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
  - Knights fear skeletons
  - Proportional to \( n \) seen
  - Count skeletons in view
    - \( O(n) \) to count skeletons
    - \( O(n^2) \) for all units

How Do We Make it Faster?

Sensing & Perception
**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 != 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
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*

- Motion Detection
- Peripheral Vision
- Short Distance
- Focused View
- Long Distance
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

NPC 1

Sense & Think

NPC 2

Act

incompatible

NPC 1

NPC 2

Sensing & Perception
Recall: Reducing Dependencies

Diagram showing the relationship between Actor1, Actor2, Gamestate, and Compute Sensing.
Recall: Reducing Dependencies

Compute Thinking

Actor1
Controller

GameState

Actor2
Controller

Actor1

Actor2
Solution: Event Driven AI

Finite State Machines

Decision Trees

Can support arbitrary (boolean) functions here

state 2

state 1

test
t
f
action
test
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

- **Pre-aggregated sense data**

  - Event Generation
    - NPC 1
    - NPC 2
  - Think
    - NPC 1
    - NPC 2
  - Act
    - NPC 1
    - NPC 2
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

Sense

Event Processor

Think

Behavior Controller

Act

Gameplay Controller

Events

Choices

Outcomes
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
Communicating Sense Events

First Hand
LOS
Sight & Sound
Communicating Sense Events

First Hand
LOS
Sight & Sound

First Hand
LOS
Sight & Sound
Communicating Sense Events

First Hand
LOS
Sight & Sound

Second Hand
Sight & Sound

First Hand
LOS
Sight & Sound
Communicating Sense Events

First Hand
LOS
Sight & Sound

Second Hand
Sight & Sound

Sensing & Perception
Sense Event Matching

- sound
- sight
- sound
- sound
- smell

Register events of interest

Event Handler

Game Loop
Sense Event Matching

Event Handler

Notify of any matching events

Game Loop

Check for any matching events
## Event Communication in LibGDX

<table>
<thead>
<tr>
<th>MessageDispatcher</th>
<th>Telegram</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Send with <code>dispatchMessage</code></td>
<td>• Stores the event message</td>
</tr>
<tr>
<td>• <code>delay</code> (0 if immediate)</td>
<td>• Entries of <code>dispatchMessage</code></td>
</tr>
<tr>
<td>• <code>sender</code> (can be null)</td>
<td>• Except for the <code>delay</code> value</td>
</tr>
<tr>
<td>• <code>target</code> (null for subscribers)</td>
<td>• Preaggregated sense in <code>data</code></td>
</tr>
<tr>
<td>• <code>type</code> (user defined int code)</td>
<td>• Received by <code>Telegraph</code></td>
</tr>
<tr>
<td>• <code>data</code> (object, like Box2D)</td>
<td>• Interface for the receiver</td>
</tr>
<tr>
<td>• Subscribe with <code>addListener</code></td>
<td>• Implemented by the NPC</td>
</tr>
<tr>
<td>• NPC to receive message</td>
<td>• One method: <code>handleMessage</code></td>
</tr>
<tr>
<td>• Type (int) to subscribe to</td>
<td></td>
</tr>
</tbody>
</table>
## Event Communication in LibGDX

<table>
<thead>
<tr>
<th><strong>MessageDispatcher</strong></th>
<th><strong>Telegram</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Send with <code>dispatchMessage</code></td>
<td>• Stores the event message</td>
</tr>
<tr>
<td>• <code>delay</code> (0 if immediate)</td>
<td>• Entries of <code>dispatchMessage</code></td>
</tr>
<tr>
<td>• <code>sender</code> (can be null)</td>
<td>• Except for the <code>delay</code> value</td>
</tr>
<tr>
<td>• <code>target</code> (null for subscribers)</td>
<td>• Preaggregated sense in <code>data</code></td>
</tr>
<tr>
<td>• <code>type</code> (user defined int code)</td>
<td>• Received by <code>Telegram</code></td>
</tr>
<tr>
<td>• <code>data</code> (object, like Box2D)</td>
<td>• Interface for the receiver</td>
</tr>
<tr>
<td>• Subscribe with <code>addListener</code></td>
<td>• Implemented by the NPC</td>
</tr>
<tr>
<td>• NPC to receive message</td>
<td>• One method: <code>handleMessage</code></td>
</tr>
<tr>
<td>• Type (int) to subscribe to</td>
<td></td>
</tr>
</tbody>
</table>
Separation Allows Many Optimizations

Sense | Think | Act

Event Processor | Behavior Controller | Gameplay Controller

Events | Choices | Outcomes
Compression: Aggregation Trees

- Number of Allies
- Strength of Allies
- Number of Enemies
- Strength of Enemies
- My Health
- Proximity to Leader
- Proximity to Base
Compression: Aggregation Trees

Number of Allies | Strength of Allies | Number of Enemies | Strength of Enemies

Allied Strength | Enemy Strength

My Health | Proximity to Leader

Proximity to Base

Slide courtesy of Dave Mark
Compression: Aggregation Trees

Slide courtesy of Dave Mark
Compression: Aggregation Trees

Number of Allies | Strength of Allies
Number of Enemies | Strength of Enemies

Allied Strength | Enemy Strength
Threat Ratio
Urgency
Proximity to Base
Proximity to Leader
My Health

Slide courtesy of Dave Mark

Sensing & Perception
Compression: Aggregation Trees

Number of Allies  | Strength of Allies  
Number of Enemies | Strength of Enemies

Allied Strength  | Enemy Strength

Threat Ratio  

Urgency  

Proximity to Leader  
Proximity to Base  
My Health  
My Morale

Slide courtesy of Dave Mark
Compression: Aggregation Trees

Number of Allies | Strength of Allies
---|---
Number of Enemies | Strength of Enemies

Allied Strength | Enemy Strength

Threat Ratio | Urgency

Proximity to Leader | Proximity to Base

My Health | My Morale

Retreat %

Slide courtesy of Dave Mark
Compression: Aggregation Trees

Number of Allies  Strength of Allies
Number of Enemies  Strength of Enemies

Allied Strength  Enemy Strength
Threat Ratio

My Health  My Morale  Proximity to Base  Proximity to Leader

Urgency

Retreat %

Computable independent of the NPC

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

Sensing & Perception
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

Sensing & Perception
Protecting Special Units

Flanking!!!

Slide courtesy of Dave Mark
Protecting Special Units

Flanking!!!
Protecting Special Units

Slide courtesy of Dave Mark
Protecting Special Units

Slide courtesy of Dave Mark

Sensing & Perception
Inversion: Influence Maps

Slide courtesy of Dave Mark
Inversion: Influence Maps

Send events to grid to make heat map

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

Behavioral Mathematics for Game AI

Dave Mark

This book is an excellent introduction to using AI in games. The author has a knack for making complex subjects accessible. The text is very clear and admirably thorough. The author has chosen wisely—to avoid the esoteric and focus on topics that are directly useful for making real computer games.

—Richard Evans, senior AI architect, Electronic Arts
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

- Number of Allies
- Number of Enemies
- Threat Ratio
- Allied Strength

### Aggregation Pushing

- Strength of Allies
- Strength of Enemies
- Enemy Strength
- Threat Ratio
These are all DB Optimizations

Data Normalization

Query Rewriting

Tactical Manager

NPC  NPC  NPC  NPC

A  B

NPC

NPC

NPC

NPC

NPC
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 $<$ NPC**
  - 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