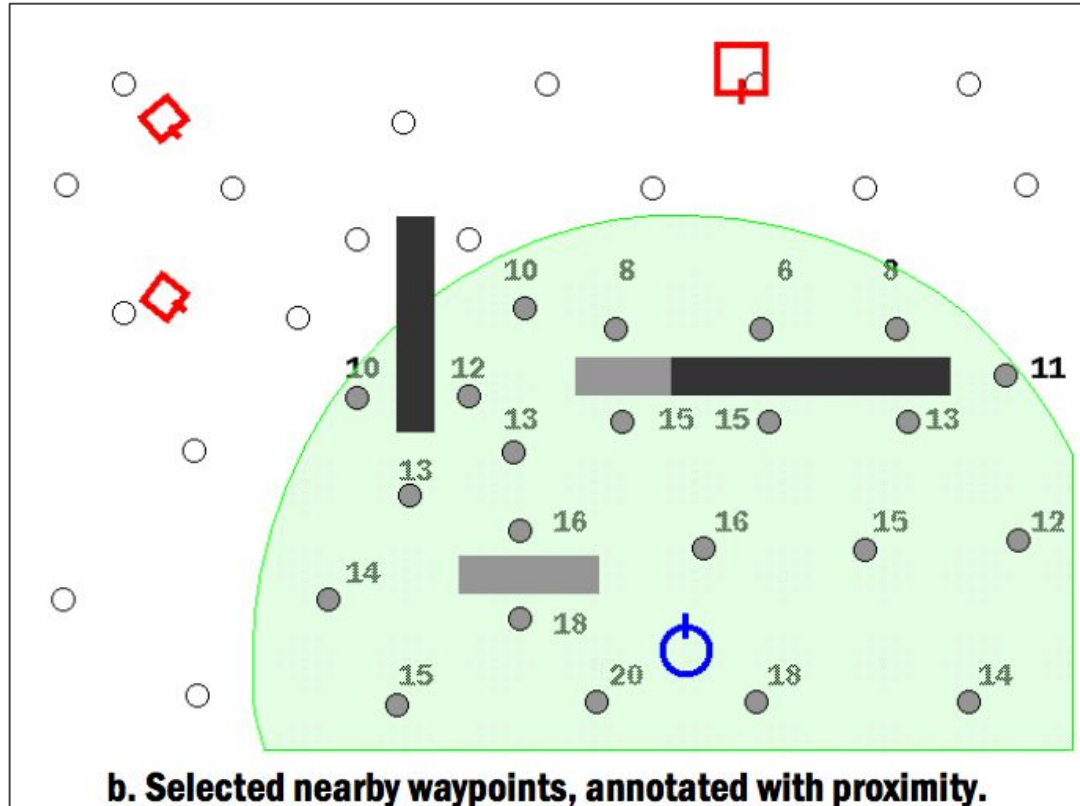
- It is done at the individual level of decision taking
- If we have a good pre-calculation it could be about turning on/off markup
- It is about calculating areas of effect for different enemies
- Then annotating “value” of each possible spot to go
- Be careful: situation changes very fast in certain games!
- Balancing out weight node values is tough!



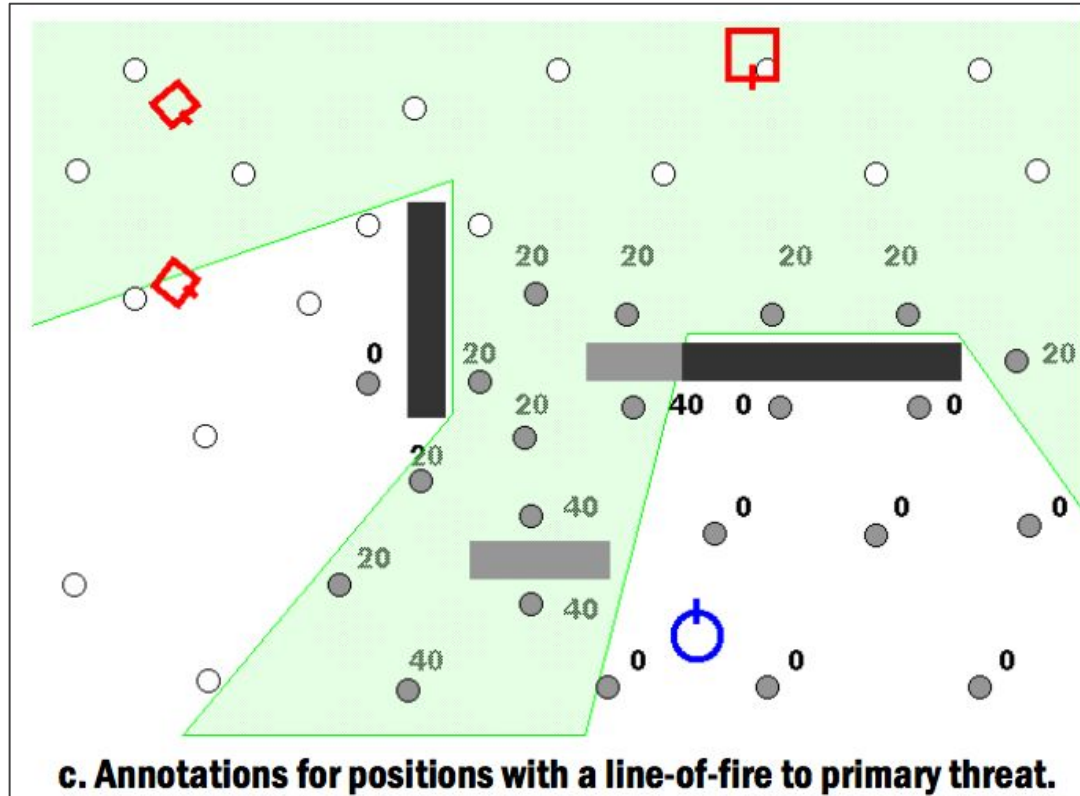
# Example: Attack Primary, defend from others



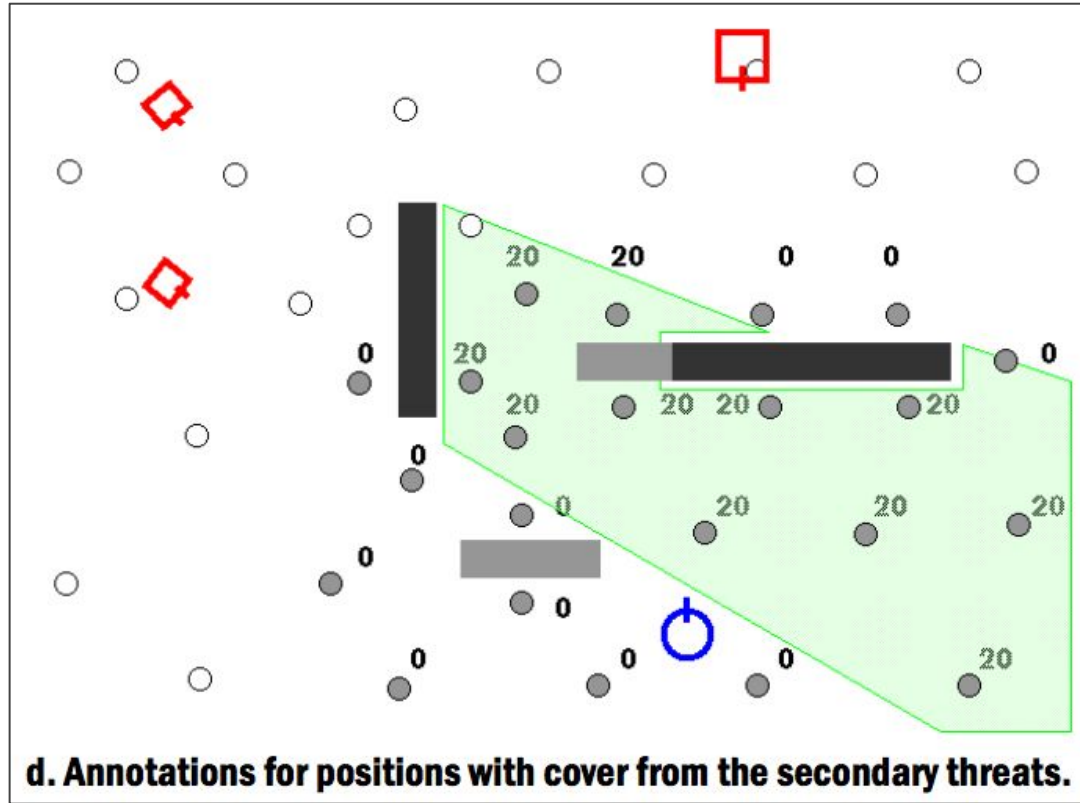
## Example: Attack Primary, defend from others



# Example: Attack Primary, defend from others

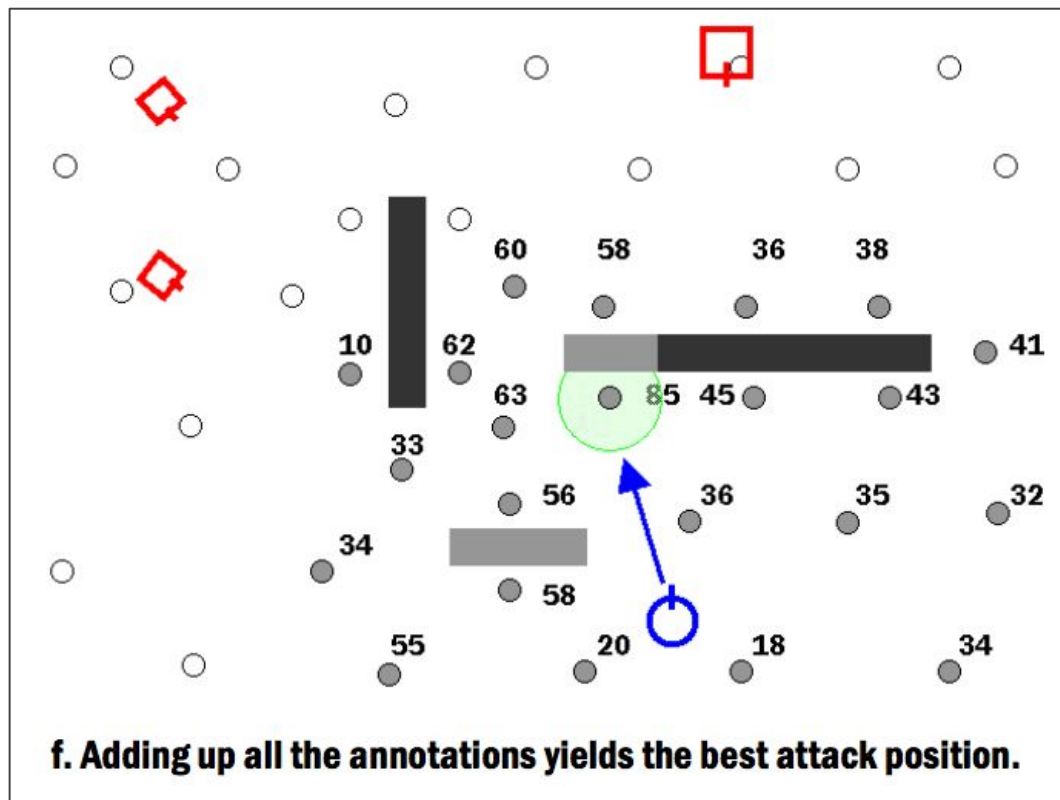


# Example: Attack Primary, defend from others





## Example: Attack Primary, defend from others



# Other criteria

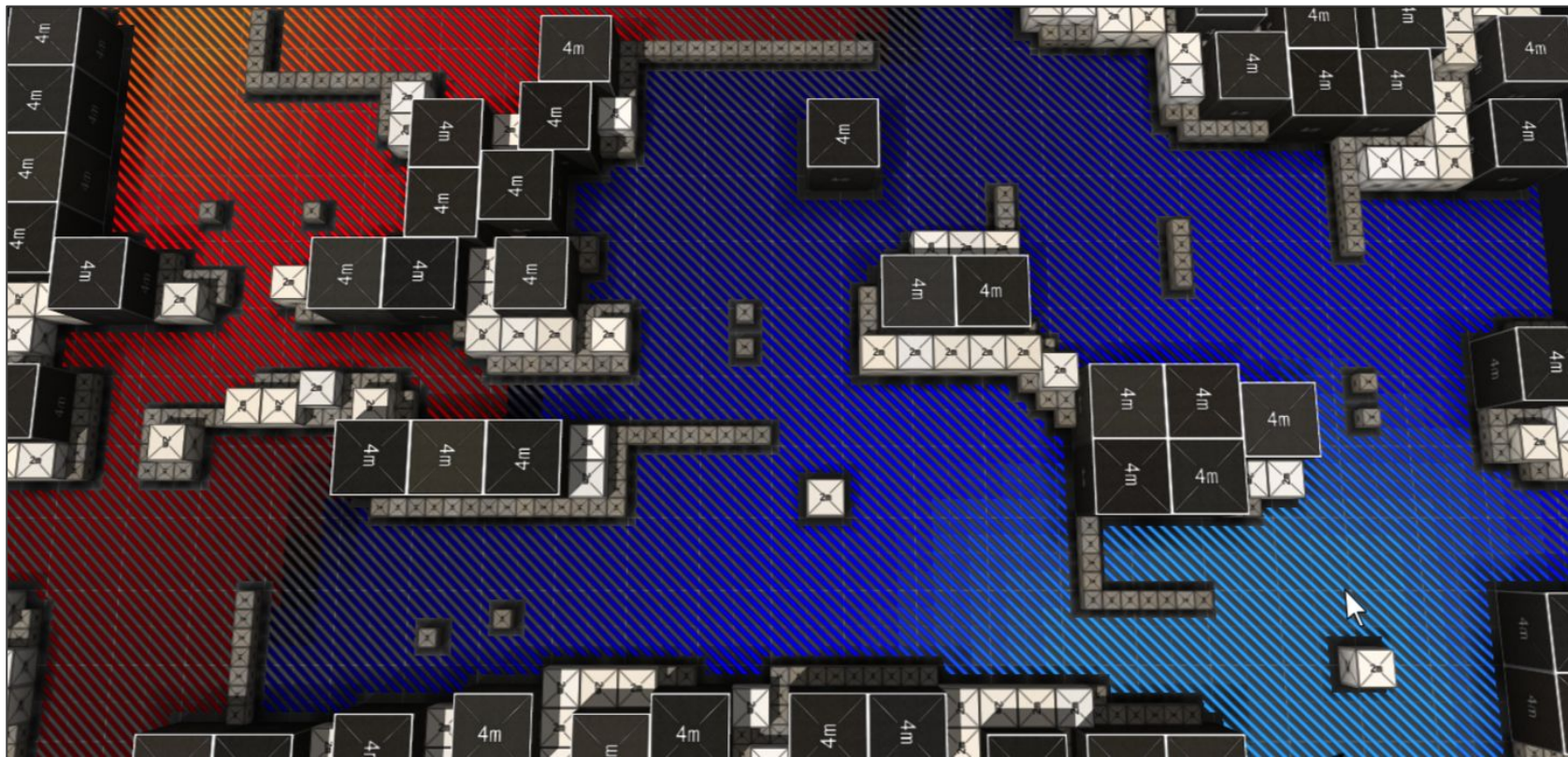
Position evaluation input	Description: (The evaluation function awards a higher score for ...)
Proximity to current position	being closer to or more quickly reached from the current position
Proximity from specific location	being closer to or more quickly reached from a specific location
Cover from primary threat	offering cover from the primary threat (given a stance)
Line-of-fire to primary threat	offering cover from the primary threat (given a stance)
Distance to primary threat	being preferred fighting distance from the primary threat
Outside danger zone	being outside the blast range of (expected) projectile
Cover from secondary threats	offering cover from one or more specified threats other than the primary threat
Outside friendly line-of-fire	being outside the line-of-fire of specified friendly units
Distance from friendly positions	being some distance away from friendly positions
Wall hugging	being close to a wall or obstacle in a specified direction
Nearby cover	being near positions offering cover from the primary threat
Player line-of-fire	not being a position across the player's line-of-fire from the current position
Preferred fighting range	being inside the preferred fighting range

# Influence Maps

- Runtime analysis of a map graph
- It weights the influence of each faction as a number
- Then smooths out it's influence based on distance
- Allow to quickly decide who controls what part of the map:
  - Where are the “borders” of the territories
  - Where the enemy has been before ?
  - Which area is most likely to be attacked next ?

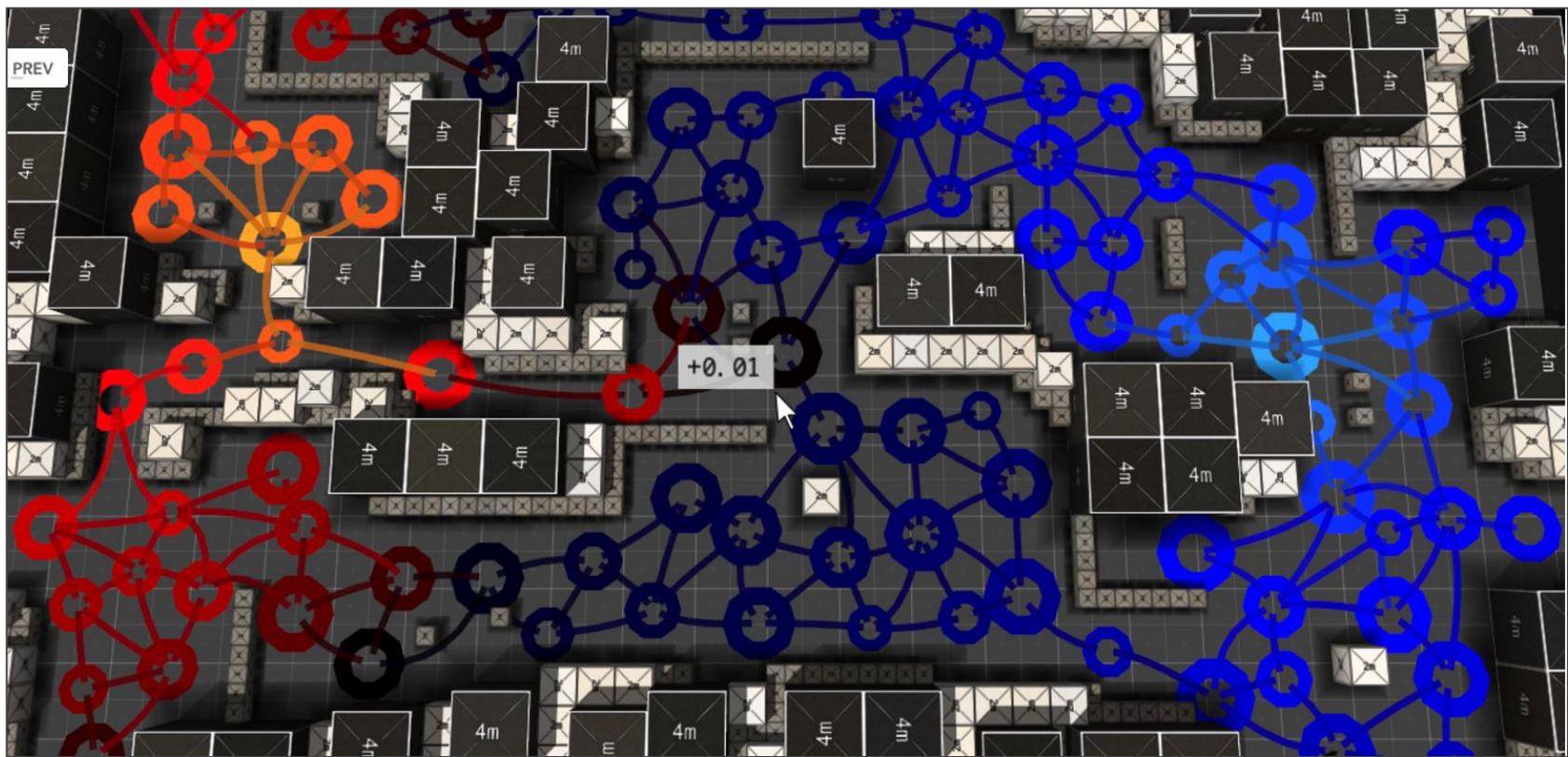


# Influence Maps with tiles





# Influence Maps with waypoints



# Influence Maps in Killzone 2

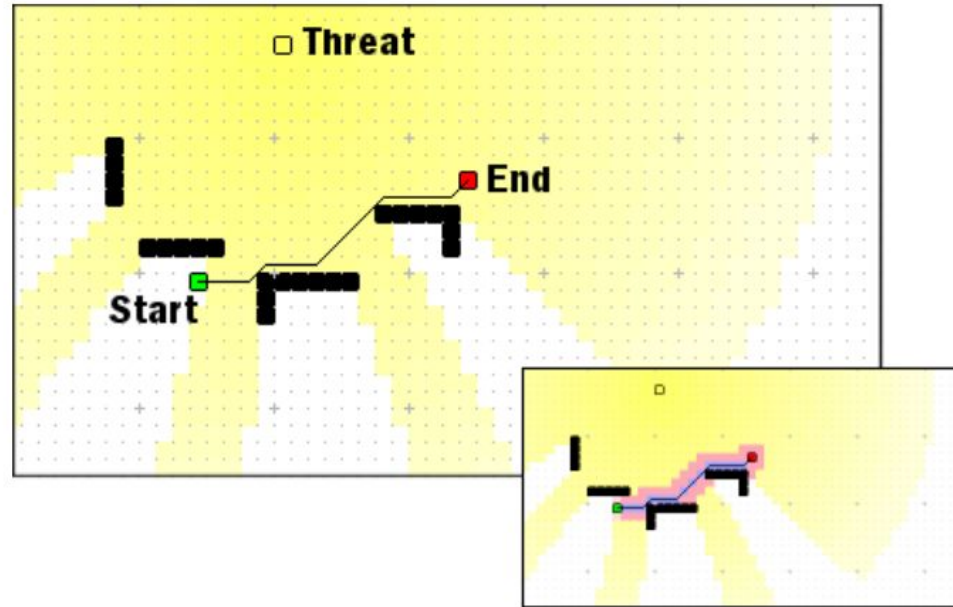


# Tactical Pathfinding

- In which situation the shortest path is not the best path ?
  - We want to move being as safe as possible!
- How we could use all previous information to improve pathfinding ?
- Influence maps and dynamic waypoint weight can feedback the A\*
- We just add weight to nodes based on previous criteria
- Again, situations change very quickly, be careful!



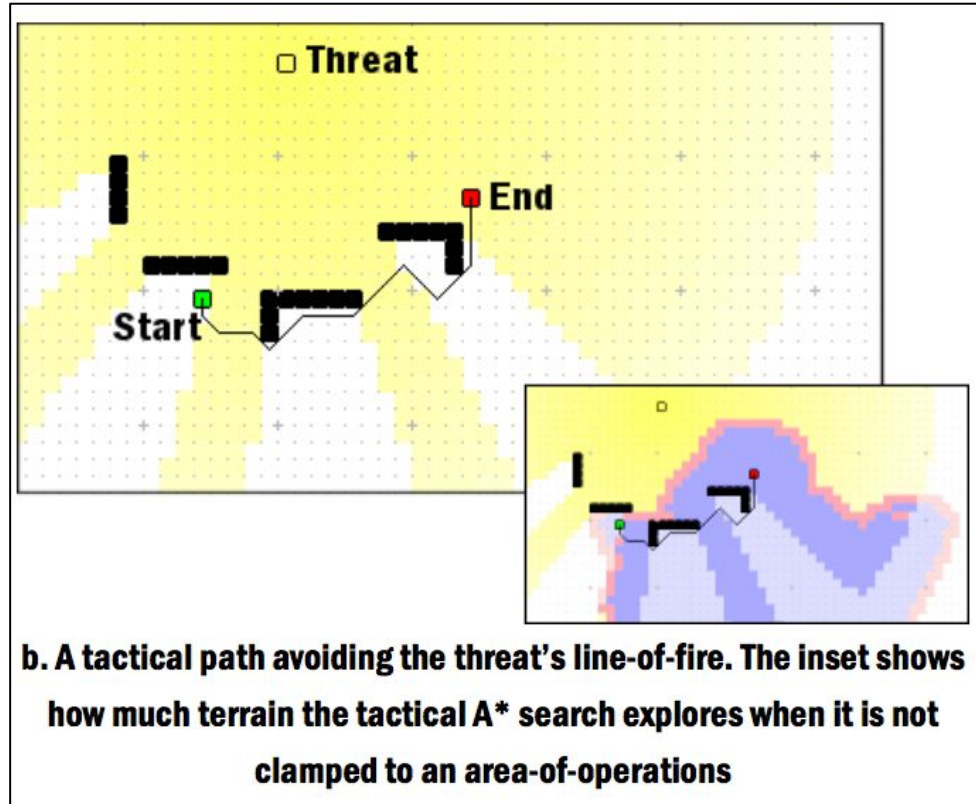
# Tactical Pathfinding



**a. A shortest path ignoring the threat's line-of-fire. The inset shows how little terrain is explored by the shortest-path A\* search.**



# Tactical Pathfinding



# References

- Google Tech Talk on [AI on Civilization](#)
- “Building a Better Battle: The Halo 3 AI Objectives System” [audio](#) and [slides](#)
- Killzone 2 AI [coverage](#) and also [here](#) (on multiplayer bots [here](#))
- AI Gamedev article on [influence maps](#) (and video [here](#))

