Lecture 21

Character AI: Sensing & Perception
Take Away for Today

- 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?
Review: 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
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

<table>
<thead>
<tr>
<th>Time per tick</th>
<th>3 units</th>
<th>2 units</th>
<th>1 unit</th>
</tr>
</thead>
</table>

Sensing & Perception
Recall: Sensing Performance

- Sensing may be slow!
  - Consider \textit{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

How Do We Make it Faster?
Aggregation

- Idea taken from databases
  - Unordered set of information
  - Combine into single value
  - Used in statistical analysis
  - **Examples**: sum, avg, mode

- **Decomposable Aggregates**
  - Split the set up into subsets
  - Aggregate on each subset
  - Combine values from subsets
  - Only for some aggregates
Aggregation

- Idea taken from databases
- Unordered set of information
- Combine into single value
- Used in statistical analysis
- **Examples**: sum, avg, mode

**Decomposable Aggregates**
- Split the set up into subsets
- Aggregate on each subset
- Combine values from subsets
- Only for some aggregates

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Average</th>
<th>Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>6, 10</td>
<td>avg = 8</td>
<td>(16, 2)</td>
</tr>
<tr>
<td>5, 7</td>
<td>avg = 6</td>
<td>(12, 2)</td>
</tr>
<tr>
<td>9, 7</td>
<td>avg = 8</td>
<td>(16, 2)</td>
</tr>
<tr>
<td>3, 9</td>
<td>avg = 6</td>
<td>(12, 2)</td>
</tr>
</tbody>
</table>
Aggregation

- Idea taken from databases
  - Unordered set of information
  - Combine into single value
  - Used in statistical analysis
  - **Examples**: sum, avg, mode

- **Decomposable Aggregates**
  - Split the set up into subsets
  - Aggregate on each subset
  - Combine values from subsets
  - Only for some aggregates

\[
\begin{align*}
&6 & \text{avg} = 8 & (16,2) & \text{avg} = 7 & (28,4) \\
&10 & \text{avg} = 6 & (12,2) & \\
&5 & \text{avg} = 6 & (12,2) & \\
&7 & \text{avg} = 7 & (28,4) & \\
&9 & \text{avg} = 8 & (16,2) & \text{avg} = 7 & (28,4) \\
&7 & \text{avg} = 6 & (12,2) & \\
&3 & \text{avg} = 6 & (12,2) & \\
&9 & & & \\
\end{align*}
\]
Aggregation

- Idea taken from databases
  - Unordered set of information
  - Combine into single value
  - Used in statistical analysis
  - **Examples**: sum, avg, mode

- **Decomposable Aggregates**
  - Split the set up into subsets
  - Aggregate on each subset
  - Combine values from subsets
  - Only for some aggregates

\[
\begin{align*}
\text{avg} & = 8 \\
(16,2) & \quad \text{avg} = 6 \\
(12,2) & \quad \text{avg} = 7 \\
(28,4) & \quad \text{avg} = 7 \\
(56,8) & \quad \text{avg} = 7
\end{align*}
\]
Aggregation

- Idea taken from databases
- Unordered set of information
- Combine into single value
- Used in statistical analysis
- **Examples**: sum, avg, mode

**Decomposable Aggregates**
- Split the set up into subsets
- Allows for fast parallel computation
- Can combine values from subsets
- Only for some aggregates
AI and Aggregation Trees

Number of Allies  Number of Enemies
Strength of Allies  Strength of Enemies
   
   Allied Strength  Enemy Strength

Proximity to Base

My Health  Proximity to Leader

Slide courtesy of Dave Mark
AI and Aggregation Trees

Number of Allies
Strength of Allies

Number of Enemies
Strength of Enemies

Allied Strength

Enemy Strength

Threat Ratio

My Health
Proximity to Leader

Proximity to Base

Slide courtesy of Dave Mark
AI and Aggregation Trees

Slide courtesy of Dave Mark

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

- Allied Strength
- Enemy Strength

- Threat Ratio
- Proximity to Base
- Urgency

- My Health
- Proximity to Leader
AI and Aggregation Trees

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

Allied Strength
Enemy Strength
Threat Ratio
Urgency

My Health
Proximity to Leader
Proximity to Base
My Morale

Slide courtesy of Dave Mark
AI and Aggregation Trees

Number of Allies  Number of Enemies

Strength of Allies  Strength of Enemies

Allied Strength  Enemy Strength

Threat Ratio  Proximity to Leader

My Health  Proximity to Base

My Morale  Urgency

Retreat %

Slide courtesy of Dave Mark
AI and Aggregation Trees

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

Allied Strength | Enemy Strength

Threat Ratio

Urgency

Proximity to Leader

My Health

My Morale

Retreat %

Proximity to Base

Computable independent of the NPC

Slide courtesy of Dave Mark
Related Approach: Tactical Managers

- “Invisible NPC”
  - Assigned to NPC Group
  - Performs all *thinking*
  - NPCs just follow orders

- Applications
  - Protecting special units
  - Flanking
  - Covering fire
  - Leapfrogging advance
Protecting Special Units

Slide courtesy of Dave Mark

19

Thinking and Acting
Protecting Special Units

Slide courtesy of Dave Mark

Thinking and Acting
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

Flanking!!!

Slide courtesy of Dave Mark
Protecting Special Units

Flanking!!!
Performance: Loop Inversion

Normal Sensing

- Loop over all NPCs
- Check what NPC senses

Inverted Sensing

- Loop over sensations
- Send these to each NPC
Performance: Loop Inversion

Normal Sensing

- Loop over all NPCs
- Check what NPC senses

Inverted Sensing

- Loop over sensations
- Send these to each NPC

Works if number of sense events is small
Sense Events

- **Event**: encoded sense data
  - Tagged with sense type
  - Information self-contained
  - $O(n)$ data is aggregated
  - $O(1)$ to combine w/ NPC

- Sensing is **event matching**
  - Each event has a type
  - NPCs “register” for a type
  - Send NPC registered events
  - Check if event is relevant
Sense Event Matching

- Event Handler
  - Register events of interest
  - Game Loop
    - sound
    - sight
    - sound
    - sound
    - smell

Sensing & Perception
Sense Event Matching

Notify of any matching events

Event Handler

Check for any matching events

Game Loop
Event Handling 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 Handling 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 `Telegram`
  - Interface for the receiver
  - Implemented by the NPC
  - One method: `handleMessage`
Recall: S-T-A Architecture

Actor1
Controller

Actor2
Controller

Compute Sensing

GameState

Actor1

Actor2
Recall: S-T-A Architecture

Actor1 Controller

GameState

Actor2 Controller

Actor1

Actor2

Pass Event

Compute Sensing
Recall: S-T-A Architecture

Actor1 Controller — GameState — Actor2 Controller

Actor1 — Compute Thinking — Actor2
Sensing: Perception Groups

- **Vision**: limited field of view
  - Gives exact object location, information
  - Limited by obstacles and range
  - Little information (motion) at periphery

- **Sound**: omni-directional
  - Gives direction & distances
  - Requires you track the “sounds” actions make

- **Smell**: omni-directional
  - No direction or distance; *proximity* only
  - Requires you track the “smells” actions make
Case Study: *Thief* Series
Stealth tip: Use WALK to move slowly and very quietly. Use CREEP to move even more slowly and be completely silent.
Line-of-Sight in *Thief*

- **Peripheral Vision**
- **Short Distance**
- **Long Distance**
- **Focused View**
Line-of-Sight in *Thief*

- **Long Distance**
- **Focused View**
- **Short Distance**
- **Peripheral Vision**
- **Motion Detection**
Sounds in *Thief*

- “Easier” than vision
  - Primarily distance-based
  - Decays probabilistically
  - Tag with level of interest
- Sounds can be blocked
  - Not same as line-of-sight
  - Use *alternate level map*
  - Or *tag* your visible map
- Not physically realistic
  - Echoes? Reflections
Sounds in *Thief*

- “Easier” than vision
  - Primarily distance-based
  - Decays probabilistically
  - Tag with level of interest
- Sounds can be blocked
  - Not same as line-of-sight
  - Use *alternate level map*
  - Or *tag* your visible map
- Not physically realistic
  - Echoes? Reflections
Sounds in *Thief*

- Sounds are general purpose
  - Resuable framework
  - Code is lightweight
  - Encodes other senses
- **Example**: Smell
  - Treated as “pseudo-sound”
  - Generate like any sound
- Again, ignores other factors
  - Wind direction
  - Masking smells
Custom Data in Events

**Lightweight**

- Memory
- Target
- ThiefExposed
- Reference to target

**Heavyweight**

- Memory
- Target
- ThiefExposed
- Copy of target
# Custom Data in Events

<table>
<thead>
<tr>
<th>Lightweight</th>
<th>Heavyweight</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Advantages</strong></td>
<td><strong>Advantages</strong></td>
</tr>
<tr>
<td>- Fast to create event</td>
<td>- Can persist past frame</td>
</tr>
<tr>
<td>- No additional memory</td>
<td>- Can retain as memory</td>
</tr>
<tr>
<td><strong>Disadvantages</strong></td>
<td><strong>Disadvantages</strong></td>
</tr>
<tr>
<td>- Must be used immediately</td>
<td>- Must allocate memory</td>
</tr>
<tr>
<td>- Lost over frame boundary</td>
<td>- Object ownership is tricky</td>
</tr>
<tr>
<td>• Ideal for <strong>fast decisions</strong></td>
<td>• Ideal for <strong>communication</strong></td>
</tr>
</tbody>
</table>
Communicating Senses
Communicating Senses

First Hand
LOS
Sight & Sound

Sensing & Perception
Communicating Senses

First Hand
LOS
Sight & Sound
Communicating Senses

First Hand
LOS
Sight & Sound

Second Hand
Sight & Sound

First Hand
LOS
Sight & Sound
Alertness: Active Senses

High Alert

First Hand
LOS
Sight & Sound

Second Hand
Sight & Sound
Alertness: Active Senses

High Alert

First Hand
LOS
Sight & Sound

Medium Alert

Second Hand
Sight & Sound

First Hand
LOS
Sight & Sound

Sensing & Perception
Alertness: Active Senses

High Alert

First Hand
LOS
Sight & Sound

Medium Alert

First Hand
Sight & Sound

High Alert

First Hand
Sight & Sound

Second Hand
Sight & Sound

Sight & Sound

High Alert

First Hand
Sound
Alertness: Active Senses

High Alert

First Hand LOS
Sight & Sound

Second Hand Sight & Sound

Medium Alert

First Hand
Sight & Sound

High Alert
Alertness: Active Senses

- High Alert
  - First Hand LOS Sight & Sound
  - Second Hand Sight & Sound
- Medium Alert
  - First Hand Sound
- High Alert
  - First Hand Sound
Spatial Optimizations

- Restrict to nearby NPCs
  - Have detection range
  - Limits events sensed
  - Easy to combine with event matching system

- Works in both directions
  - Nimbus: “can see” radius
  - Aura: “can be seen” radius

- Area of interest management
Spatial Optimizations

- Restrict to nearby NPCs
  - Have detection range
  - Limits events sensed
  - Easy to combine with event matching system

- Works in both directions
  - Nimbus: “can see” radius
  - Aura: “can be seen” radius
  - Area of interest management
Spatial Optimizations

- Restrict to nearby NPCs
  - Have detection range
  - Limits events sensed
  - Easy to combine with event matching system

- Works in both directions
  - Nimbus: “can see” radius
  - Aura: “can be seen” radius
  - Area of interest management
Thief: Sense Events and Aggregation

- Position
- Lighting
- Movement
- Exposure
- Visibility
- Sound System
- Sound Queue
- Non-specific Spatial Events
- Viewcone Selector
- Viewcone
- Look
- Listen
- Sense Pulse Receiver
- Ramp Up Delay
- Cool-down Capacitor
- Inter-Agent Communication
- Inter-Agent Observation
- Sense Links

Game Mechanics and Configuration
Summary

- Sensing is the most expensive part of AI
  - Each character “looks” at every object in game
  - Often leads to $O(n^2)$ behavior (bad!)

- Can **optimize** sense gathering
  - Aggregation is amenable to parallelization
  - Can piggyback some data onto pathfinding

- Event matching **inverts** the sensing problem
  - Creation of sense makes a data event
  - Forward event to “relevant” NPCs