15-466
Computer Game Programming

AI Game Architecture

Maxim Likhachev
Robotics Institute
Carnegie Mellon University
AI in Game Architecture

from “Artificial Intelligence for Games” by I. Millington & J. Funge
decides which AI tasks to run, when, and for how long

from “Artificial Intelligence for Games” by I. Millington & J. Funge
AI in Game Architecture

Why “for how long”?

decides which AI tasks to run, when, and for how long

from “Artificial Intelligence for Games” by I. Millington & J. Funge
AI in Game Architecture

gets the right world info and in the right format to appropriate AI modules

from “Artificial Intelligence for Games” by I. Millington & J. Funge
AI in Game Architecture

translates AI decisions into the actual (animated) actions
AI in Game Architecture

AI gets given processor time

Execution management

AI gets its information

Group AI

Strategy

Decision making

Character AI

Movement

World interface

Content creation

Scripting

AI has implications for related technologies

Animation

Physics

AI gets turned into on-screen action

translates AI decisions into the actual (animated) actions

translates actions into changes to the world
Execution Management

• Modern AI processing in games can often take 5-50% of processing

• Someone needs to control the limited processing time
  - dividing available time among AI tasks

    - scheduling algorithms to work over time

    - making preferences for “important” characters
Execution Management

- Dividing available time among AI tasks

*from Age of Empires*
Execution Management

• Dividing available time among AI tasks

How is it different from graphics-related tasks that are also done every frame?

from Age of Empires
Execution Management

• Dividing available time among AI tasks

(from “Artificial Intelligence for Games” by I. Millington & J. Funge)
Execution Management

• Dividing available time among AI tasks: **Frequency-based scheduler**
  - takes in $N$ tasks, each task $T_i$ has execution frequency $f_i$
  - runs each task $T_i$ at its frequency

*Any issues?*
Execution Management

• Dividing available time among AI tasks: **Frequency-based scheduler**
  - takes in $N$ tasks, each task $T_i$ has execution frequency $f_i$
  - runs each task $T_i$ at its frequency

*Example:* $f(A) = 2, f(B) = 4, f(C) = 8$

*from “Artificial Intelligence for Games” by I. Millington & J. Funge*
Execution Management

• Dividing available time among AI tasks: **Frequency-based scheduler**
  - takes in $N$ tasks, each task $T_i$ has execution frequency $f_i$
  - runs each task $T_i$ at its frequency

**Example:** $f(A) = 2$, $f(B)=4$, $f(C)=8$

Any ideas how to mitigate this?

from “Artificial Intelligence for Games” by I. Millington & J. Funge
• Dividing available time among AI tasks: **Frequency-based scheduler**

  - takes in $N$ tasks, each task $T_i$ has execution frequency $f_i$
  - runs each task $T_i$ at its frequency and with its own phase $p_i$

**Example:**

*Initialization:*

```plaintext
for i = 1...100
    task[i].freq = 10;
    task[i].phase = i;
```

*Main Loop:*

```plaintext
for i = 1...100
    task[i].freq % (frame+task[i].phase) == 0
    run task[i];
```
Execution Management

- Dividing available time among AI tasks: **Frequency-based scheduler**

  - computing a good quality $p_i$ according to Wright’s method:
  
  whenever new task $T_i$ with frequency $f_i$ needs to be scheduled
  
  run the scheduler for $K$ frames
  
  pick frame $F < f_i$ with least total # of tasks executed at $F,F+f_i,...$  
  
  set phase $p_i$ to $F$

**Example**: Task with $f=5$ comes in

from “Artificial Intelligence for Games” by I. Millington & J. Funge
Execution Management

• Dividing available time among AI tasks: Frequency-based scheduler
  - computing a good quality $p_i$ according to Wright’s method:

  whenever new task $T_i$ with frequency $f_i$ needs to be scheduled
  run the scheduler for $K$ frames
  pick frame $F < f_i$ with least total # of tasks executed at $F, F+f_i, ...$
  set phase $p_i$ to $F$

  Example: Task with $f=5$ comes in $F=3$

  from “Artificial Intelligence for Games” by I. Millington & J. Funge
Execution Management

- Dividing available time among **Interruptible** AI tasks
  - Interruptible tasks can be paused and resumed as time allows
    - can use threads (less common in games)
    - can use software threads (tasks are supposed to return as soon as their time expires)

*Load-balancing Scheduler:*
  during each frame
  it distributes time among the tasks scheduled for this frame
Execution Management

• Dividing available time among **Interruptible** AI tasks
  - Interruptible tasks can be paused and resumed as time allows
    - can use threads (less common in games)
    - can use software threads (tasks are supposed to return as soon as their time expires)

**Priority Scheduler:**
  during each frame
  it distributes time among the tasks scheduled for this frame proportionally to the priority of the tasks
Execution Management

- Dividing available time among **Interruptible** AI tasks
  - Interruptible tasks can be paused and resumed as time allows
  - Can use threads (less common in games)
  - Can use software threads (tasks are supposed to return as soon as their time expires)

**Hierarchical Scheduler:**

*from “Artificial Intelligence for Games” by I. Millington & J. Funge*
Execution Management

• **Anytime AI tasks**
  - compute the best solution they can within provided time
  - can often improve the solution within the subsequent (next cycle) executions
  - common in Path Finding

- we will learn more about it in the later classes
Execution Management

• **Anytime AI tasks**
  - compute the best solution they can within provided time
  - can often improve the solution within the subsequent (next cycle) executions
  - common in Path Finding
Execution Management

• **Level of Details-based management**
  - less important characters run at lower frequencies
  - less important characters run at lower priorities
  - less important characters get less AI tasks and behaviors to run
Execution Management

- **Level of Details-based management**
  - less important characters run at lower frequencies
  - less important characters run at lower priorities
  - less important characters get less AI tasks and behaviors to run

*Example?
World Interfacing

• To be believable, characters need to know the “right” world information at the “right” time
World Interfacing

• Information can be received via polling or event processing

  - polling (e.g., Is this door open or closed?)

  - events (e.g., need to know whenever the door closes)
    - tasks subscribe to events of interest
    - other entities send out events
    - event manager receives and dispatches messages to those tasks that subscribed to them
World Interfacing

• Getting the “right” information:
  - the believable sensory ability of AI characters
World Interfacing

• Getting the “right” information:
  -the believable sensory ability of AI characters
  -typical senses used in games (in the order of popularity): sight, touch, hearing, smell
World Interfacing

• Getting the “right” information:
  - the believable sensory ability of AI characters

  - typical senses used in games (in the order of popularity): sight, touch, hearing, smell

Any others?
World Interfacing

• Simulating realistic sight
  - sight cone

from Doom
World Interfacing

• Simulating realistic sight
  - sight cone

from “Artificial Intelligence for Games” by I. Millington & J. Funge
World Interfacing

- Simulating realistic sight cone

theoretically:
- about 120° vertically
- about 220° horizontally

in reality, it is much less

What does it depend on?

from “Artificial Intelligence for Games” by I. Millington & J. Funge
World Interfacing

• Simulating realistic sight
  - sight cone

\[ \theta \]

commonly used cone: 60°

from “Artificial Intelligence for Games” by I. Millington & J. Funge
World Interfacing

- Simulating realistic sight, other factors (in the order of popularity):
  - line-of-sight

_from Splinter Cell_
World Interfacing

• Simulating realistic sight, other factors (in the order of popularity):
  - line-of-sight
  - distance (typically, hard limit)

*from Ghost Recon*
World Interfacing

• Simulating realistic sight, other factors (in the order of popularity):
  - line-of-sight
  - distance (typically, hard limit)
  - brightness

from Splinter Cell
• Simulating realistic sight, other factors (in the order of popularity):
  - line-of-sight
  - distance (typically, hard limit)
  - brightness
  - contrast with background

*from Ghost Recon*
World Interfacing

• Simulating realistic hearing
  - if implemented, then it typically travels at 100m/sec for some distance

from Conflict: Desert Storm
World Interfacing

- Simulating realistic touch
  - typically, limited to collision checking
  - sometimes used in stealthy games

_from Thief: The Dark Project_
World Interfacing

• Simulating smell
  - distance-limited, slow-speed propagation
  - rarely used in games

from Alien vs. Predator
World Interfacing

- Managing sense signals and sensors: Region Sense Manager
  - potential sensors are registered

```
Region Sense Manager

A
position, orientation, sensor capabilities

B
position, orientation, sensor capabilities
```
• Managing sense signals and sensors: Region Sense Manager
  - potential sensors are registered
  - signals from emitters received
World Interfacing

- Managing sense signals and sensors: **Region Sense Manager**
  - Aggregation phase: find all sensors within the max range of the emitter given its modality
  - Testing phase: test against their sensor capabilities

![Diagram](attachment:region_sense_manager_diagram.png)

- Position, orientation, sensor capabilities
- Modality (vision, sound, smell)
- Intensity
- Position

**Region Sense Manager**

- Node A
- Node B
- Node e1

Maxim Likhachev
Carnegie Mellon University
World Interfacing

• Managing sense signals and sensors: **Region Sense Manager**
  - aggregation phase: find all sensors within the max range of the emitter given its modality
  - testing phase: test against their sensor capabilities
  - notification phase: notify sensors that pass at “right” times

```
position, orientation, sensor capabilities

Region Sense Manager

modality (vision, sound, smell)
intensity
position

A

el is sensed

B

el

e1 is sensed

position, orientation, sensor capabilities
```
World Interfacing

• Managing sense signals and sensors: Region Sense Manager
  - aggregation phase: find all sensors within the max range of the emitter given its modality
  - testing phase: test against their sensor capabilities
  - notification phase: notify sensors that pass at “right” times

Example: attenuation=0.9/meter

from “Artificial Intelligence for Games” by I. Millington & J. Funge
World Interfacing

- Managing sense signals and sensors: **Region Sense Manager**
  - aggregation phase: find all sensors within the max range of the emitter given its modality
  - testing phase: test against their sensor capabilities
  - notification phase: notify sensors that pass at "right" times

**Example:** attenuation = 0.9/meter

*Issues leading to non-realistic effects?*

*vision signals are tested for other line-of-sight, sight cone,...*

*from “Artificial Intelligence for Games” by I. Millington & J. Funge*
World Interfacing

• Managing sense signals and sensors: **Region Sense Manager**
  - aggregation phase: find all sensors within the max range of the emitter given its modality
  - testing phase: test against their sensor capabilities
  - notification phase: notify sensors that pass at “right” times

*Issues leading to non-realistic effects?*

*from “Artificial Intelligence for Games” by I. Millington & J. Funge*
World Interfacing

- Managing sense signals and sensors: **Region Sense Manager**
  - aggregation phase: find all sensors within the max range of the emitter given its modality
  - testing phase: test against their sensor capabilities
  - notification phase: notify sensors that pass at “right” times

*from “Artificial Intelligence for Games” by I. Millington & J. Funge*
World Interfacing

• Managing sense signals and sensors: Finite Element Model Sense Manager
  - split the space into nodes (e.g., rooms)
  - compute sense graphs (sight, sound and smell graphs)
  - edges have attenuation and distance values

Similar for modeling video cameras in Sight graph
World Interfacing

• Managing sense signals and sensors: **Finite Element Model Sense Manager**
  - split the space into nodes
  - compute sense graphs (sight, sound, smell graphs)
  - edges have attenuation and distance values

For sight graph, no intermediate nodes – all line-of-sight nodes have direct edges

Why?
World Interfacing

• Managing sense signals and sensors: **Finite Element Model Sense Manager**
  - split the space into nodes
  - compute sense graphs (sight, sound, smell graphs)
  - edges have attenuation and distance values

*For sight graph, additional line-of-sight tests have to be done in real-time*

*Why?*
World Interfacing

- Managing sense signals and sensors: **Finite Element Model Sense Manager**
  - split the space into nodes
  - compute sense graphs (sight, sound, and smell graphs)
  - edges have attenuation and distance values

*For sight graph, additional line-of-sight tests have to be done in real-time*  
*Why?*
World Interfacing

• Managing sense signals and sensors: **Finite Element Model Sense Manager**
  
  - similar aggregation, testing, notification phases except…
  
  - for Sight: all aggregated nodes of signal in $u$ are: sensors in $u$ and its immediate neighbors according to Sight graph.
• Managing sense signals and sensors: **Finite Element Model Sense Manager**
  
  - similar aggregation, testing, notification phases except…
  - for Sound: all aggregated nodes of signal in $u$ are: sensors in $u$ and its descendants reachable in Sound graph without fully dissipated

*How to find?*
World Interfacing

• Managing sense signals and sensors: **Finite Element Model Sense Manager**

  - similar aggregation, testing, notification phases except…
  - for Smell: at every frame, smell is being dissipated and propagated to all successors, and aggregated nodes with value > threshold