Lecture 22

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

[Image: Character design with thought bubble]

- **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
Recall: Sensing Performance

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

- Example: morale
  - $n$ knights, $n$ skeletons
  - Knights fear skeletons
  - Proportional to the number of objects seen

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

---

**How Do We Make it Faster?**

<table>
<thead>
<tr>
<th>Time per tick</th>
<th>3 units</th>
<th>2 units</th>
<th>1 unit</th>
</tr>
</thead>
</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
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>Values</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>12</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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>avg = 8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(16,2)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>avg = 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(12,2)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>avg = 7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(28,4)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>avg = 8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(16,2)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>avg = 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(12,2)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>avg = 7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(28,4)</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

6
10
5
7
9
3
9

```
avg = 8 (16,2)
avg = 6 (12,2)
```

```
avg = 7 (28,4)
```

```
avg = 7 (56,8)
```
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
- Calculate averages from subsets
- Only for some aggregates

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

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

My Health | Proximity to Leader

Proximity to Base

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

Proximity to Leader

My Health

Proximity to Base

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  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

Urgency

Slide courtesy of Dave Mark
AI and Aggregation Trees

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  Proximity to Base

Urgency  My Health  Proximity to Leader

My Morale  Retreat %

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

Urgency | Proximity to Base

My Morale

Retreat %

Computable independent of the NPC

Slide courtesy of Dave Mark
Idea: Loop Inversion

Normal Sensing

- Loop over all NPCs
- Check what NPC senses

Inverted Sensing

- Loop over sensations
- Send these to each NPC
Idea: 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

---

**Pre-aggregated**

---

Sensing & Perception
Sense Event Matching

Register events of interest

Event Handler

Game Loop

sound
sight sound
sound smell
Sense Event Matching

Event Handler

Notify of any matching events

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 `Telegram`
  - 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
- GameState
- Compute Sensing
- Actor1
- Actor2
Recall: S-T-A Architecture

![Diagram of S-T-A Architecture]

- Actor1 Controller
- Actor2 Controller
- GameState

Pass
Event

Thinking and Acting
Recall: S-T-A Architecture

Actor1
Controller

Actor1

GameState

Actor2
Controller

Actor2

Compute
Thinking

Thinking and Acting
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**
- **Focused View**
- **Long Distance**
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

Lightweight

- Advantages
  - Fast to create event
  - No additional memory

- Disadvantages
  - Must be used immediately
  - Lost over frame boundary

- Ideal for **fast decisions**

Heavyweight

- Advantages
  - Can persist past frame
  - Can retain as memory

- Disadvantages
  - Must allocate memory
  - Object ownership is tricky

- Ideal for **communication**
Communicating Senses
Communicating Senses

First Hand
LOS
Sight & Sound
Communicating Senses

First Hand
LOS
Sight & Sound

First Hand
LOS
Sight & Sound

Sensing & Perception
Communicating Senses

First Hand
LOS
Sight & Sound

Second Hand
LOS
Sight & Sound
Communicating Senses

First Hand
LOS
Sight & Sound

Second Hand
Sight & Sound

?
Alertness: Active Senses

High Alert

First Hand LOS
Sight & Sound

Second Hand
Sight & Sound

First Hand
LOS
Sight & Sound

Sensing & Perception
Alertness: Active Senses

High Alert

First Hand
LOS
Sight & Sound

Second Hand
Sight & Sound

Medium Alert

First Hand
LOS
Sight & Sound

Sensing & Perception
Alertness: Active Senses

High Alert

First Hand
LOS
Sight & Sound

Second Hand
Sight & Sound

Medium Alert

First Hand
LOS
Sight & Sound

First Hand
Sound

High Alert

Sensing & Perception

45
Alertness: Active Senses

High Alert

First Hand LOS Sight & Sound

Second Hand Sight & Sound

Medium Alert

First Hand LOS Sight & Sound

Sensing & Perception
Alertness: Active Senses

High Alert
First Hand LOS Sight & Sound
Second Hand Sight & Sound

Medium Alert
First Hand LOS Sight & 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
- Game Mechanics and Configuration
- Inter-Agent Communication
- Inter-Agent Observation
- Sense Links
Influence Maps: Pathfinding and AI

Slide courtesy of Dave Mark
Influence Maps: Pathfinding and AI

Slide courtesy of Dave Mark
Influence Maps: Pathfinding and AI

Slide courtesy of Dave Mark
Implementing Influence Maps

- Use the pathfinding **grid**
  - Track movement in square
  - Track if friend or foe

- Keep count as a **queue**
  - Count is sum of queue
  - Allows us to “time out”
  - Otherwise, marked forever

- Use queue as a **predictor**
  - Look at rate of change
  - Also valuable for AI

Sensing is at grid, not NPC
Advantages of Influence Maps

Slide courtesy of Dave Mark
Advantages of Influence Maps

Slide courtesy of Dave Mark
Advantages of Influence Maps

Slide courtesy of Dave Mark
Advantages of Influence Maps

Influence data reflects changes

Slide courtesy of Dave Mark
Advantages of Influence Maps

Slide courtesy of Dave Mark

A

B

Sensing & Perception
Advantages of Influence Maps

Add *a priori* assumptions

Slide courtesy of Dave Mark
Advantages of Influence Maps

Slide courtesy of Dave Mark
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