

Lecture 21

Sensing & Perception

Recall: Sense-Think-Act

- **Sense:**
 - Perceive the world
 - Reading the game state
 - **Example:** enemy near?
- **Think:**
 - Choose an action
 - Often merged with sense
 - **Example:** fight or flee
- **Act:**
 - Update the state
 - Simple and fast
 - **Example:** reduce health

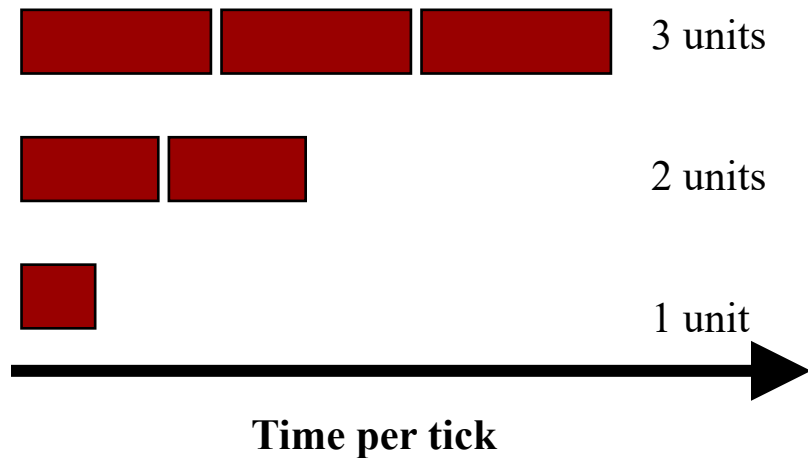


Take Away for this Lecture

- Sensing as the primary bottleneck
 - Why is sensing so problematic?
 - What types of things can we do to improve it?
- Optimized sense computation
 - Can we improve sense computation performance?
 - Can we share sensing between NPCs?
- Sense event matching
 - What are events and how are they represented?
 - What is the advantage of an event system?

Recall: Sensing Performance

- Sensing may be slow!
 - Consider *all* objects
- Example: morale
 - n knights, n skeletons
 - Knights fear skeletons
 - Proportional to # seen
- Count skeletons in view
 - $O(n)$ to count skeletons
 - $O(n^2)$ for all units



Recall: Sensing Performance

- Sensing may be slow!
 - Consider *all* objects

- Example: morale

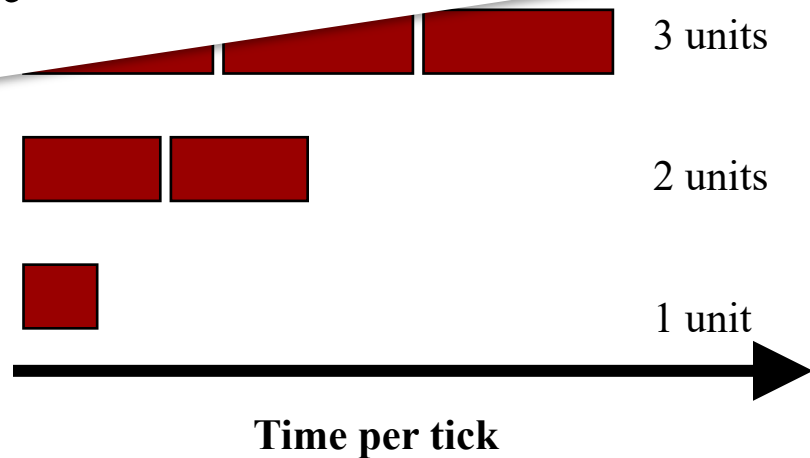
- n knights, n skeletons
- 1 knight can see n skeletons
- n knights can see n skeletons

- Count skeletons in view

- $O(n)$ to count skeletons
- $O(n^2)$ for all units



How Do We Make it Faster?



Example: Collision Detection

Naively $O(n^2)$

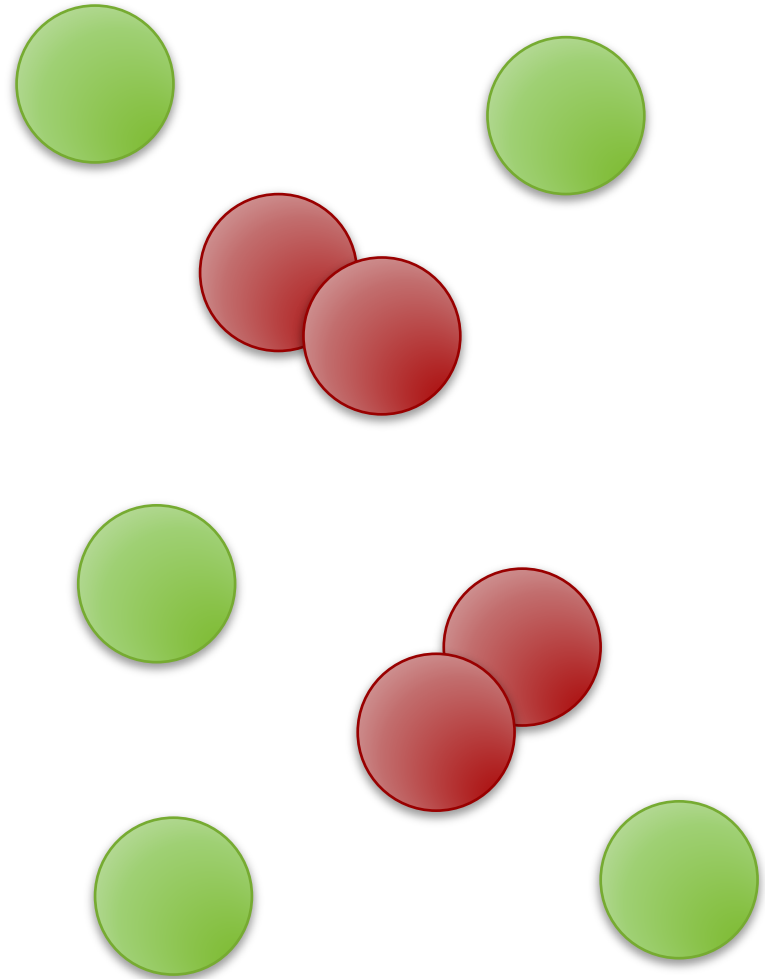
for each object x:

for each object y:

if x not y and x, y collide:

resolve collision of x, y

Checks objects obviously
far apart from each other



Example: Collision Detection

Lab Optimization

for each object x:

put x into cell slot

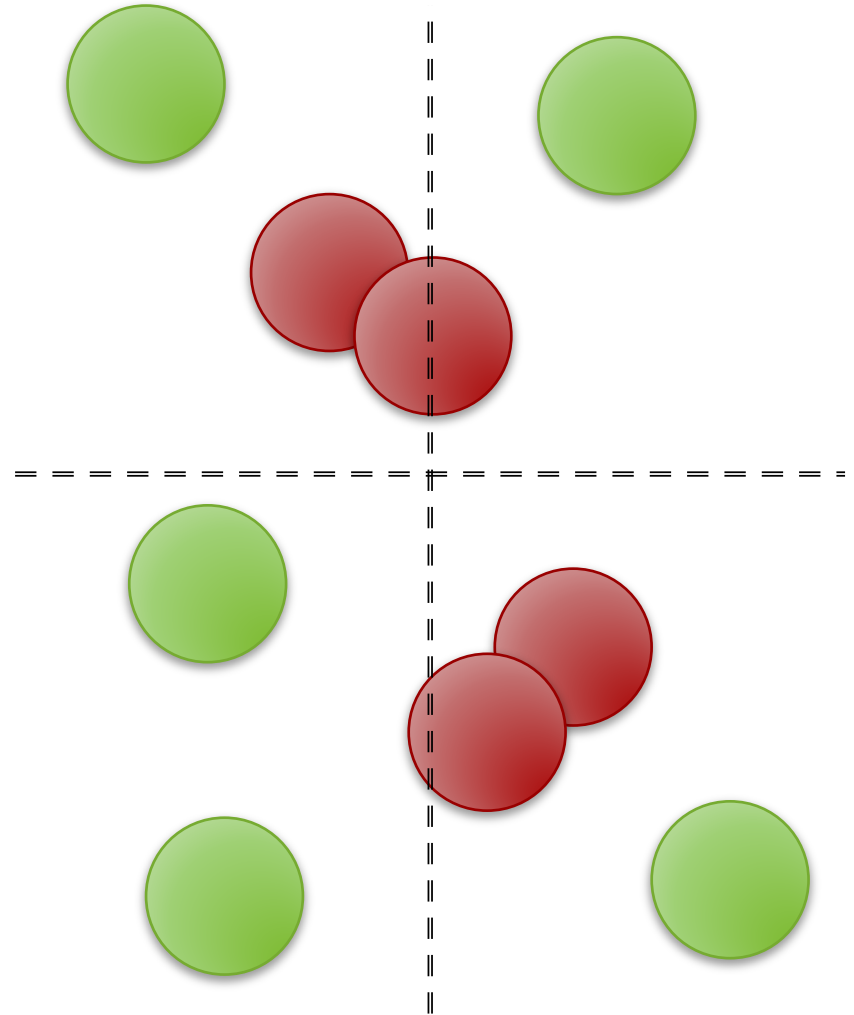
for each cell location:

for each object x:

for each object y:

if $x \neq y$ and x, y collide:

resolve collision



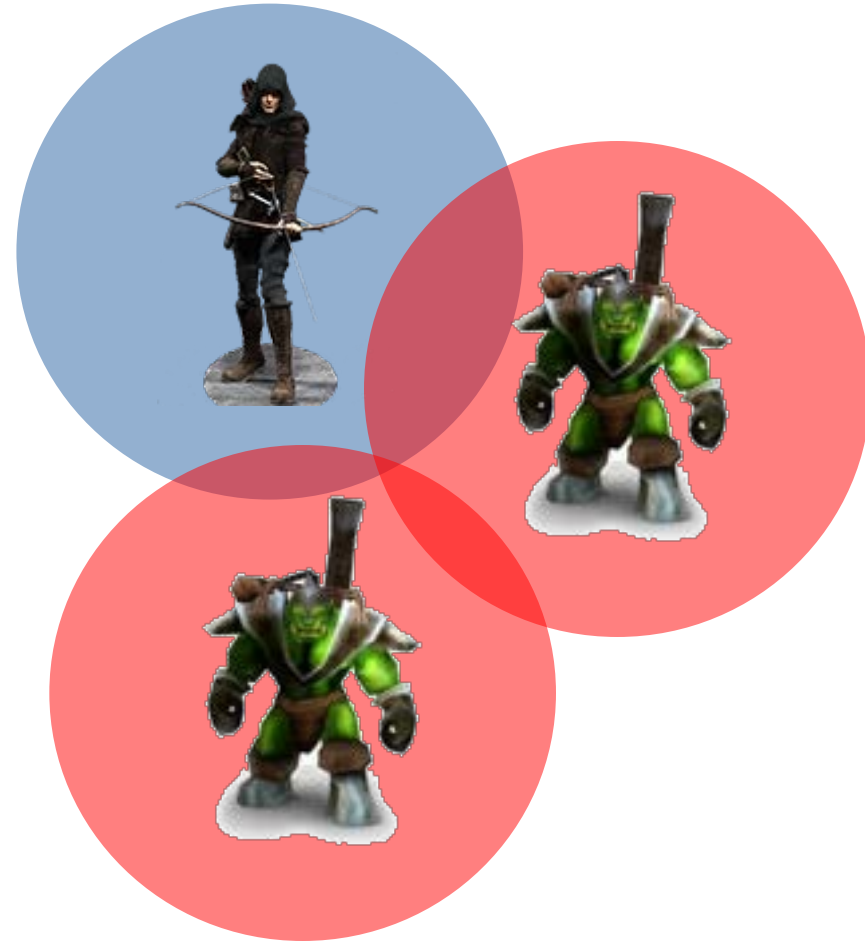
Similar Ideas Exist in AI

- **Area of Interest**
 - Limit the sensing range
 - Only “see” what in range
 - Used in targeting, stealth
- Works in both directions
 - **Nimbus**: “can see” radius
 - **Aura**: “can be seen” radius
- Can use cell optimization



Similar Ideas Exist in AI

- **Area of Interest**
 - Limit the sensing range
 - Only “see” what in range
 - Used in targeting, stealth
- Works in both directions
 - **Nimbus**: “can see” radius
 - **Aura**: “can be seen” radius
- Can use cell optimization



THIEF

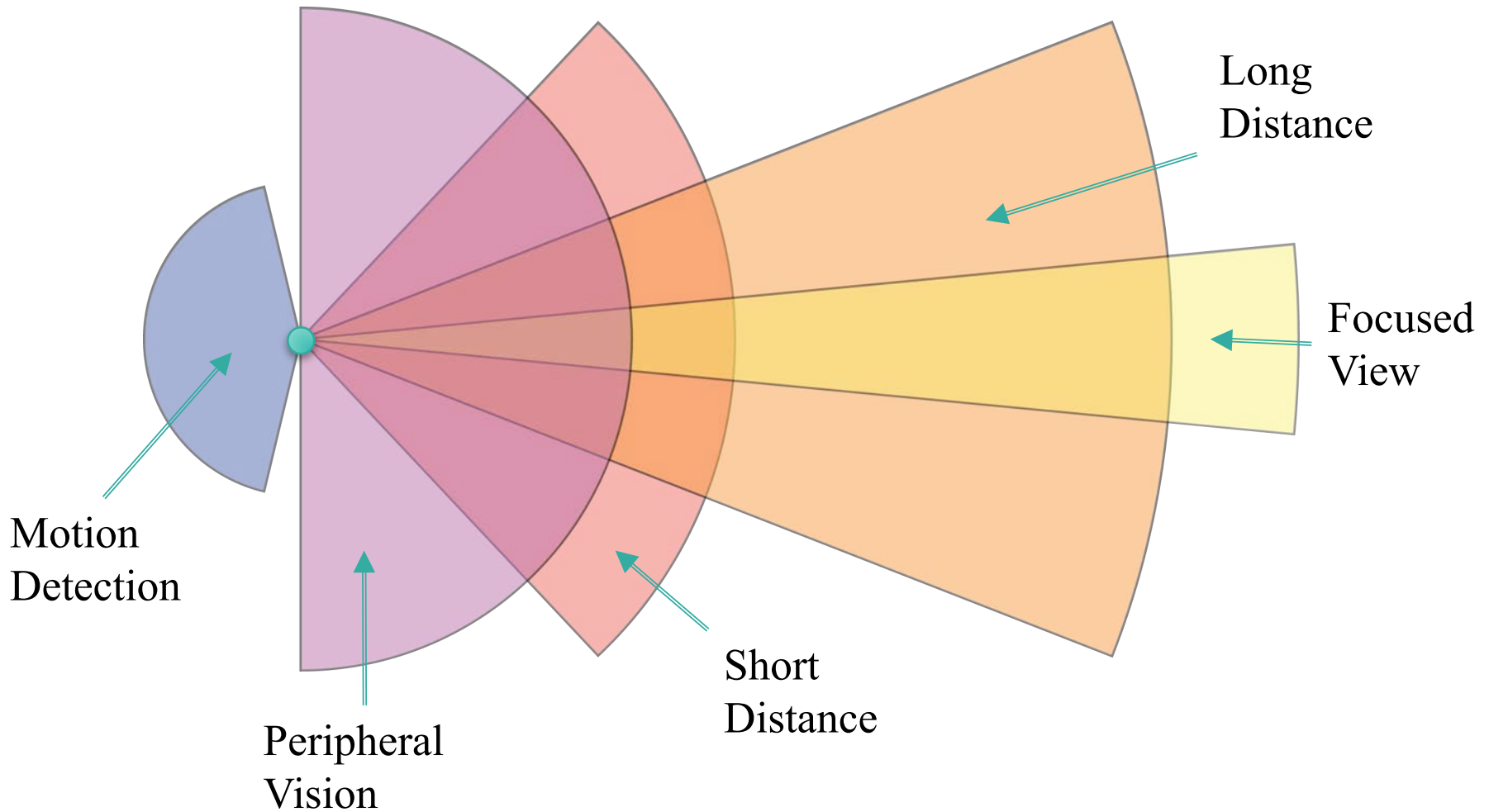
DEADLY SHADOWS

Loading...



Stealth tip: Use WALK to move slowly and very quietly. Use CREEP to move even more slowly and be completely silent.

Area of Interest Management *Thief*



Problem with this Idea

Cell-Based AI

for each entity x:

put x into cell slot

for each cell location:

for each entity x:

for each entity y:

if x can see y:

add y to sense of x

Sense &
Think

Act

incompatible

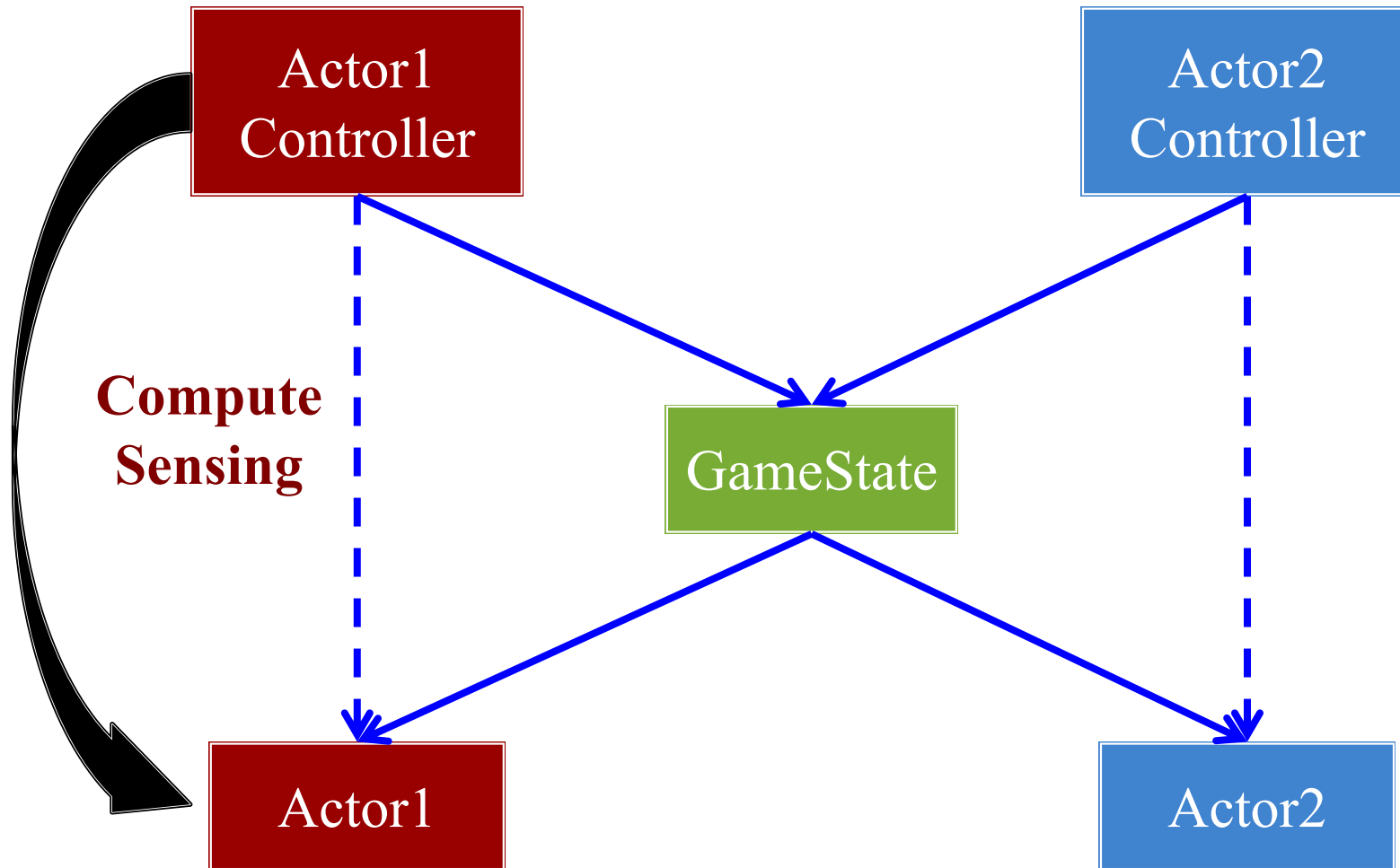
NPC 1

NPC 2

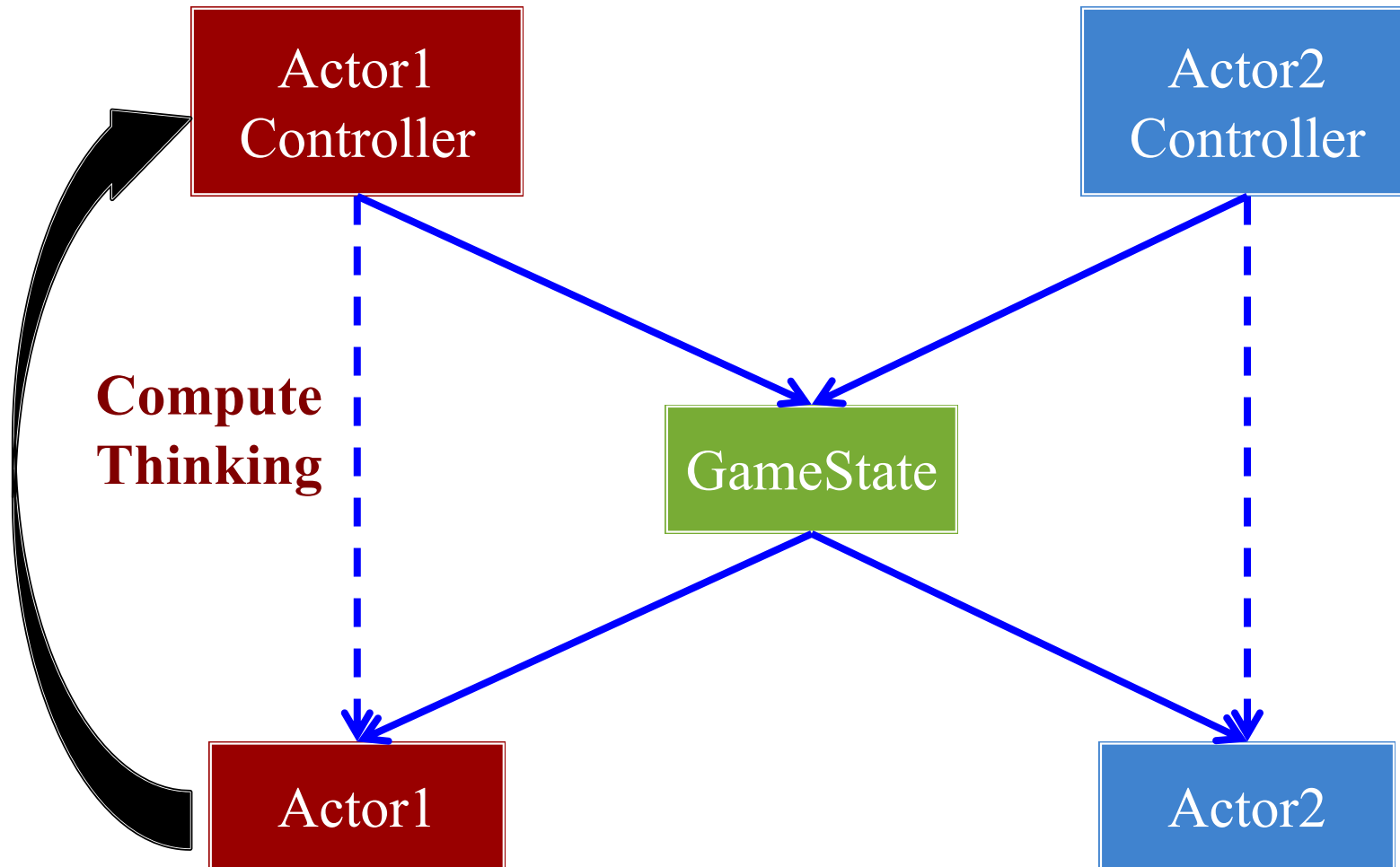
NPC 1

NPC 2

Recall: Reducing Dependencies



Recall: Reducing Dependencies

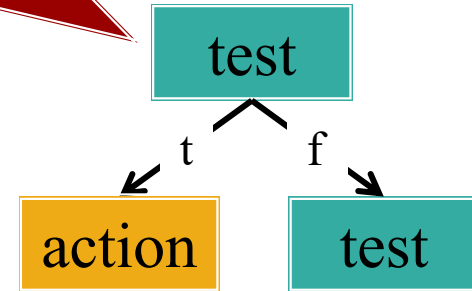
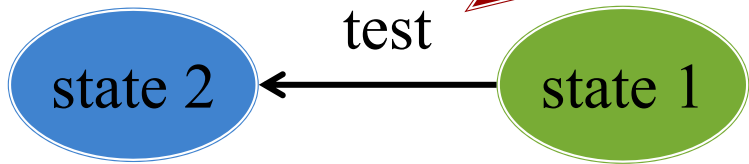


Solution: Event Driven AI

Finite State Machines

Decision Trees

Can support arbitrary (boolean) functions here

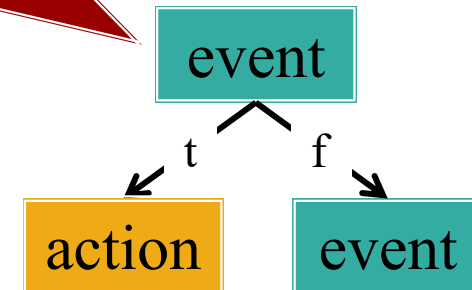
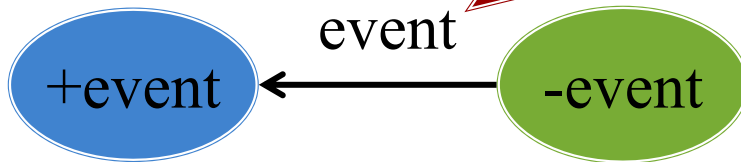


Solution: Event Driven AI

Finite State Machines

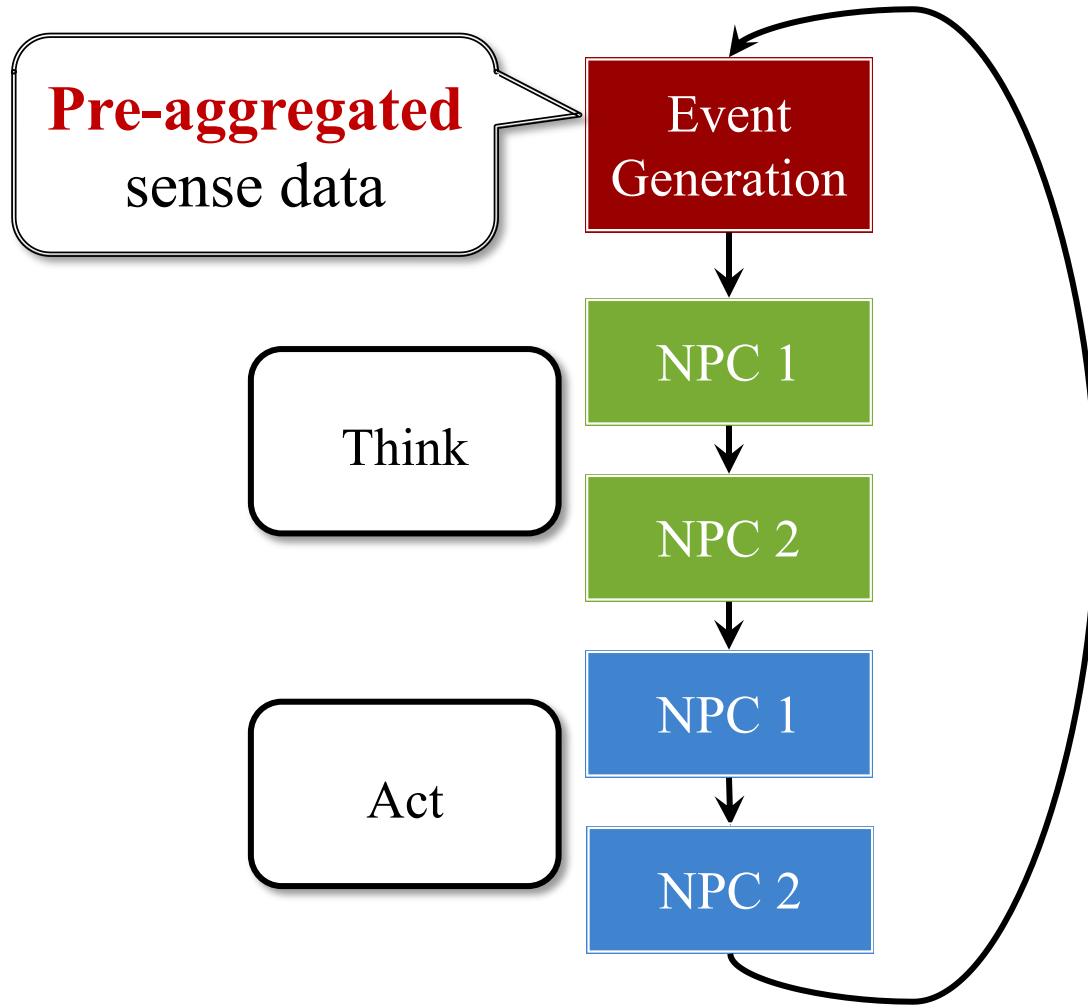
Decision Trees

But we only want simple tests!



Event: *Precomputed* result before AI thinking starts

The True AI Loop



Event: Encoded Sense Data

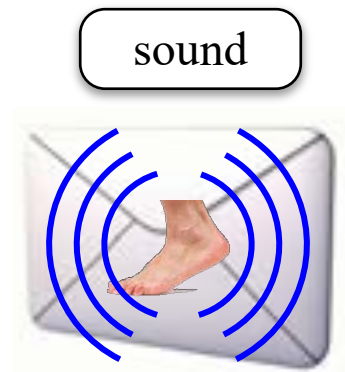
- **Sight Event**

- Type of entity seen
- *Location* of entity seen



- **Sound Event**

- Type of sound heard
- *Direction* of sound heard

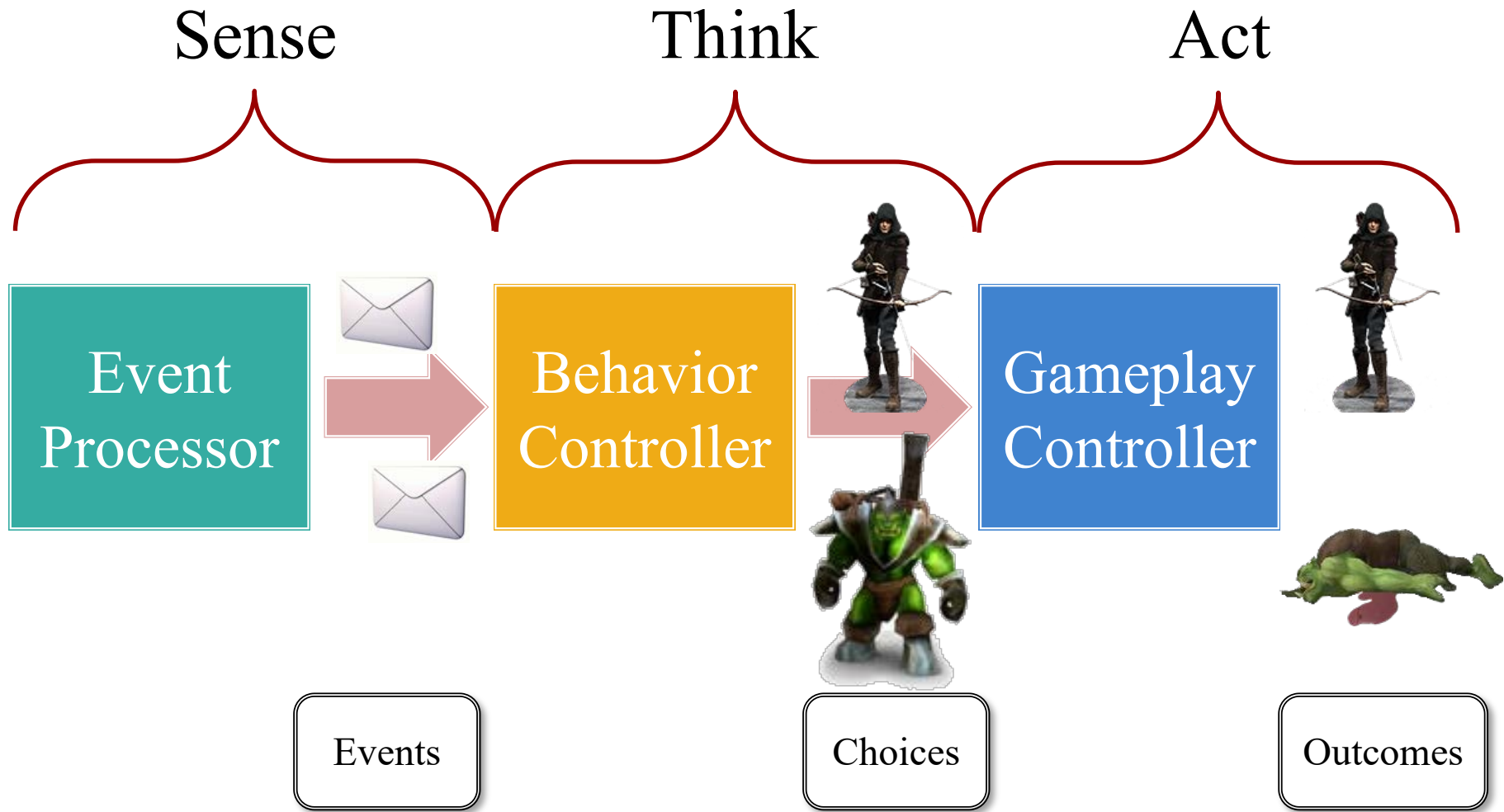


- **Smell Event**

- Type of smell perceived
- *Proximity* of the smell

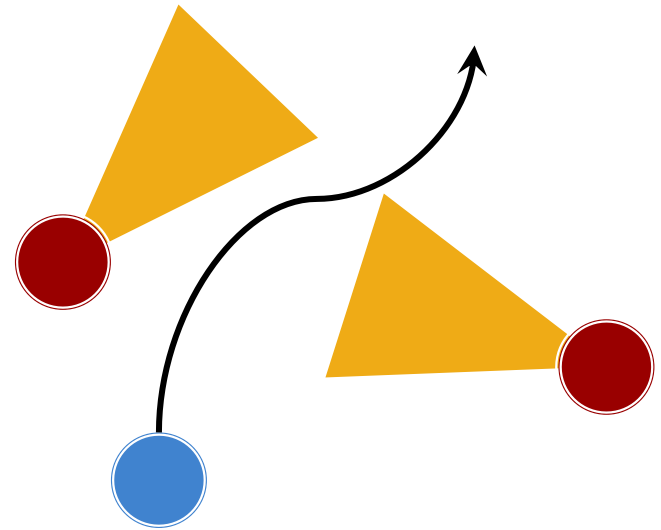


Sense-Think-Act Revisited

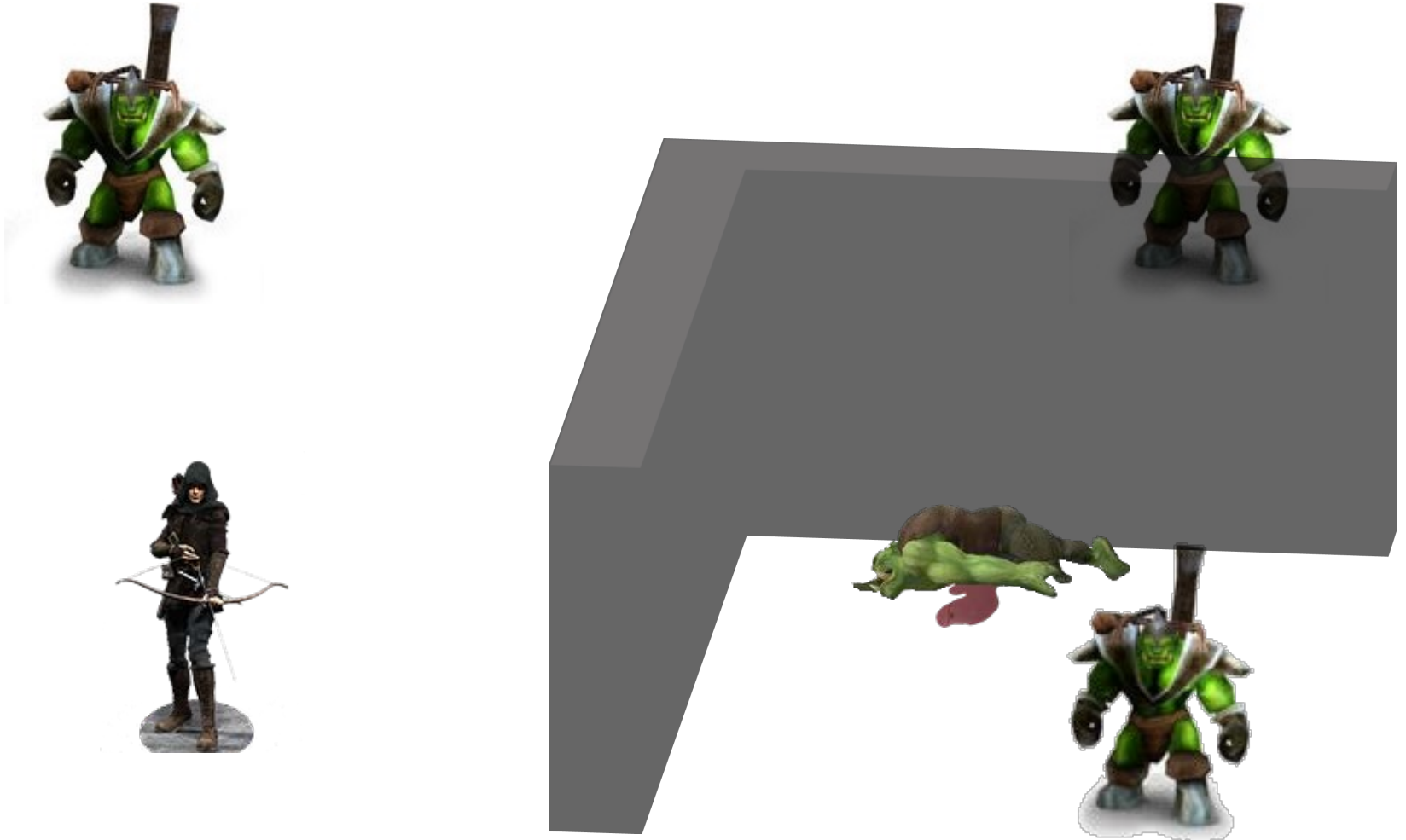


Example: Line-of-Sight

- Use **Box2D** for sensing
 - Method rayCast in World
 - Provide a RayCastCallback
- Think about the **callback**
 - Happens *after* physics done
 - Often later than AI phase
- It should **generate an event**
 - Can be processed next phase
 - Keeps order of code clean

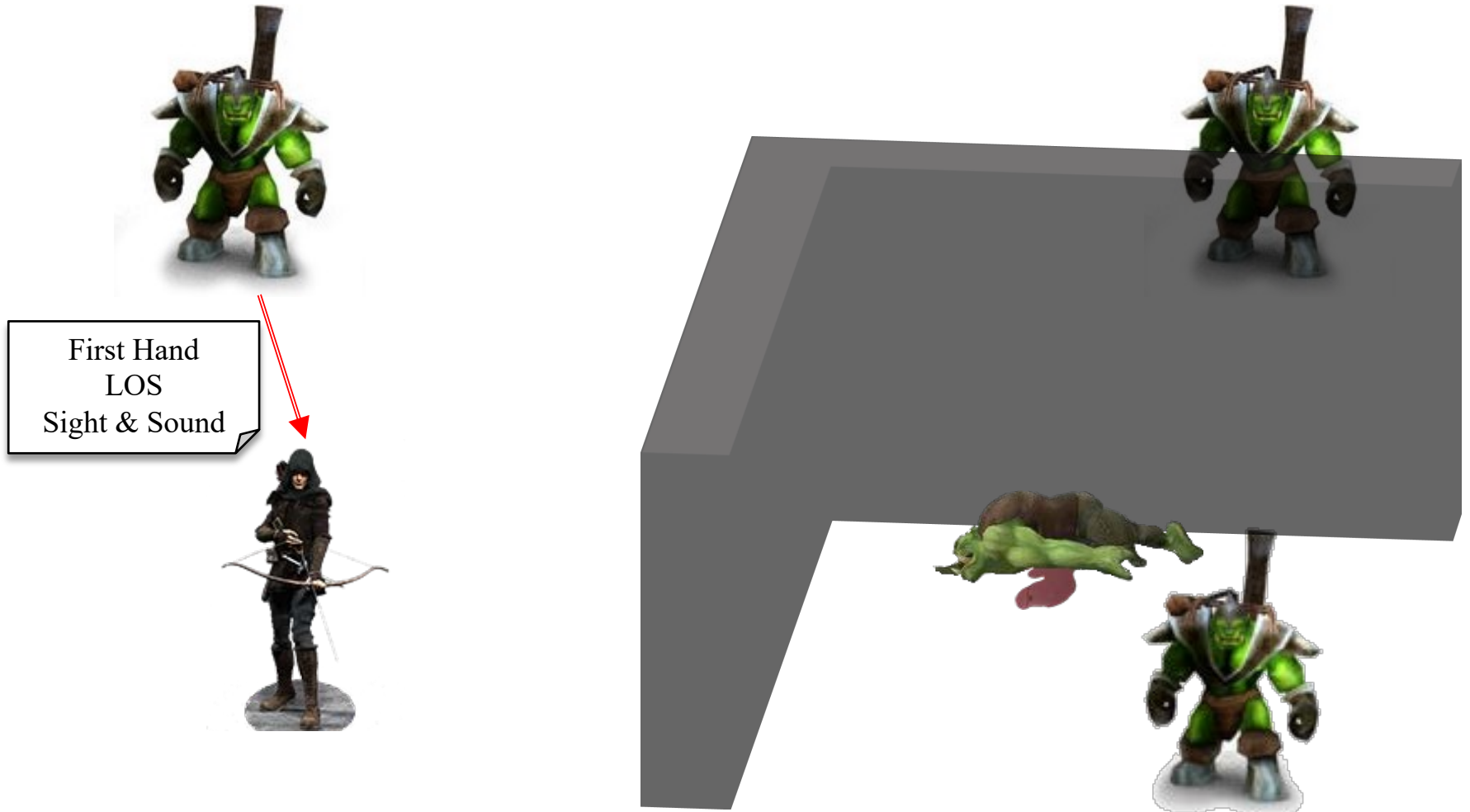


Communicating Sense Events

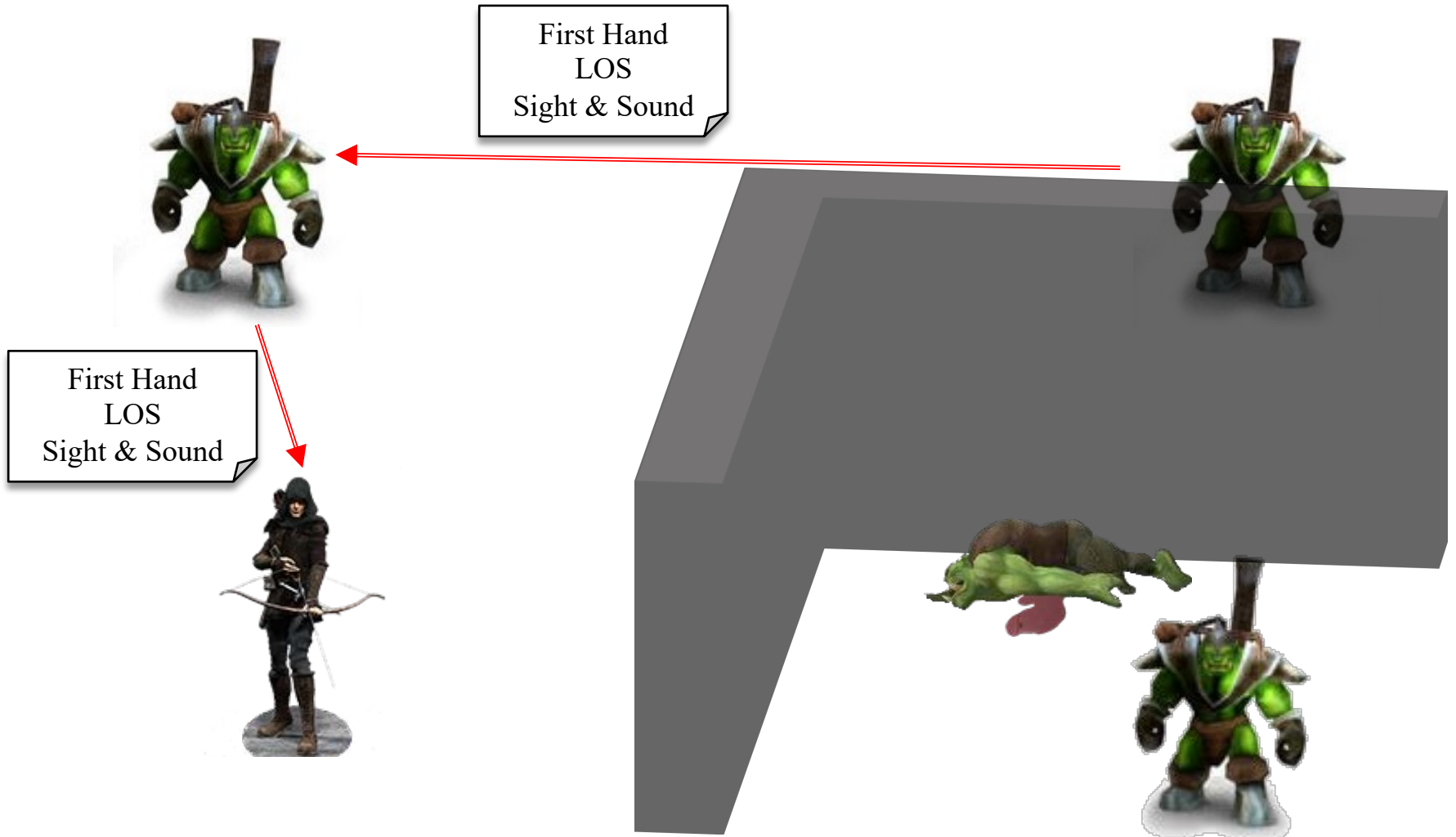


Sensing & Perception

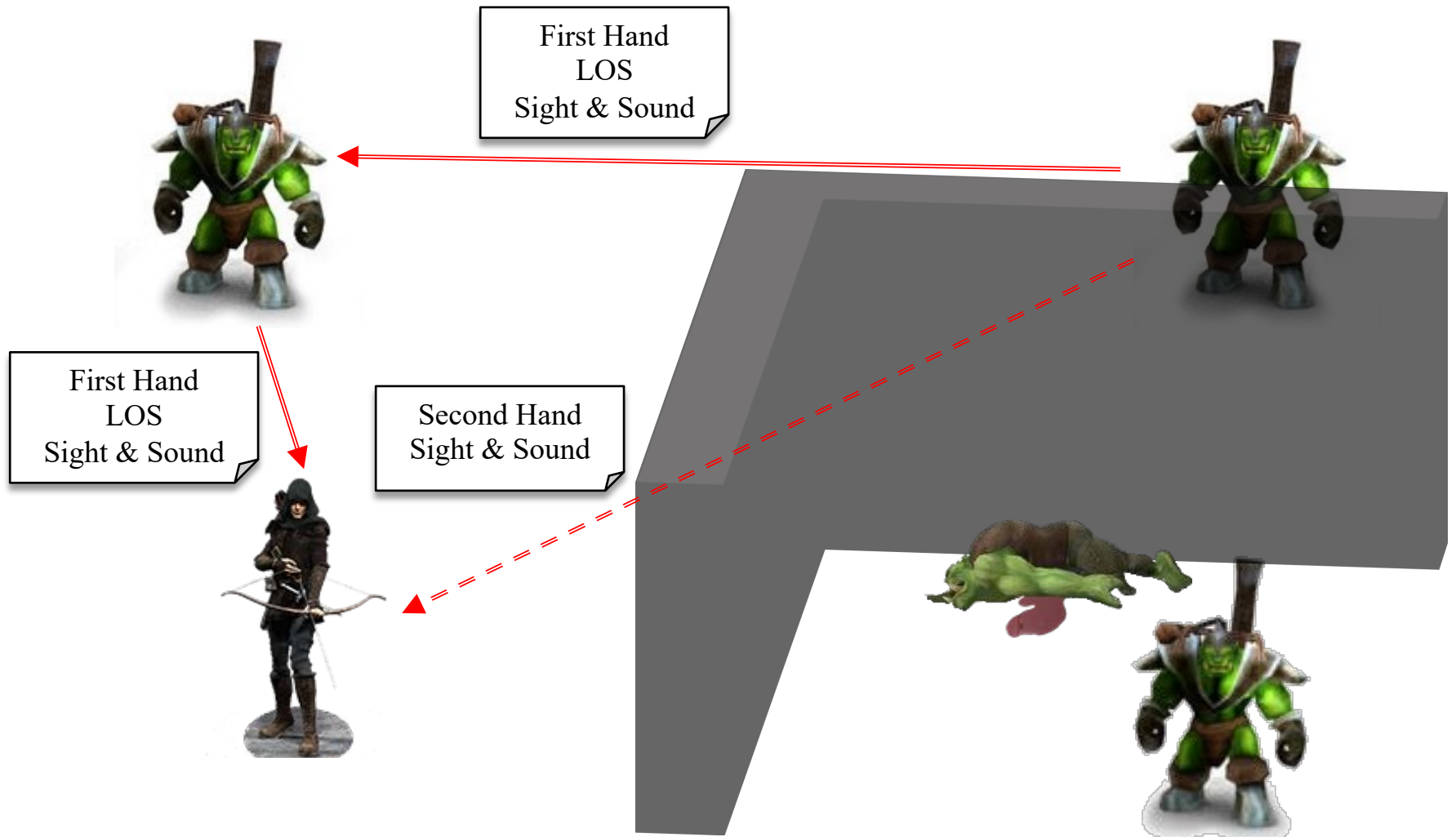
Communicating Sense Events



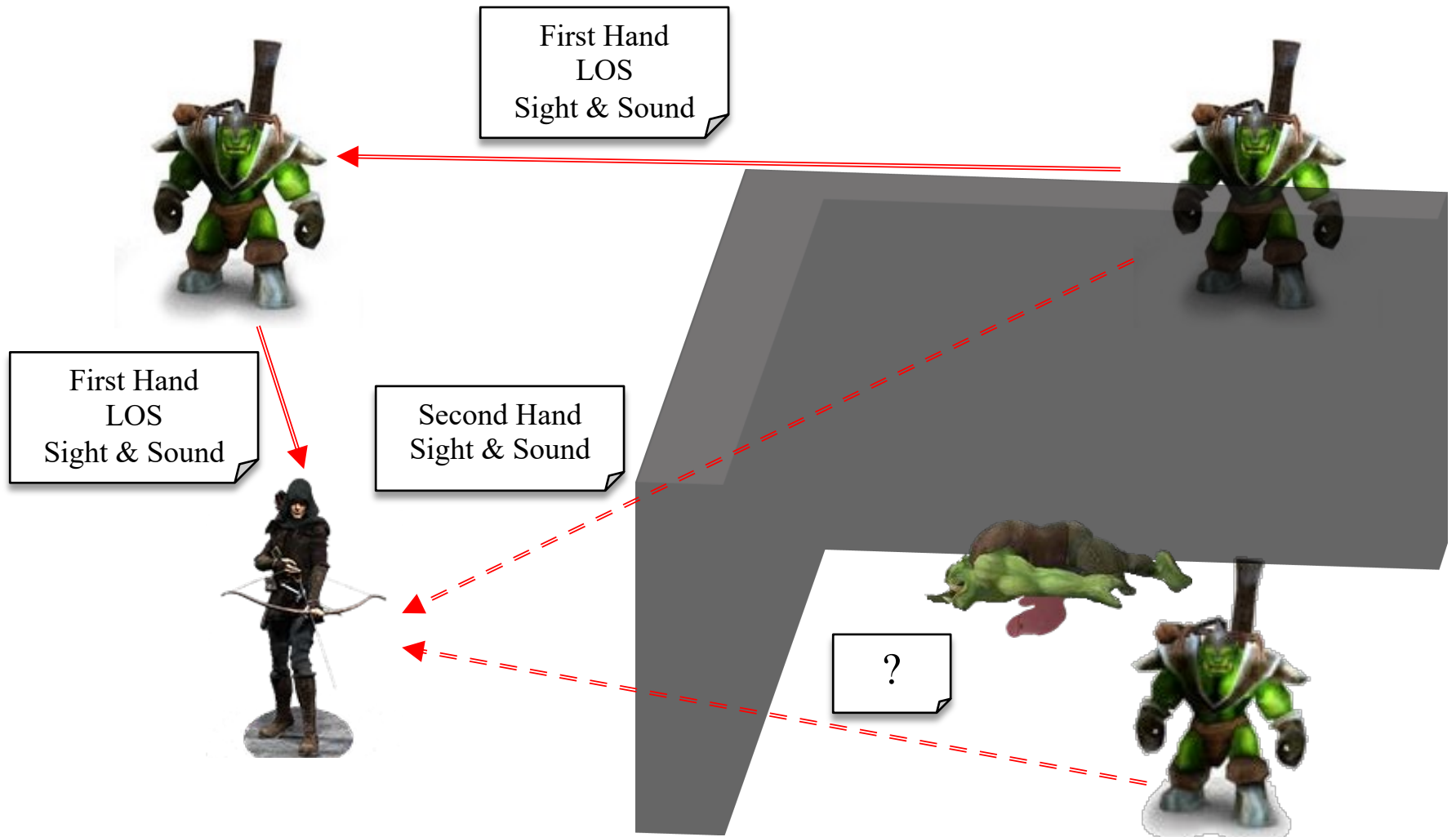
Communicating Sense Events



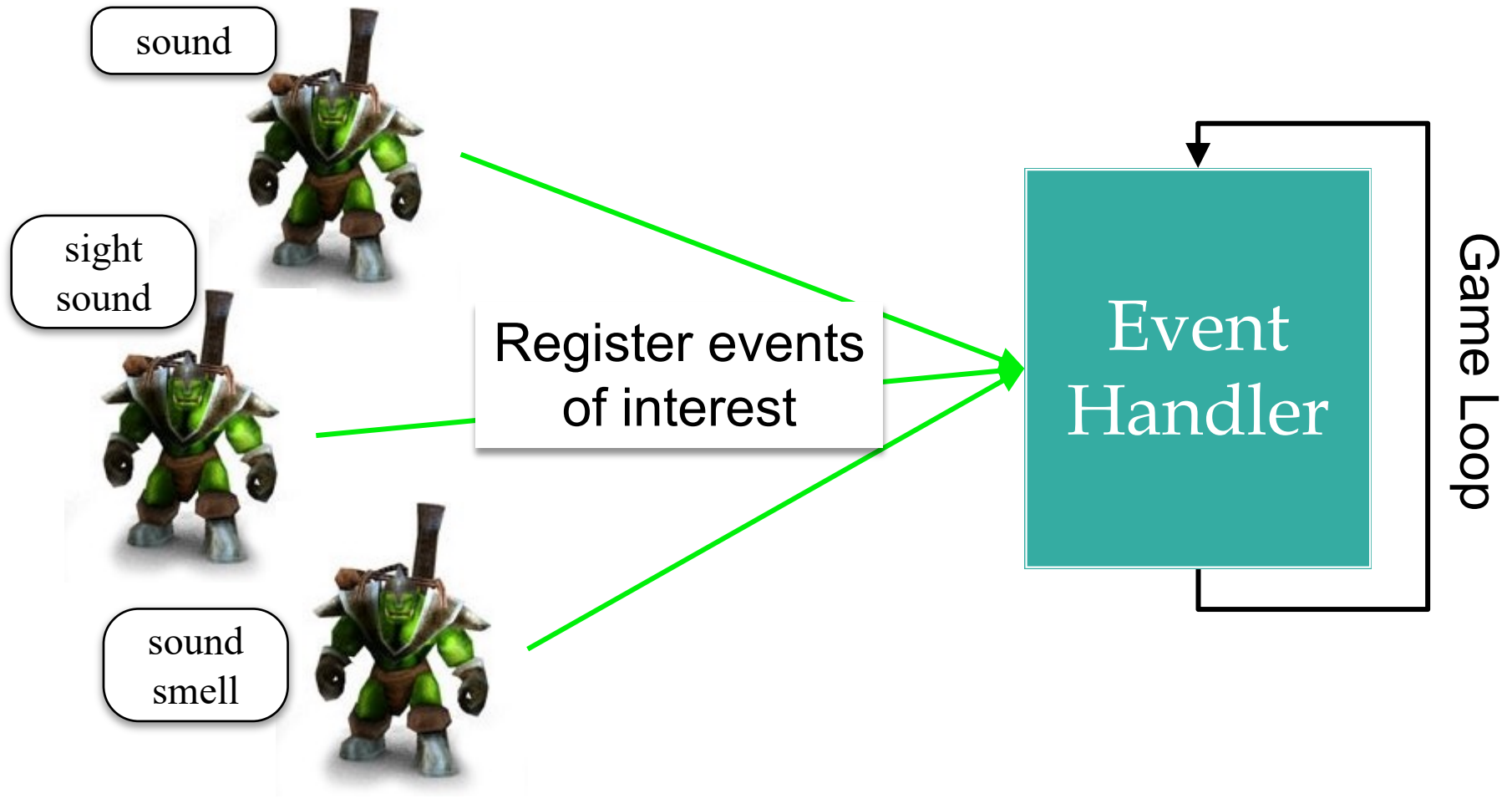
Communicating Sense Events



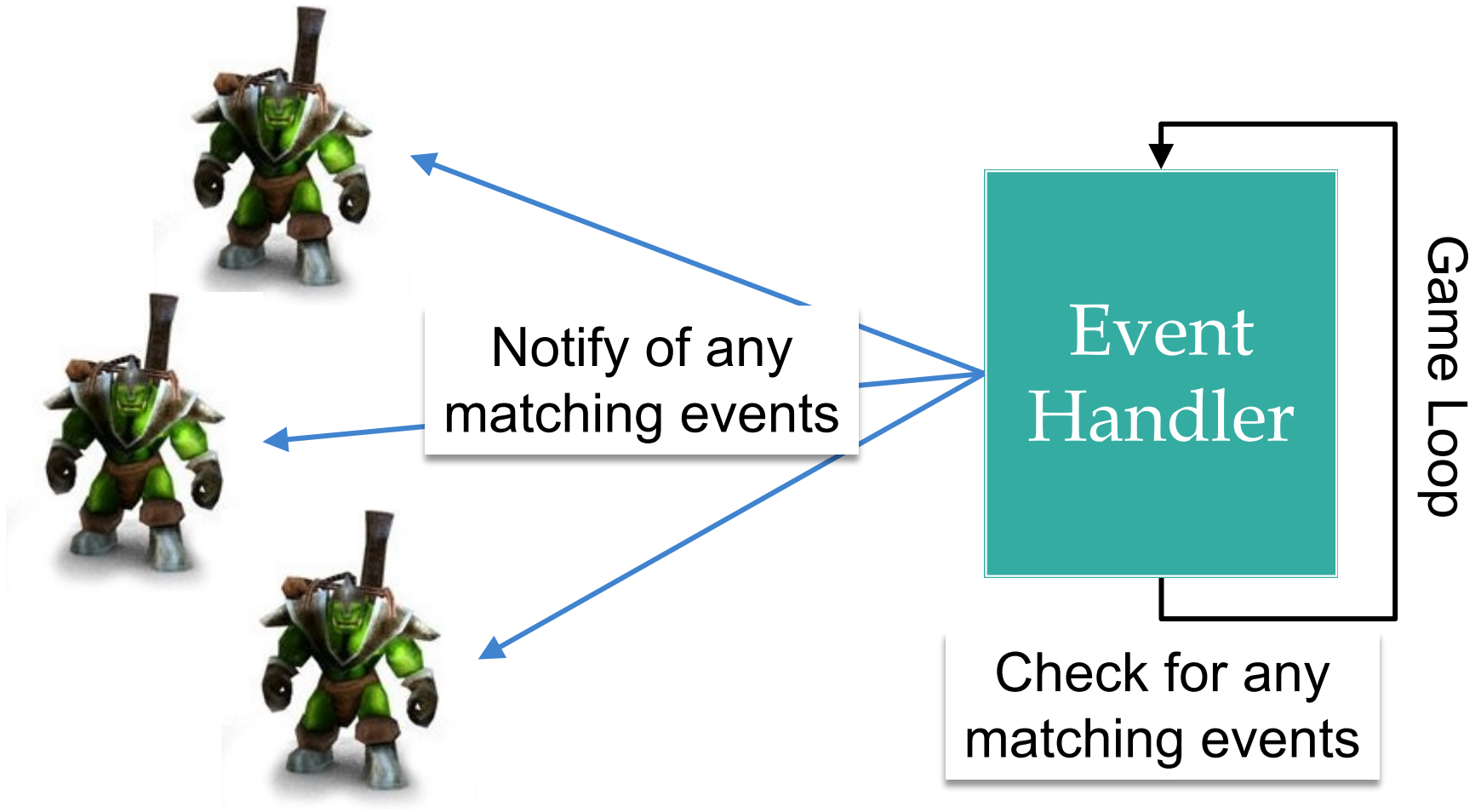
Communicating Sense Events



Sense Event Matching



Sense Event Matching



Event Communication in LibGDX

MessageDispatcher

- Send with `dispatchMessage`
 - `delay` (0 if immediate)
 - `sender` (can be null)
 - `target` (null for subscribers)
 - `type` (user defined int code)
 - `data` (object, like Box2D)
- Subscribe with `addListener`
 - NPC to receive message
 - Type (int) to subscribe to

Telegram

- Stores the event message
 - Entries of `dispatchMessage`
 - Except for the `delay` value
 - Preaggregated sense in `data`
- Received by `Telegraph`
 - Interface for the receiver
 - Implemented by the NPC
 - One method: `handleMessage`

Event Communication in LibGDX

MessageDispatcher

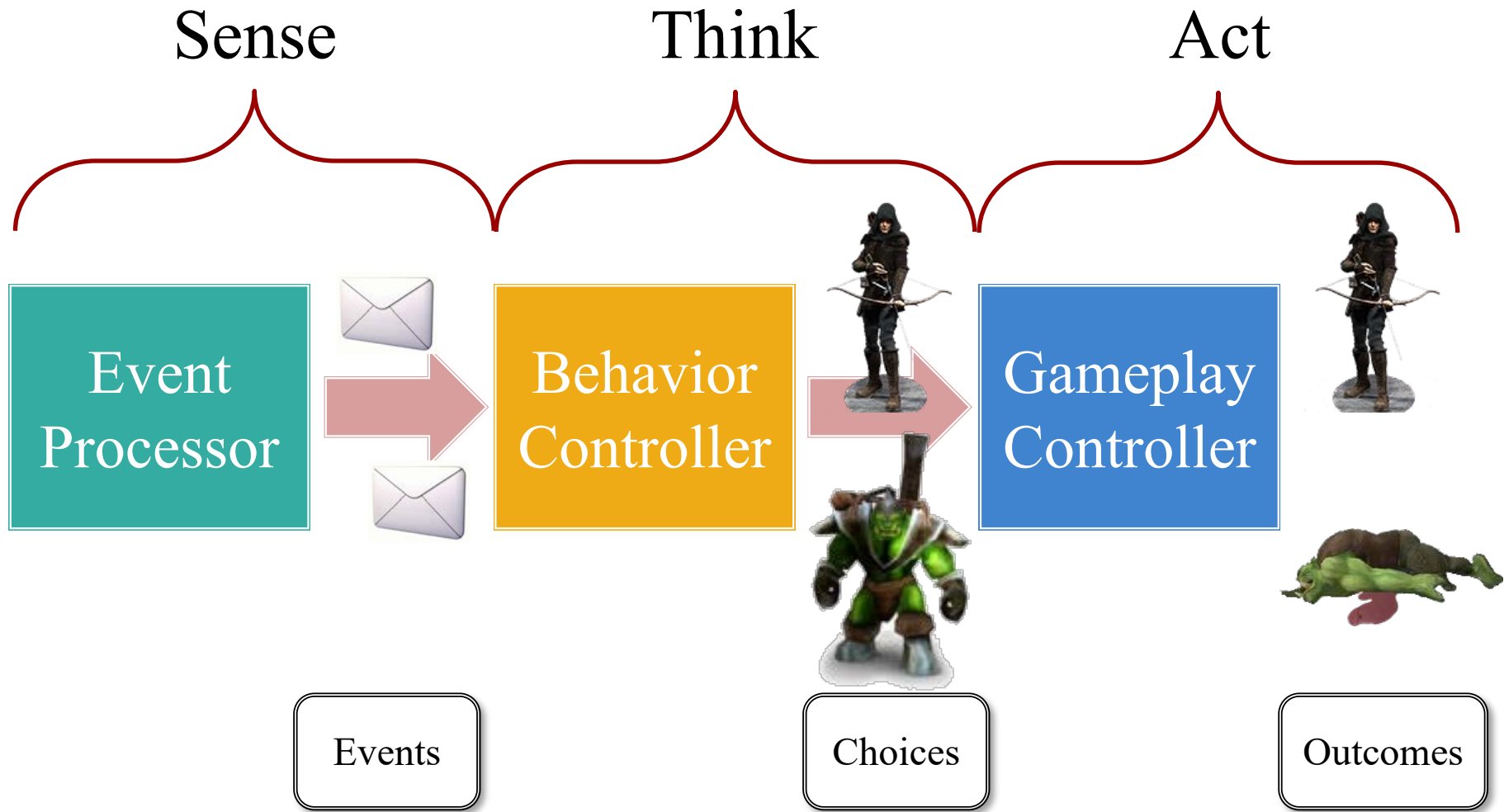
- Send with `dispatchMessage`
 - `delay` (0 if immediate)
 - `sender` (can be null)
 - `target` (null for subscribers)
 - `type` (user defined int code)
 - `data` (object, like Box2D)
- Subscribe with `addListener`
 - NPC to receive message
 - Type (int) to subscribe to



Telegram

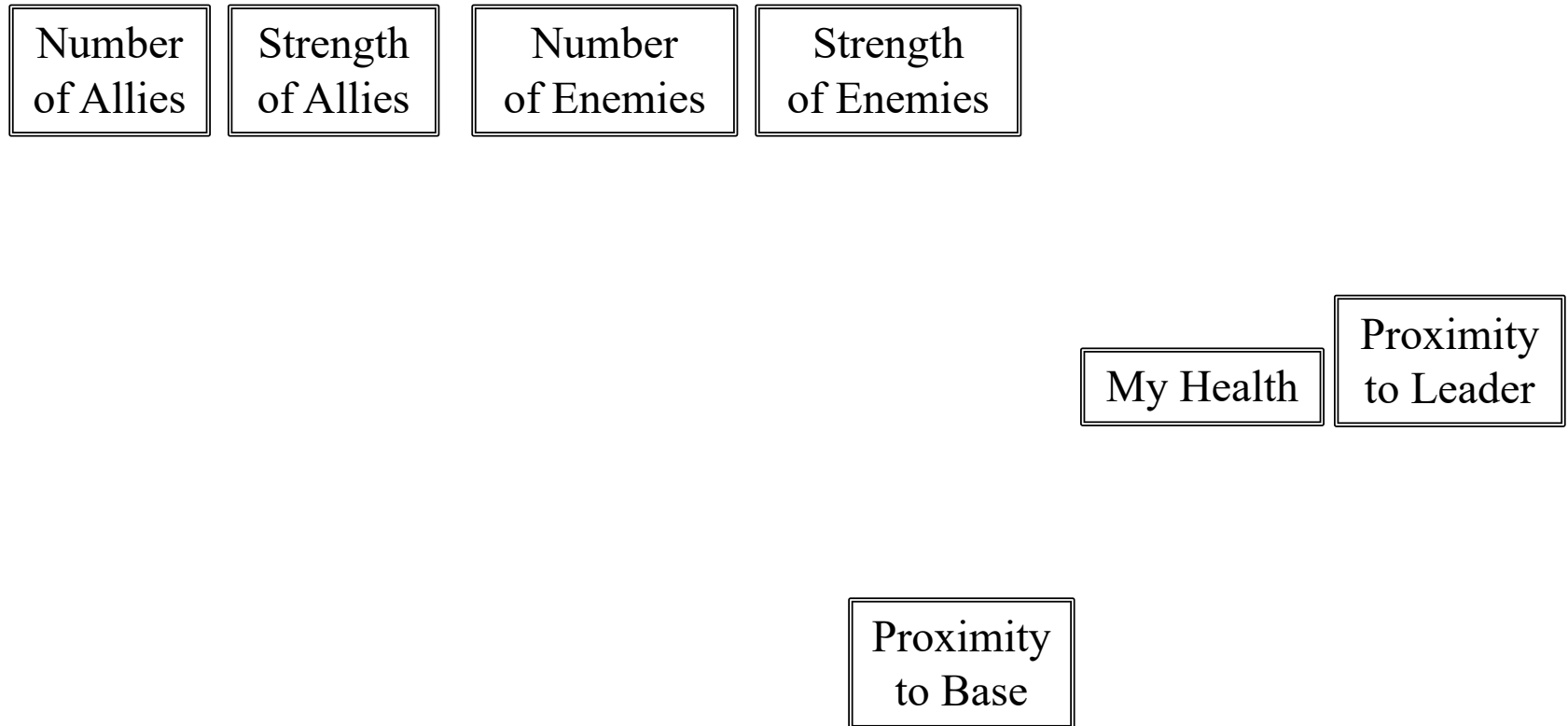
- Stores the event message
 - Entries of `dispatchMessage`
 - Except for the `delay` value
 - Preaggregated sense in `data`
- Received by `Telegraph`
 - Interface for the receiver
 - Implemented by the NPC
 - One method: `handleMessage`

Separation Allows Many Optimizations



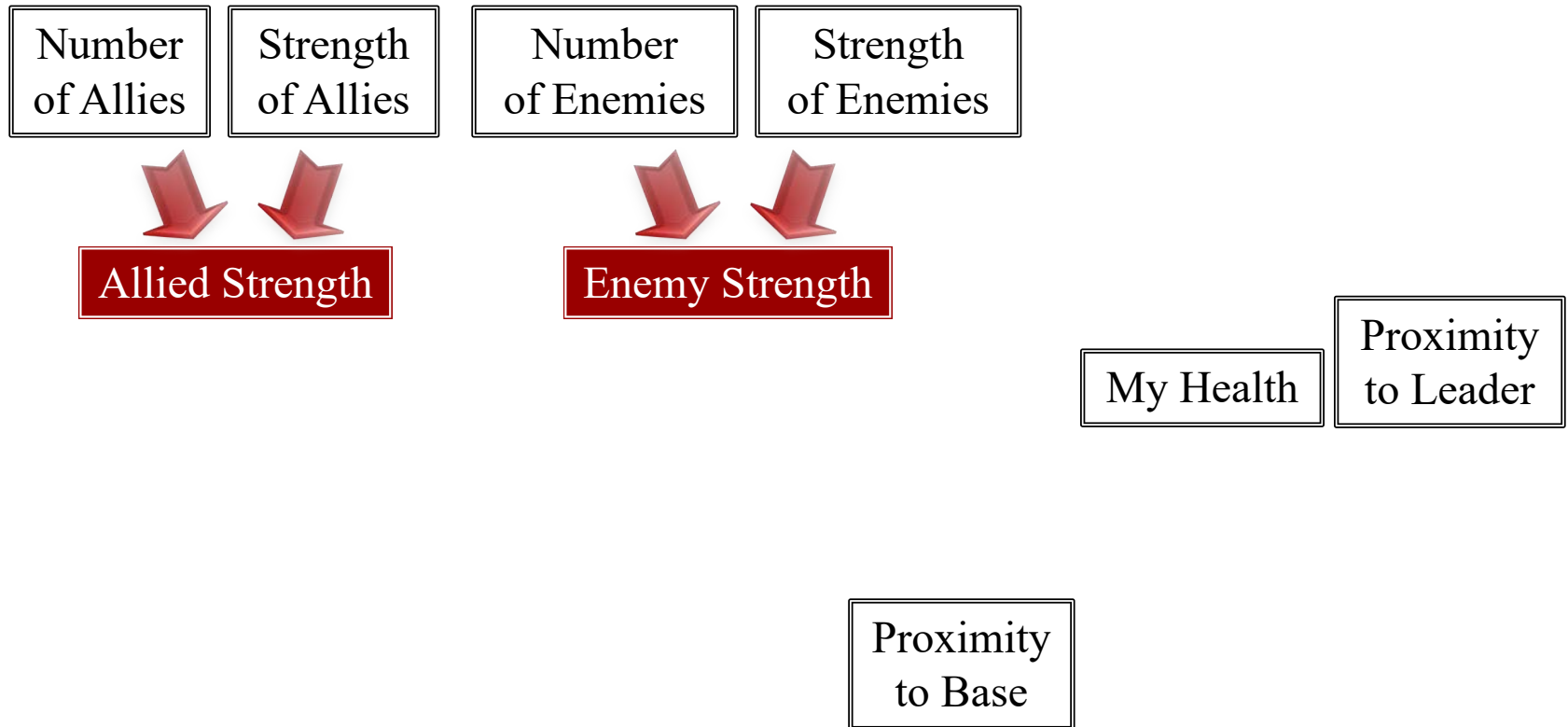
Compression: Aggregation Trees

Slide courtesy of Dave Mark



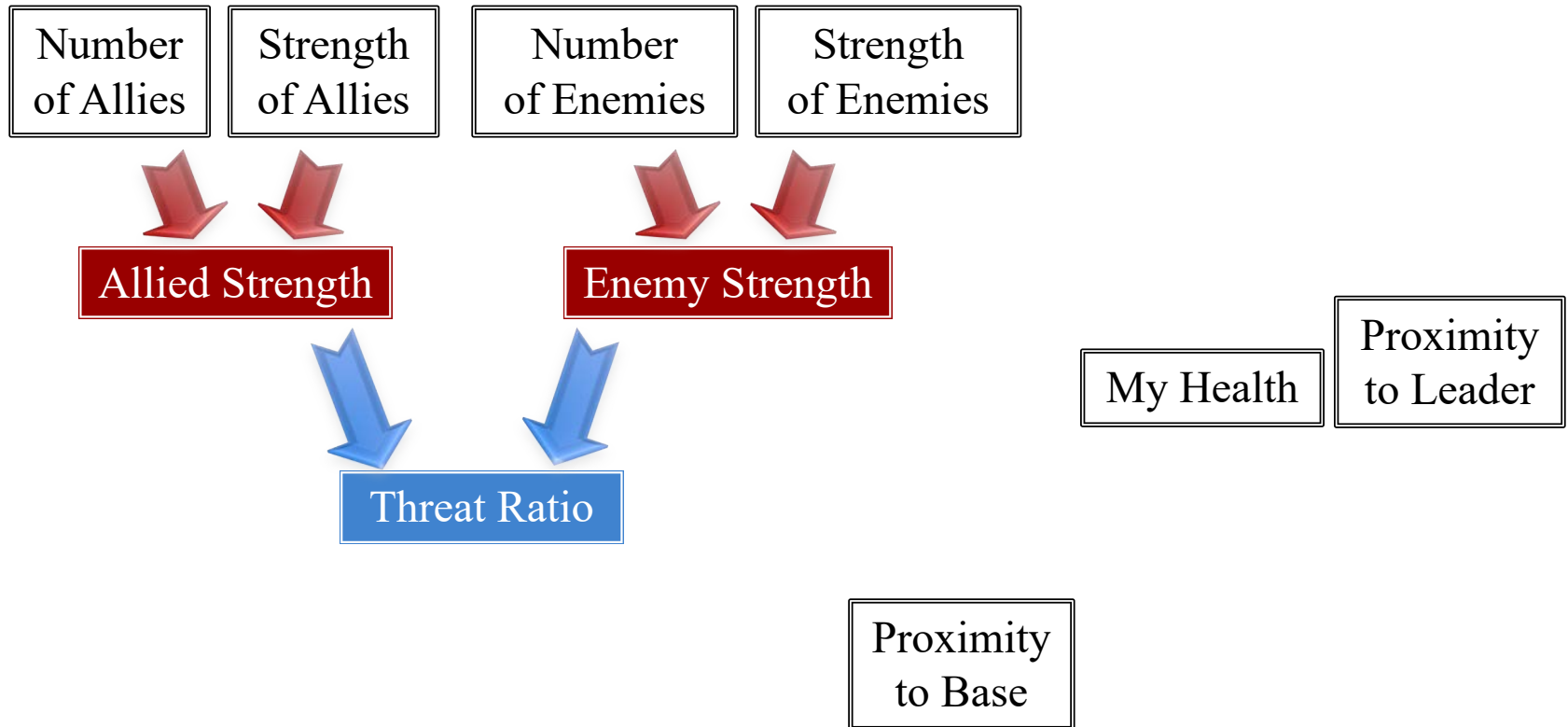
Compression: Aggregation Trees

Slide courtesy of Dave Mark



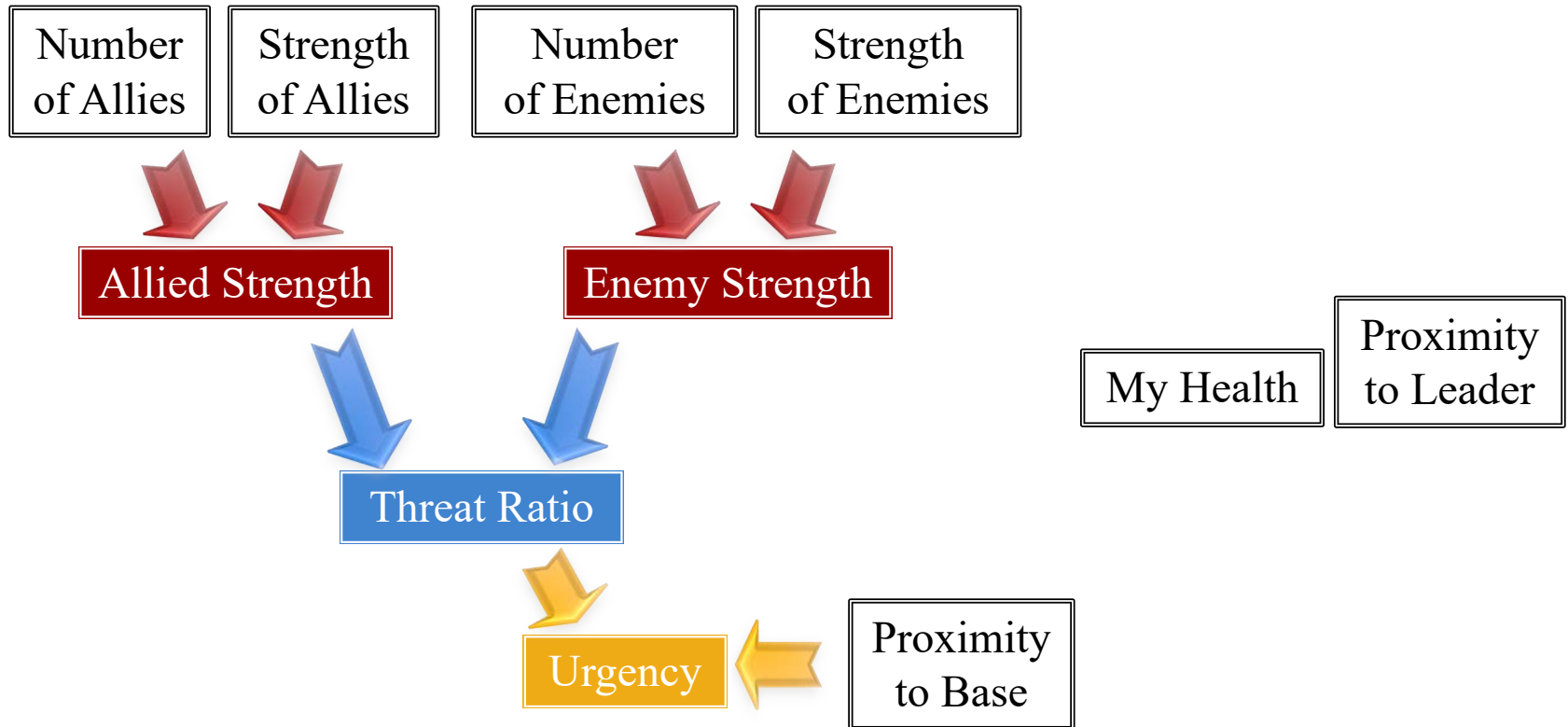
Compression: Aggregation Trees

Slide courtesy of Dave Mark



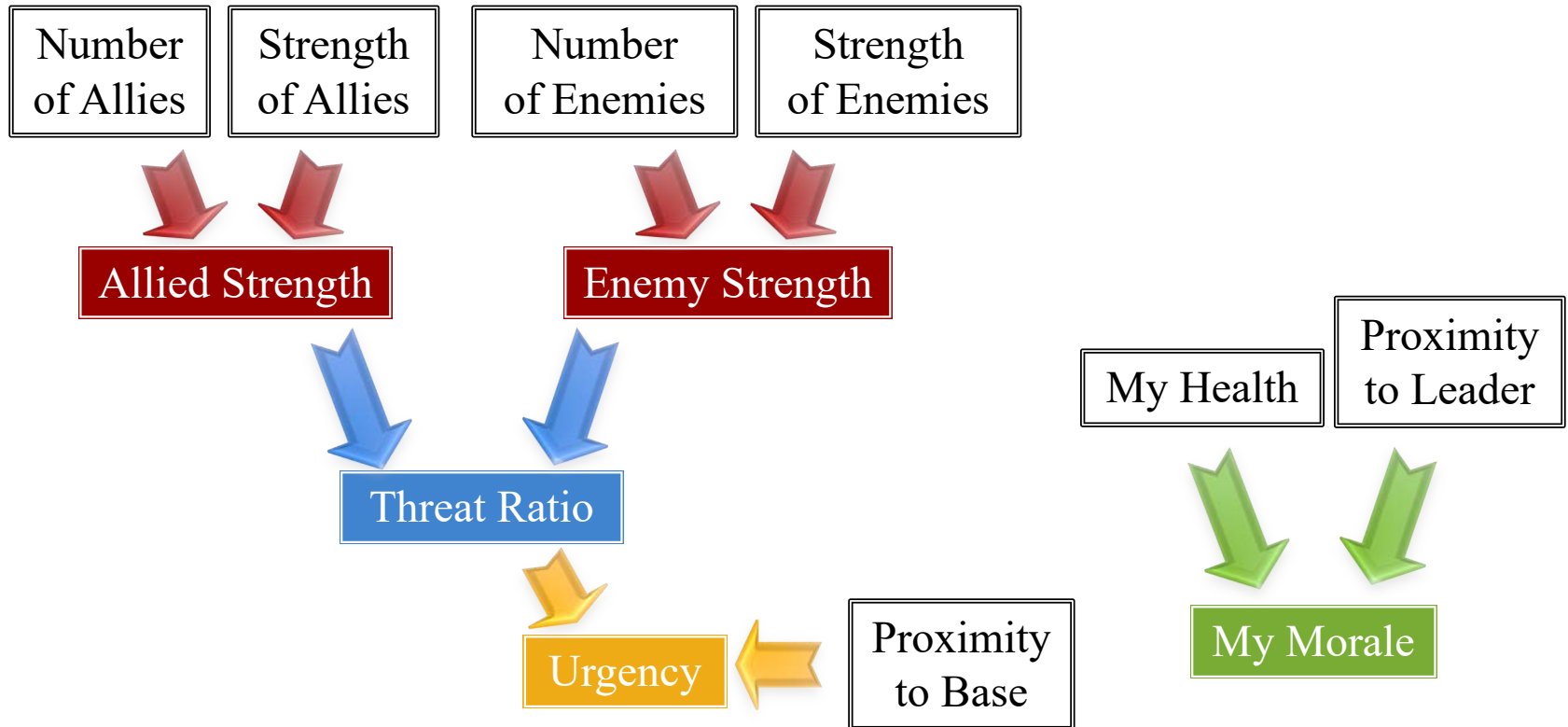
Compression: Aggregation Trees

Slide courtesy of Dave Mark



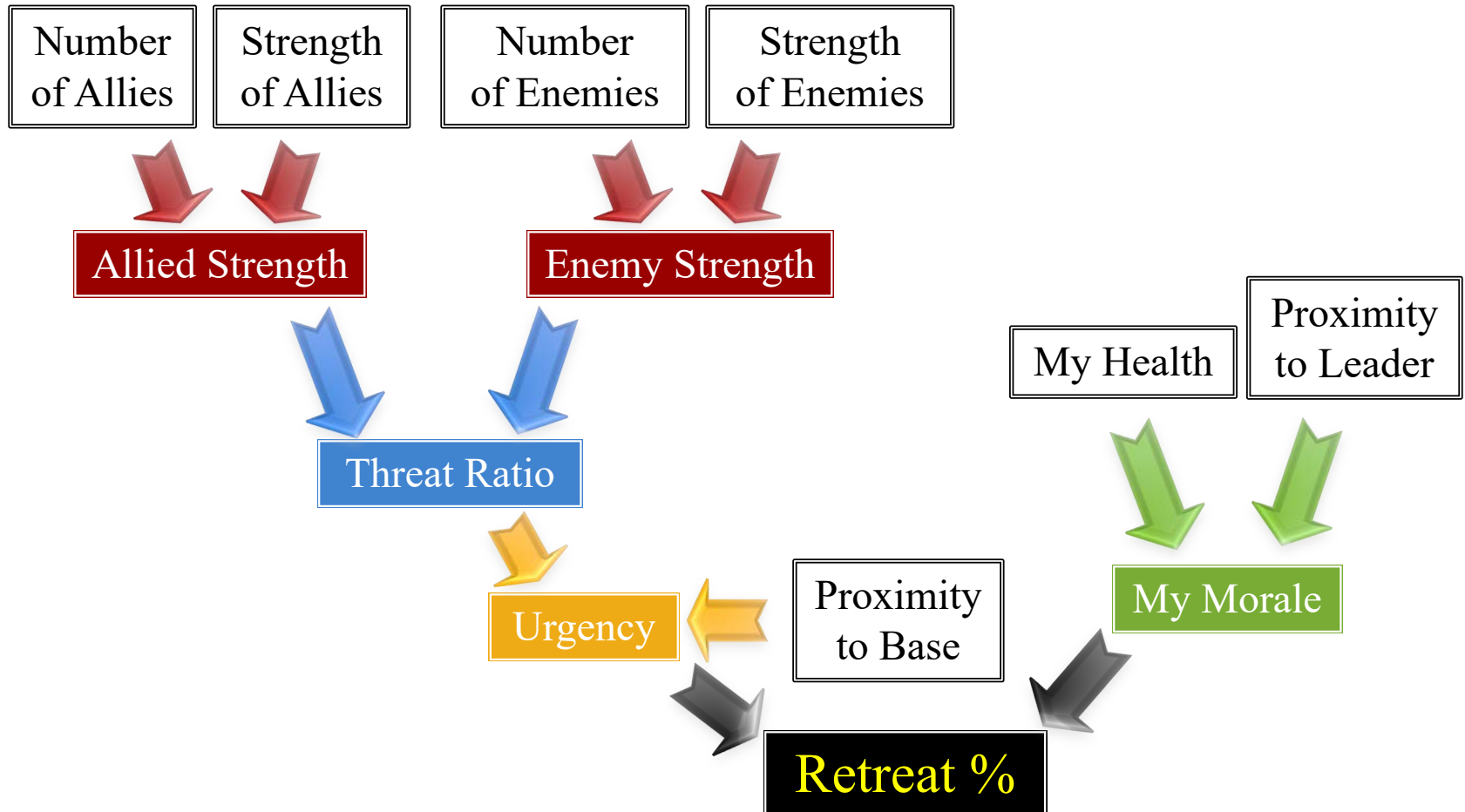
Compression: Aggregation Trees

Slide courtesy of Dave Mark



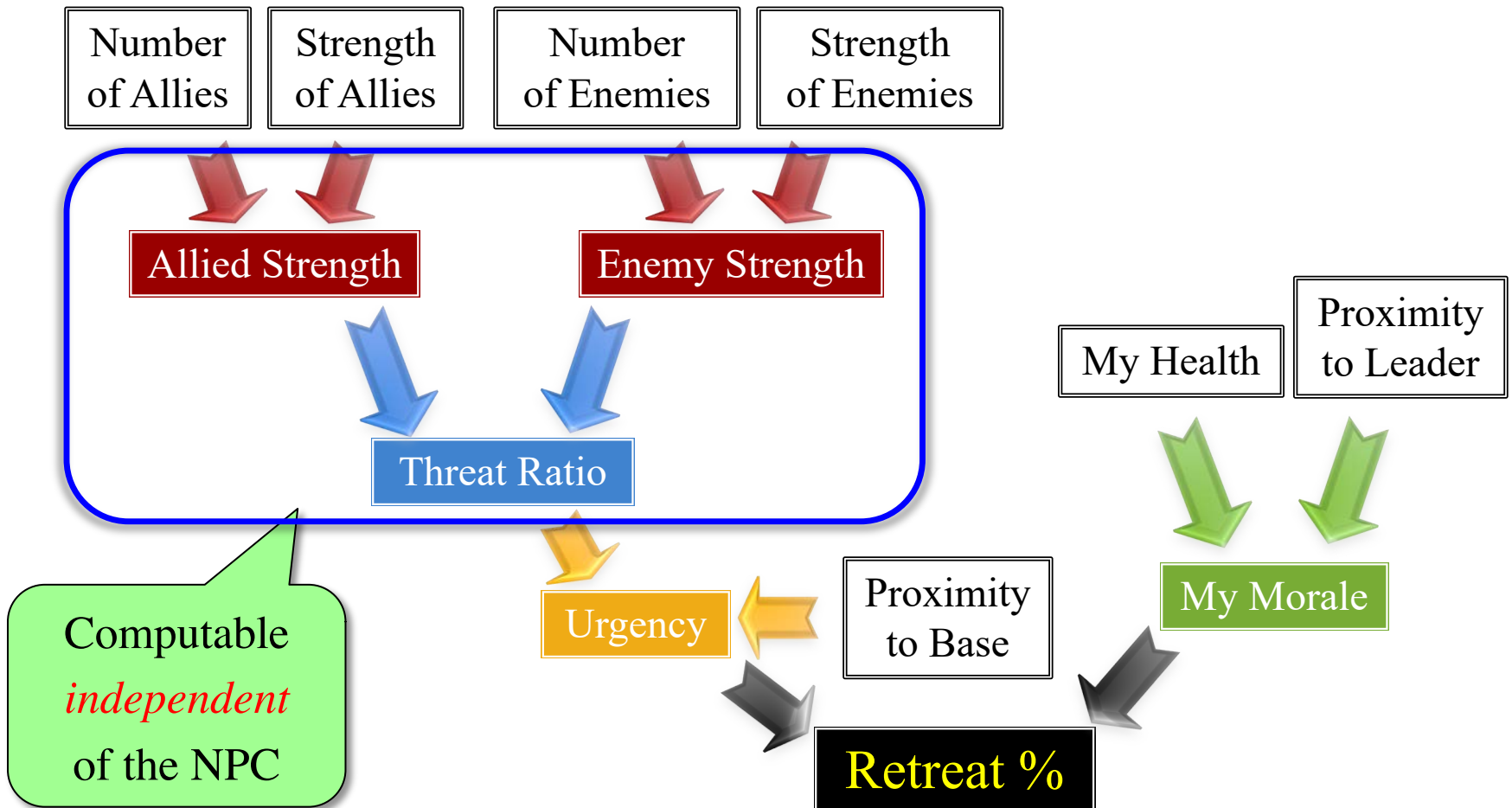
Compression: Aggregation Trees

Slide courtesy of Dave Mark



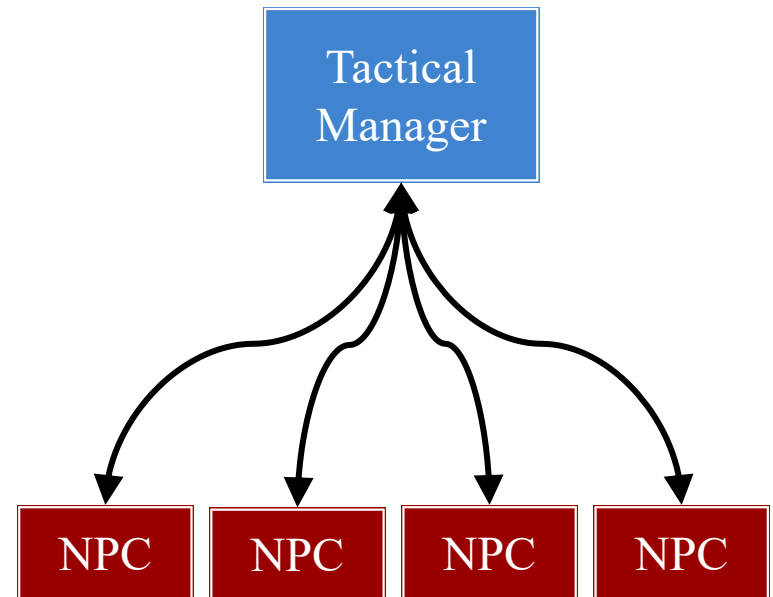
Compression: Aggregation Trees

Slide courtesy of Dave Mark

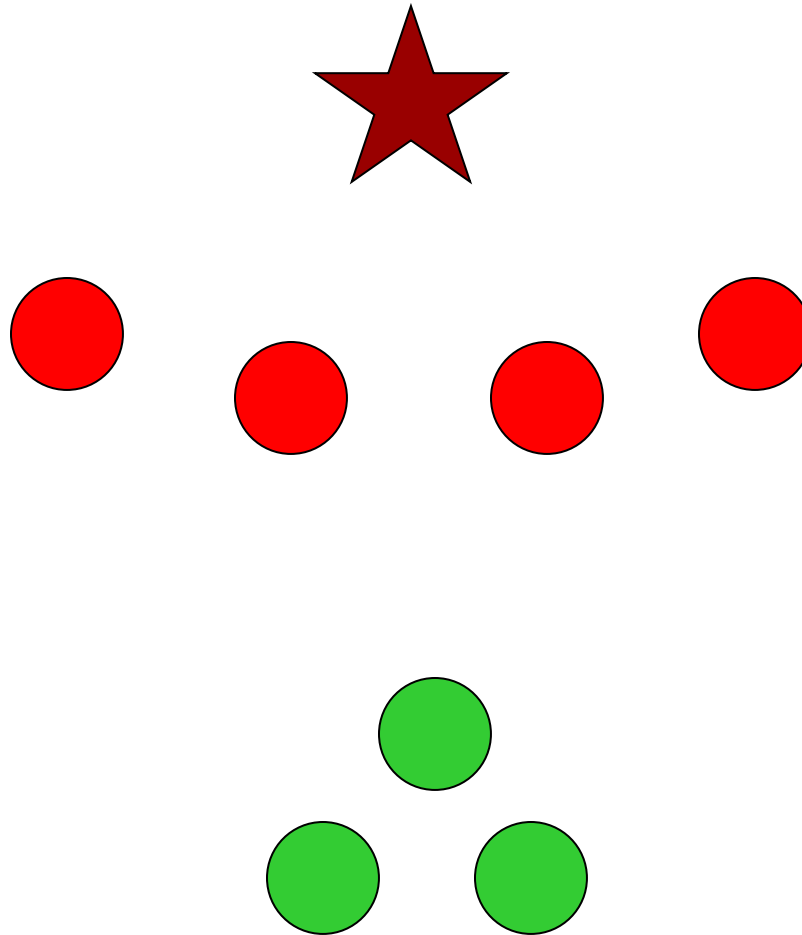


Delegation: Tactical Managers

- “Invisible NPC”
 - Assigned to NPC Group
 - Both *senses* and *thinks*
 - Sends *commands* as events
- **Applications**
 - Protecting special units
 - Flanking
 - Covering fire
 - Leapfrogging advance

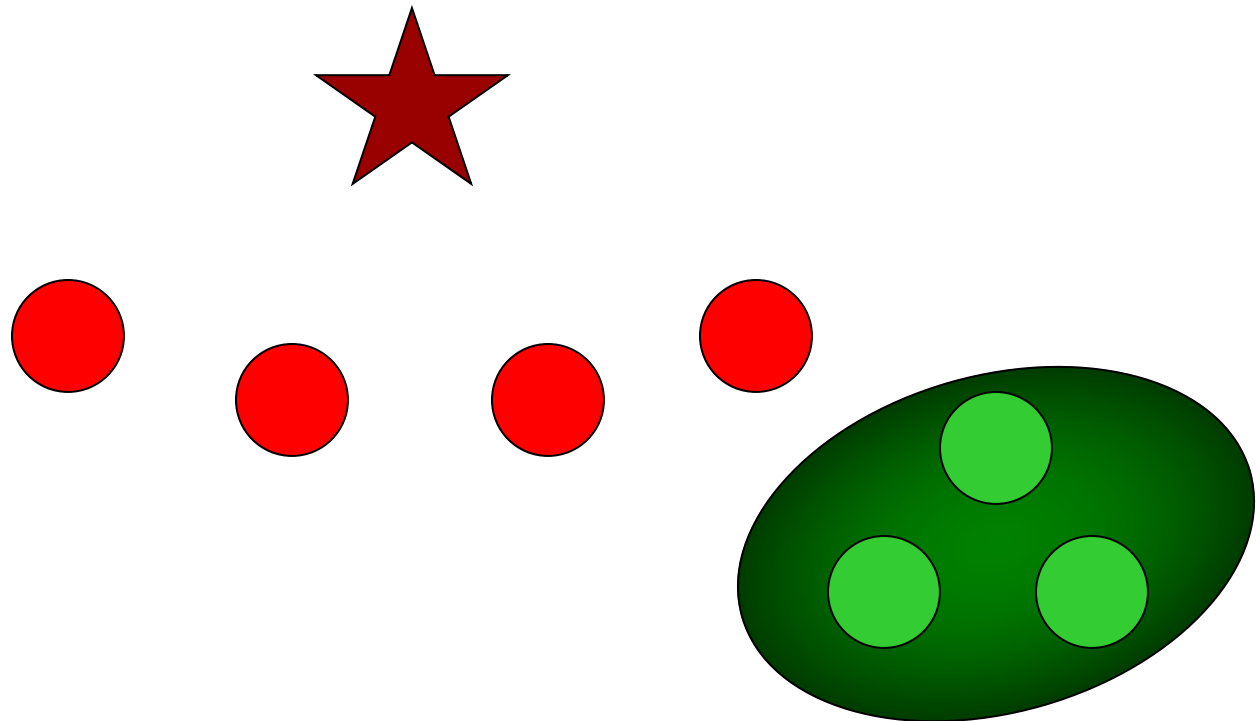


Protecting Special Units



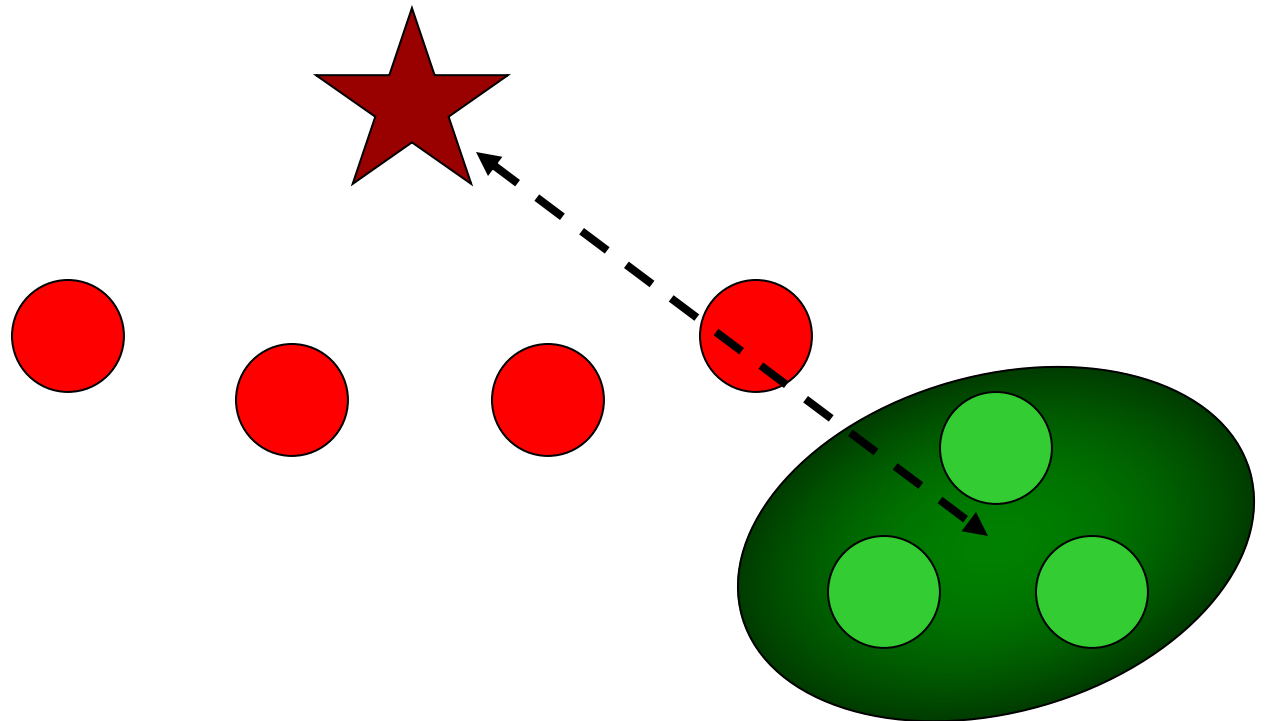
Slide courtesy of Dave Mark

Protecting Special Units



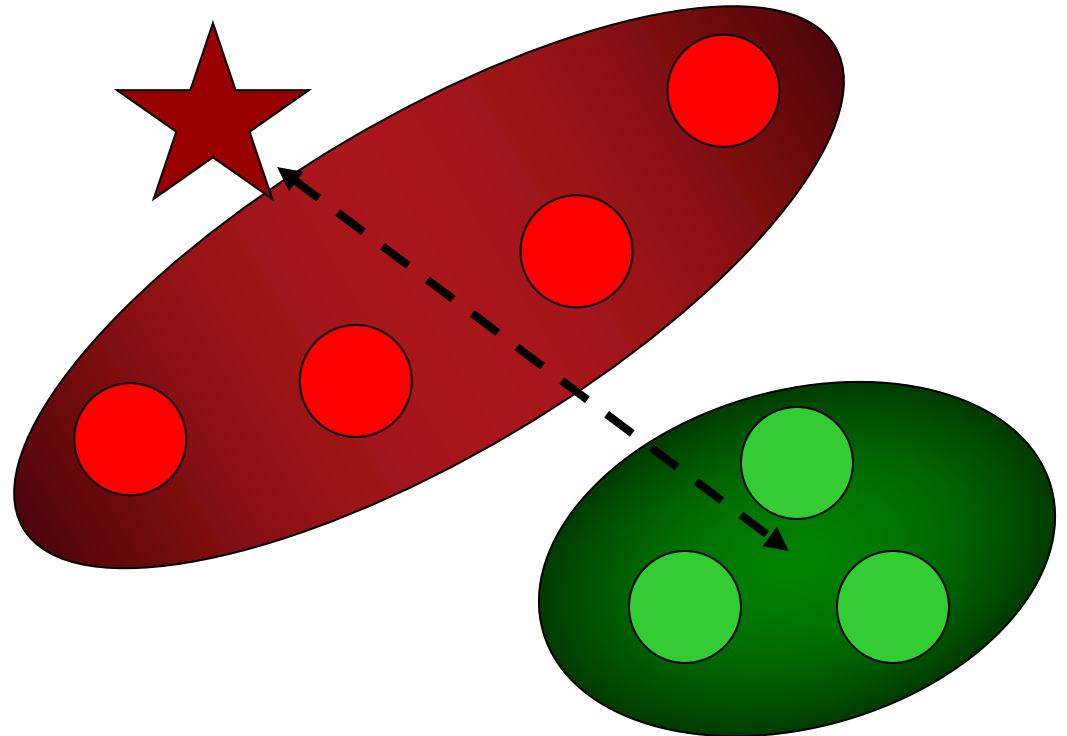
Slide courtesy of Dave Mark

Protecting Special Units



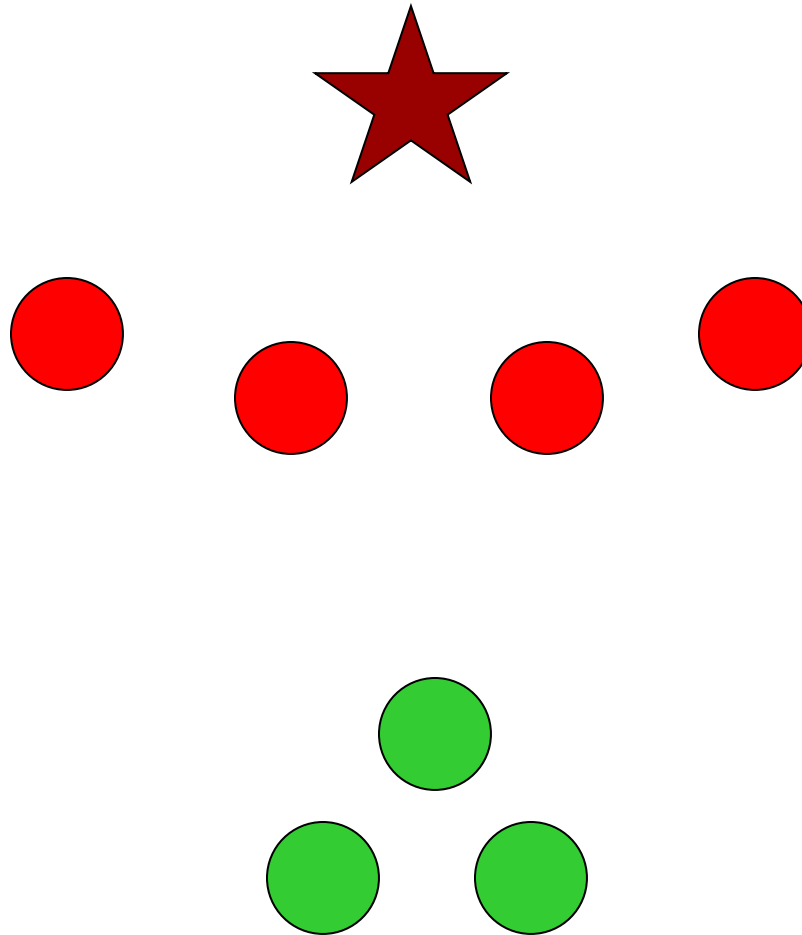
Slide courtesy of Dave Mark

Protecting Special Units



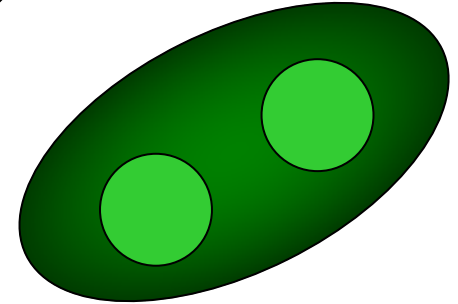
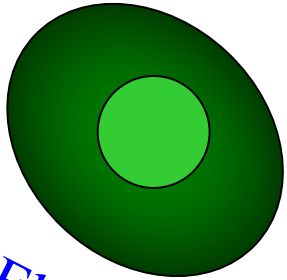
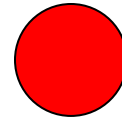
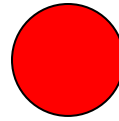
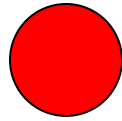
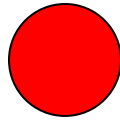
Slide courtesy of Dave Mark

Protecting Special Units



Slide courtesy of Dave Mark

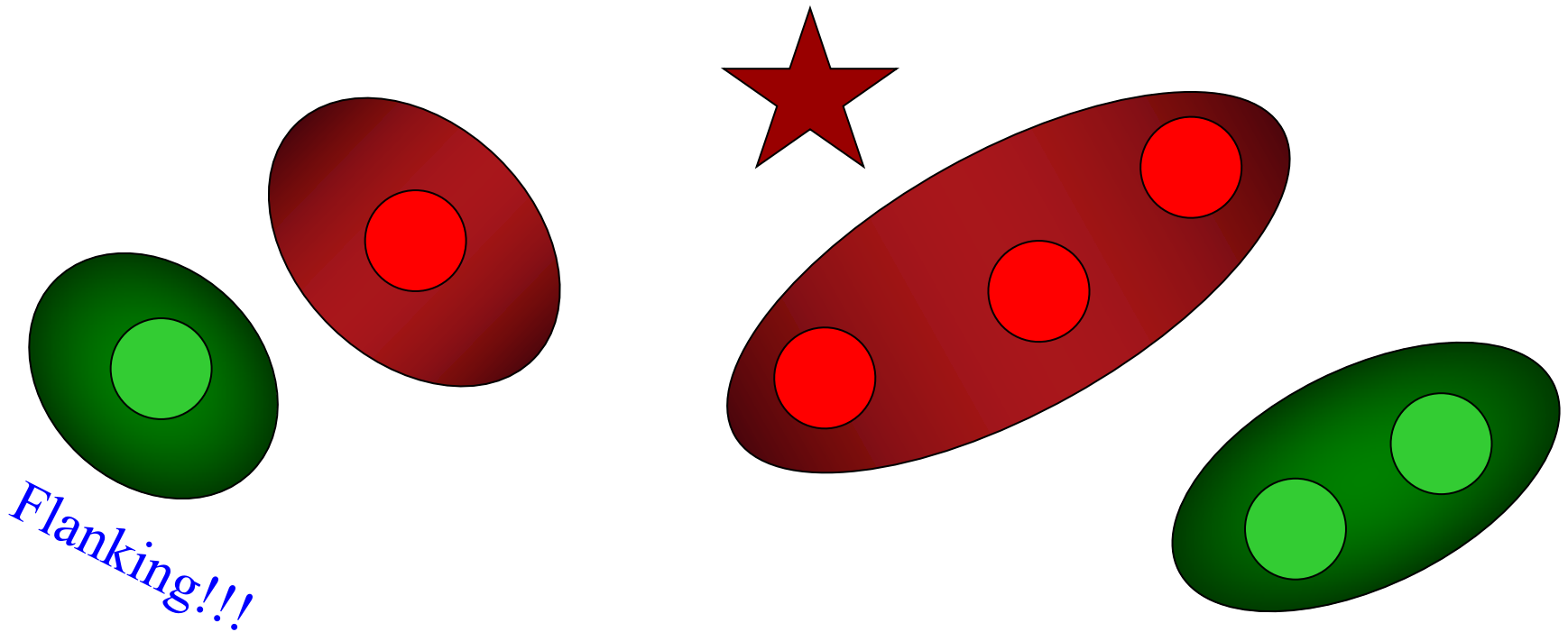
Protecting Special Units



Flanking!!!

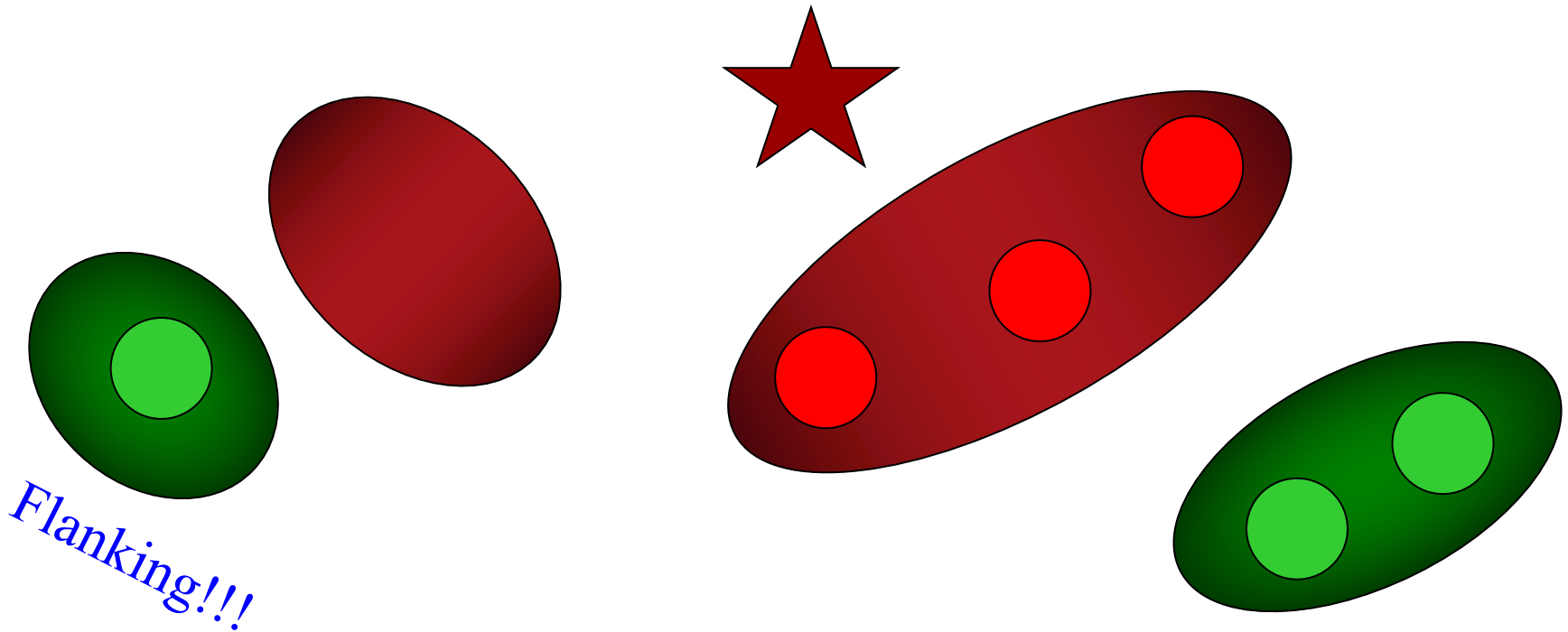
Slide courtesy of Dave Mark

Protecting Special Units



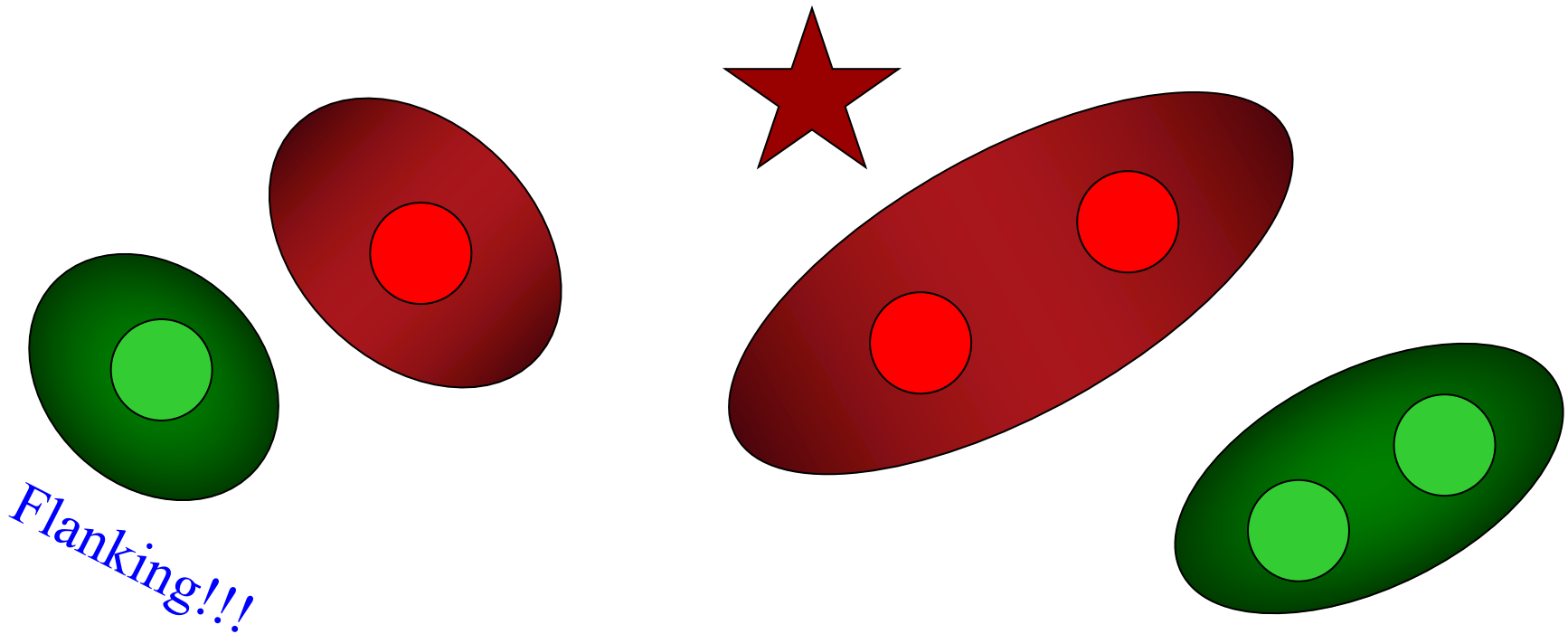
Slide courtesy of Dave Mark

Protecting Special Units



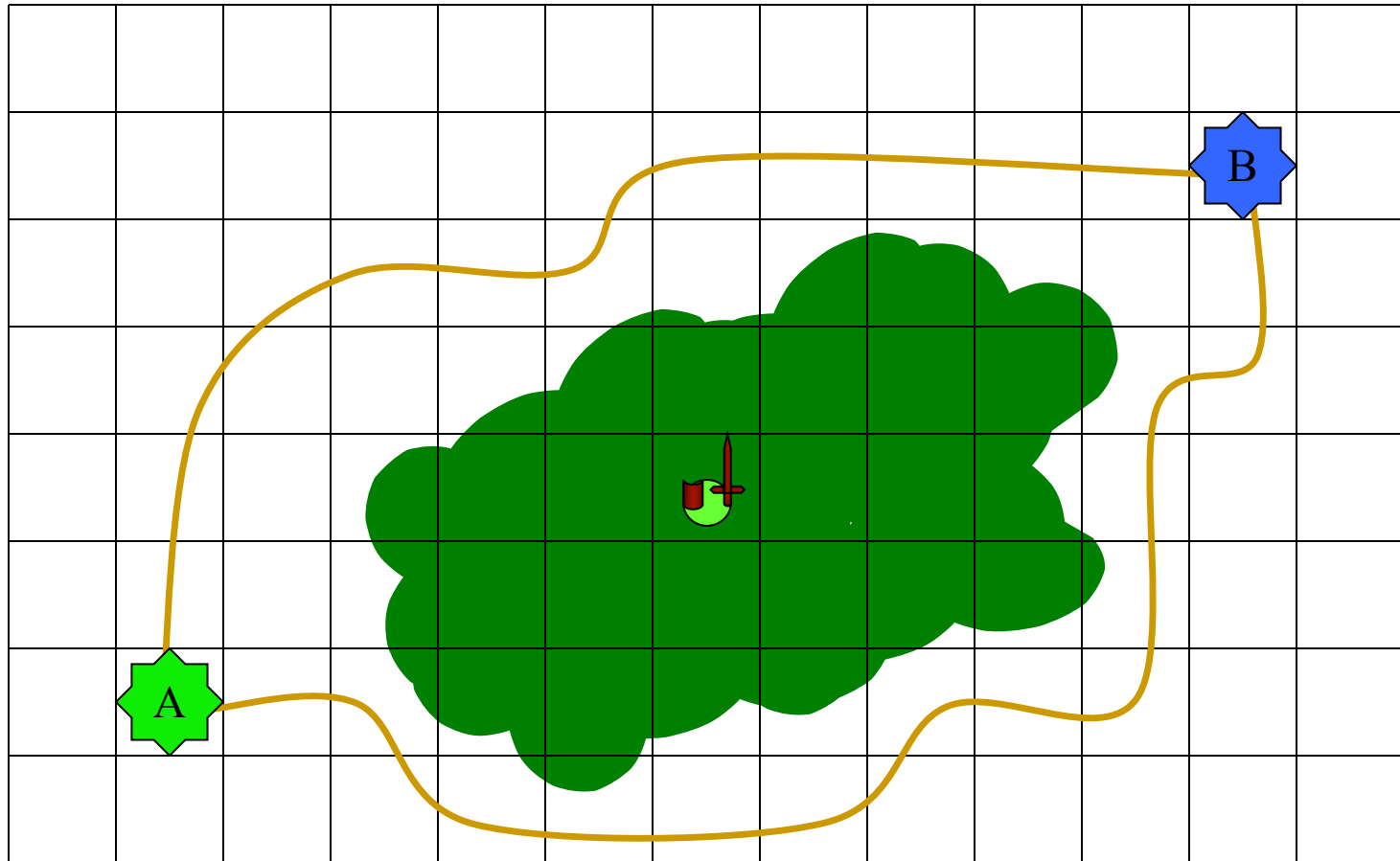
Slide courtesy of Dave Mark

Protecting Special Units



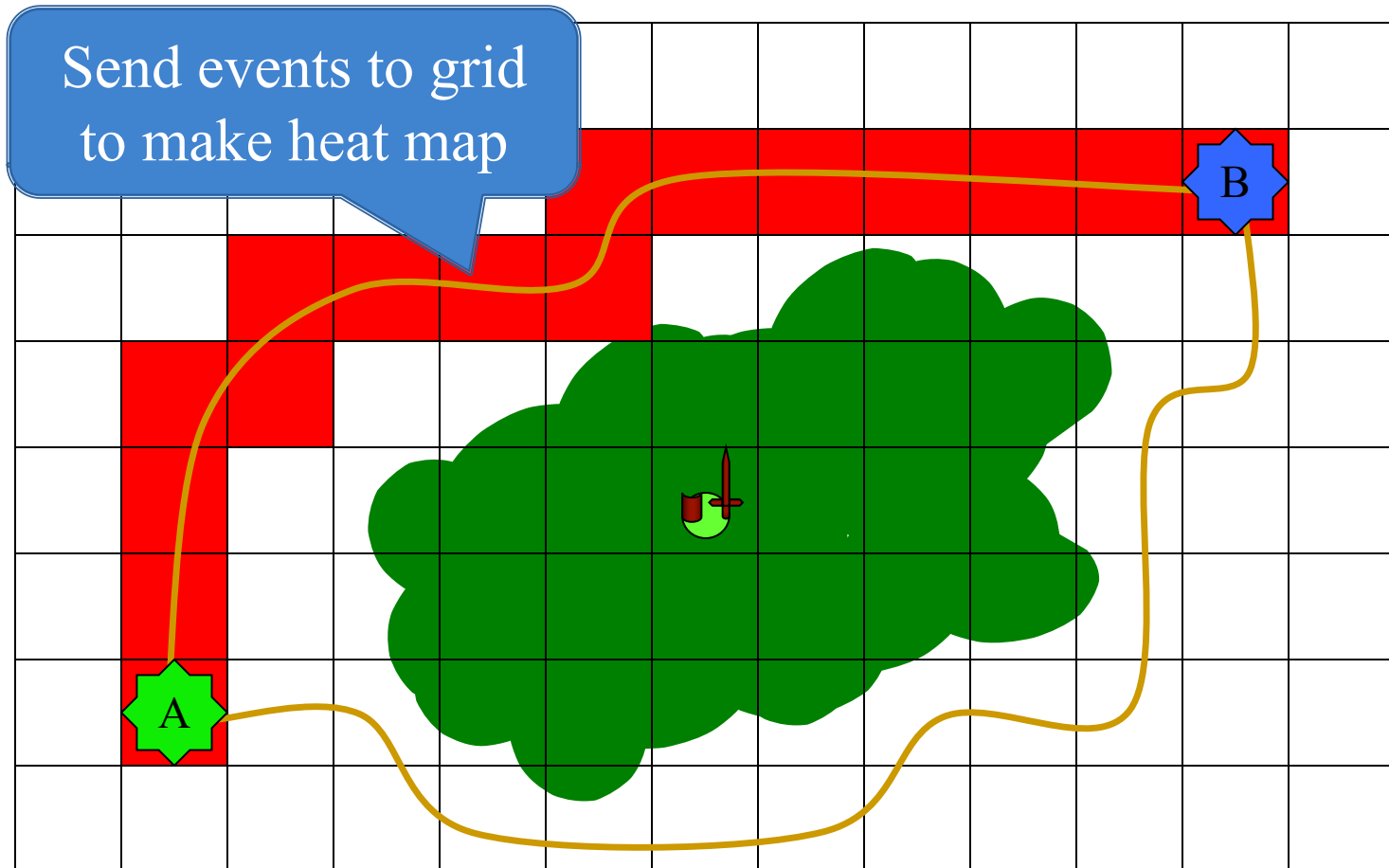
Slide courtesy of Dave Mark

Inversion: Influence Maps



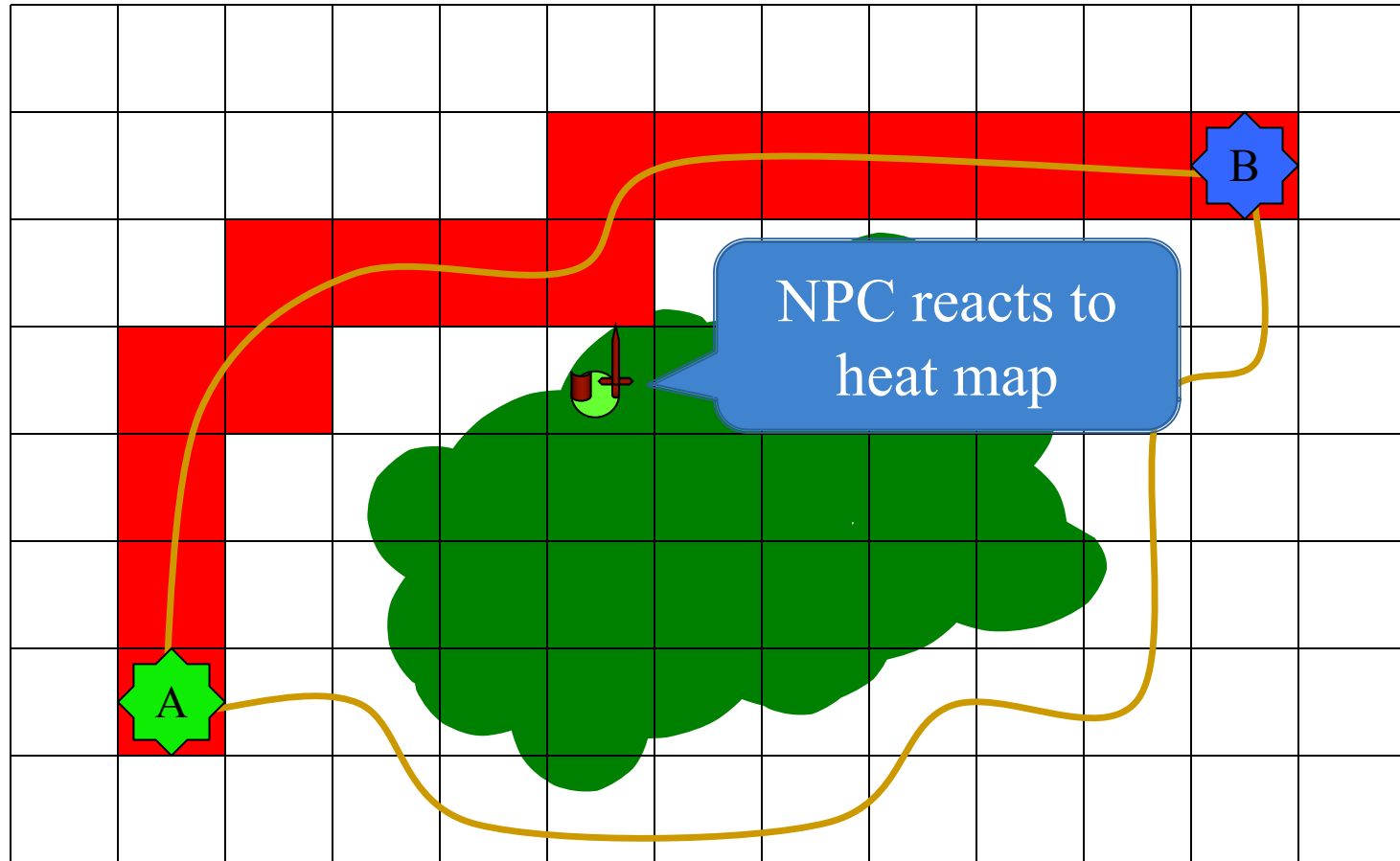
Slide courtesy of Dave Mark

Inversion: Influence Maps



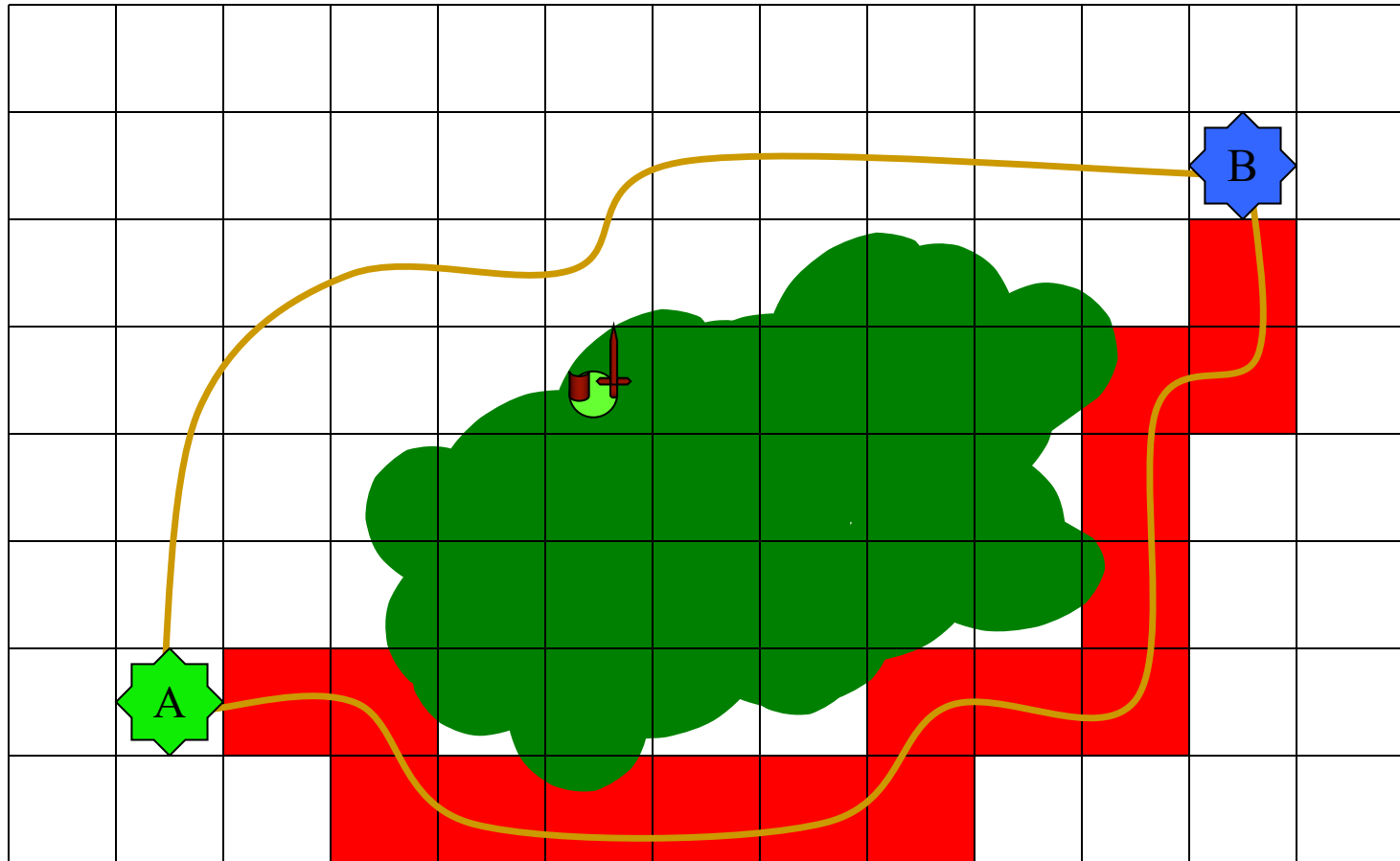
Slide courtesy of Dave Mark

Inversion: Influence Maps



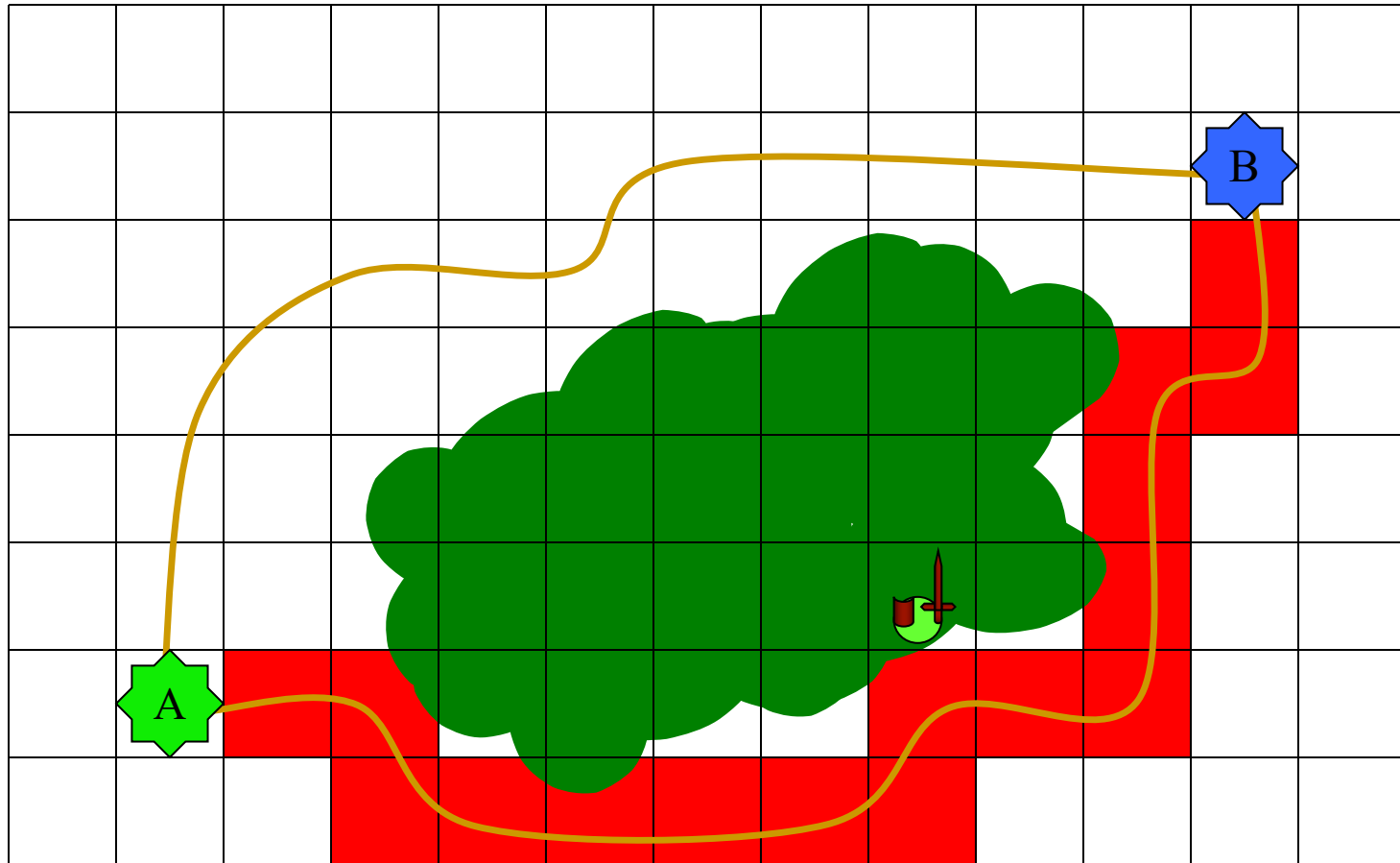
Slide courtesy of Dave Mark

Inversion: Influence Maps



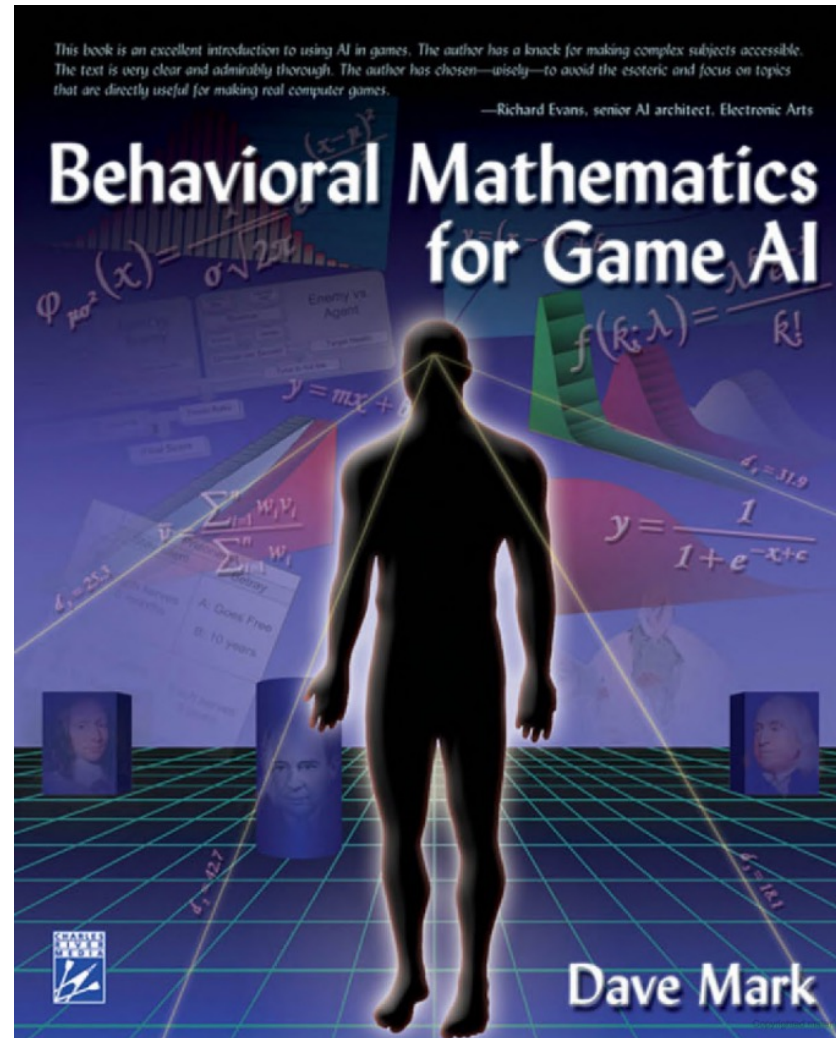
Slide courtesy of Dave Mark

Inversion: Influence Maps



Slide courtesy of Dave Mark

Resource for Sense Optimization



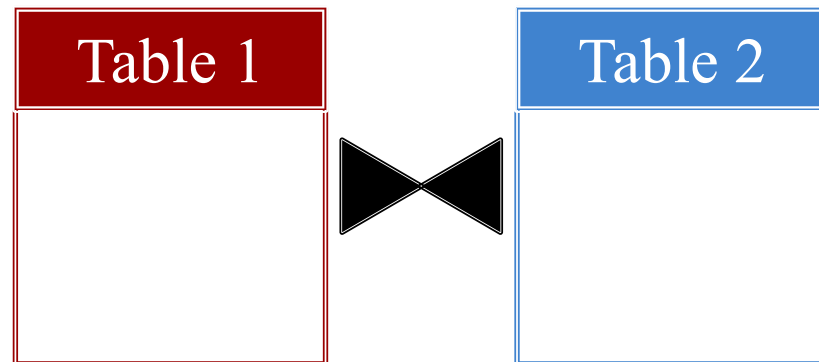
A Final Observation

for each entity x:

for each entity y:

if x senses y:

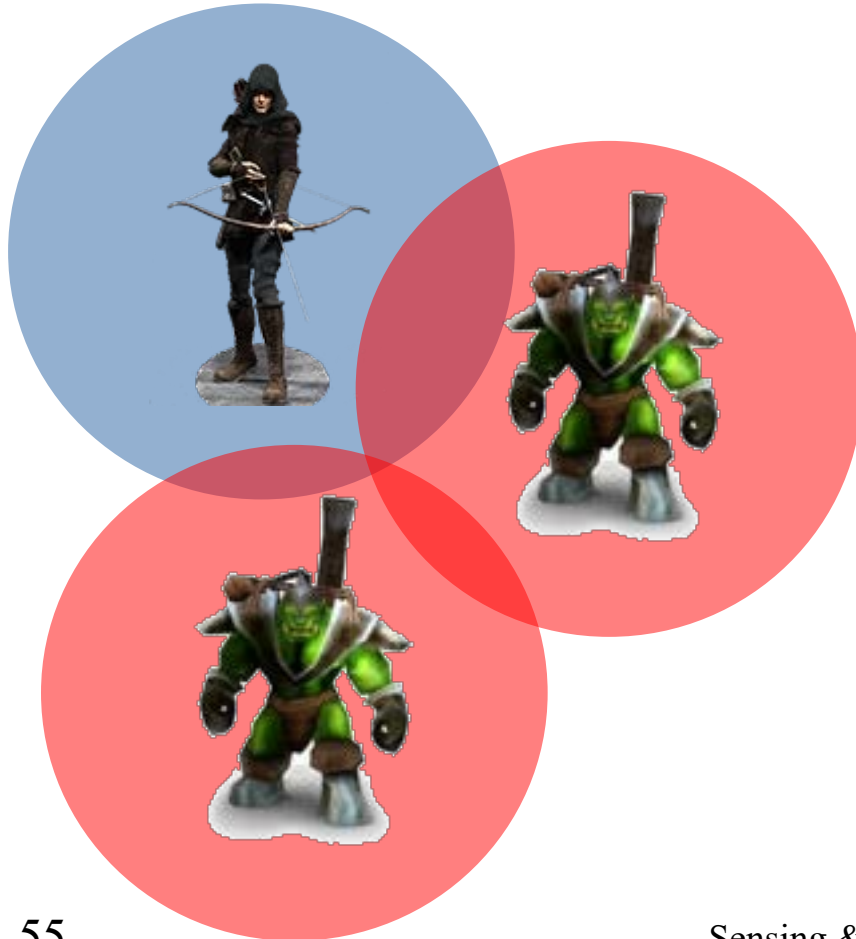
output event



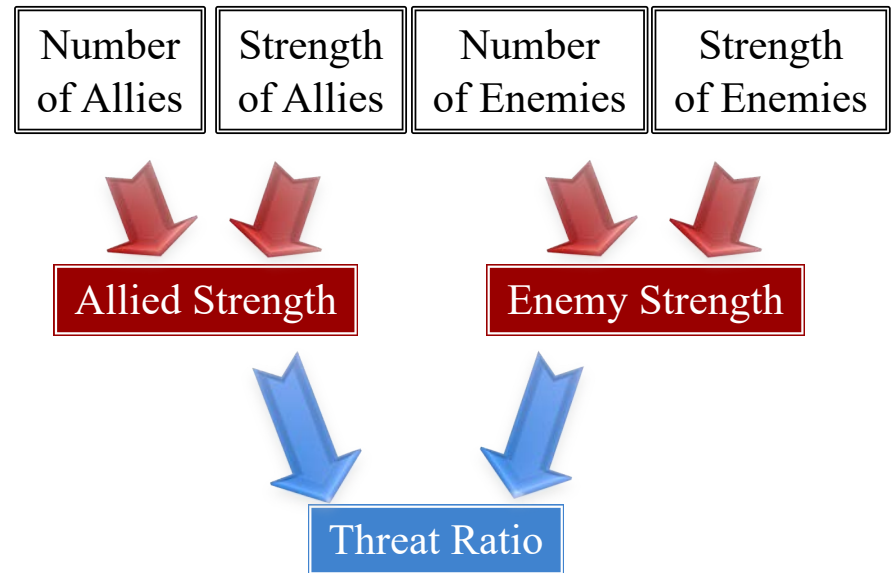
Sensing is a database **table join**

These are all DB Optimizations

Selection Pushing

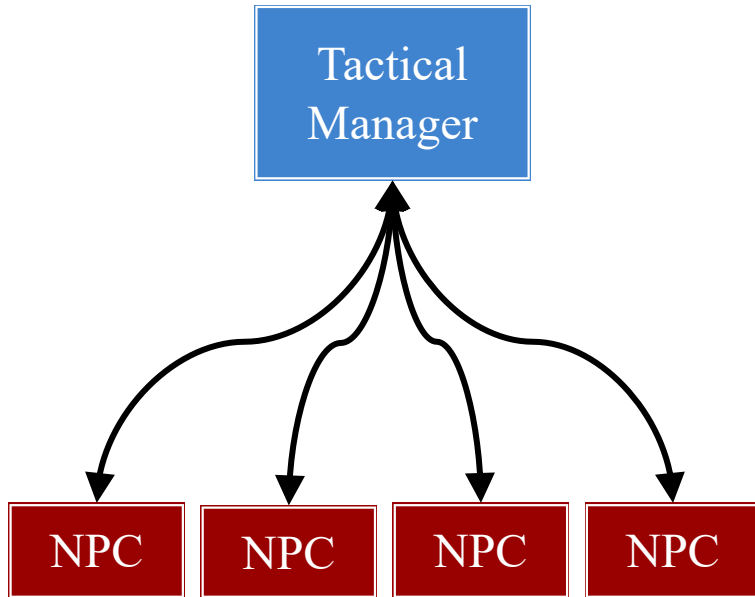


Aggregation Pushing

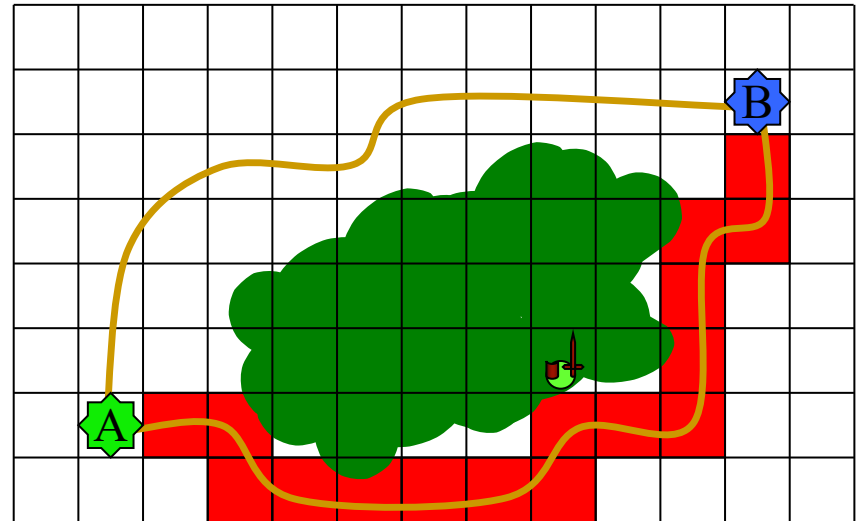


These are all DB Optimizations

Data Normalization



Query Rewriting



And This is Where it All Began

- *Scaling Games to Epic Proportions (SIGMOD 2007)*
 - Allow designers to write code naively as $O(n^2)$ loop
 - Use DB technology to optimize processing
- Requires that **behaviors** \ll **NPCs**
 - NPCs have different state, but use similar scripts
 - Each NPC is a tuple in database query
- **Challenge:** Making the language user-friendly
 - Requires major restrictions to language
 - Similar issue with Microsoft LINQ