15-466
Computer Game Programming

Intelligence I:
Basic Decision-Making Mechanisms

Maxim Likhachev
Robotics Institute
Carnegie Mellon University
AI Architecture

from “Artificial Intelligence for Games” by I. Millington & J. Funge
Decision-making Framework

- **Internal knowledge**
  - e.g., health level, availability of ammunition, ...

- **External knowledge**
  - e.g., map, see enemy, hear sound, ...

- **Decision maker**

- **Action request**

- **Internal changes**

- **External changes**

from “Artificial Intelligence for Games” by I. Millington & J. Funge
Decision-making Framework

- e.g., health level, availability of ammunition, ...

- e.g., attack, flee, explore the sound, ...

- e.g., map, see enemy, hear sound, ...

- Any ideas for how to implement decision-making?

from “Artificial Intelligence for Games” by I. Millington & J. Funge
Basic Decision-making Mechanisms for this Class

- Decision Trees
- Finite-state Machines
- Basic Behavior Trees
Basic Decision-making Mechanisms for this Class

- Decision Trees
- Finite-state Machines
- Basic Behavior Trees
**Decision Trees**

- Formalization of a set of nested if-then rules
- Very popular: easy-to-implement, intuitive (=easy-to-debug) and fast
- Require careful manual design (theoretically, learning trees is also possible)

---

*from “Artificial Intelligence for Games” by I. Millington & J. Funge*
Decision Trees

- Formalization of a set of nested if-then rules
- Very popular: easy-to-implement, intuitive (=easy-to-debug) and fast
- Require careful manual design (theoretically, learning trees is also possible)

\[ \text{Is enemy visible?} \]
\[ \text{Is enemy audible?} \]
\[ \text{Is enemy <10 m away?} \]
\[ \text{Creep on flank?} \]
\[ \text{Attack} \]
\[ \text{Move} \]

*K-ary trees are possible but binary decision trees are much more common*

*From “Artificial Intelligence for Games” by I. Millington & J. Funge*
Decision Trees

• Support for multi-valued input variables in binary decision-trees

Example:

Depending on the size of the enemy troops, attack, stand ground or retreat

How to implement in a binary decision trees?
Decision Trees

- Support for multi-valued input variables in binary decision-trees

Example:

*Depending on the size of the enemy troops, attack, stand ground or retreat*

```
<table>
<thead>
<tr>
<th>N</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;2</td>
<td>Attack</td>
</tr>
<tr>
<td>&lt;5</td>
<td>Attack</td>
</tr>
<tr>
<td>≥5</td>
<td>Stand Ground</td>
</tr>
</tbody>
</table>
```
Decision Trees

• Support for continuous input variables in binary decision-trees

Example:

*Depending on the distance to the enemy,*

hand-to-hand combat, shoot, hide

How to implement in a binary decision trees?
Decision Trees

• Support for continuous input variables in binary decision-trees

Example:

*Depending on the distance to the enemy, hand-to-hand combat, shoot, hide*

```
Is d < 5.0m?
Yes

Is d < 20.0m?
Yes

Hand-to-hand combat

No

Hide

No

Shoot

d = distance to enemy
```
Decision Trees

• Support for complex decision formulae

*Example:*

*Attack whenever*

*enemy is close OR (low-on-health AND not too far)*

```
Is enemy close?
    Yes
    Attack
    No

Is enemy low-on-health?
    Yes
    Attack
    No

Do nothing

Is enemy too far?
    Yes
    Do nothing
    No
    Attack
```
Decision Trees

• Support for complex decision formulae

Example:

Attack whenever
enemy is close OR (low-on-health AND not too far)

Can we represent any Boolean formula with a binary decision tree?

Proof?

Is enemy close?

Yes

No

Is enemy low-on-health?

Yes

No

Do nothing

Is enemy too far?

Yes

No

Attack

Do nothing
Decision Trees

- Support for complex decision formulae

**Example:**

*Attack whenever*

*enemy is close OR (low-on-health AND not too far)*

\[
\text{A OR (B AND C)}
\]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>No</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Yes</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>No</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Yes</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Each row is a branch in the tree

True?

Proof?
Decision Trees

• Making the decision tree traversal fast is important

Which decision tree is better for decision-making?

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

Maxim Likhachev
Carnegie Mellon University
Decision Trees

• Making the decision tree traversal fast is important.

What does it depend on?

Which decision tree is better for decision-making?

from “Artificial Intelligence for Games” by I. Millington & J. Funge
Decision Trees

• Making the decision tree traversal fast is important.

Which decision tree is better for decision-making?

What does it depend on?

• Frequency (probability) of outcome (e.g., what if A happens 99% of the time)

• The computational complexity of the test (e.g., what if testing for G is very expensive)

from “Artificial Intelligence for Games” by I. Millington & J. Funge
Decision Trees

- Making the decision tree traversal fast is important

*Merging the branches:*

*from “Artificial Intelligence for Games” by I. Millington & J. Funge*
Decision Trees

• How to deal with the predictability of AI?

Any ideas?
Decision Trees

• How to deal with the predictability of AI?

*Random Decision Trees:*

To avoid switching every frame: use hysteresis (memory)

from “Artificial Intelligence for Games” by I. Millington & J. Funge
Basic Decision-making Mechanisms for this Class

- Decision Trees
- Finite-state Machines
- Basic Behavior Trees
Finite State Machines

• Basic FSM

![Finite State Machine Diagram]

Initial State

from “Artificial Intelligence for Games” by I. Millington & J. Funge
Finite State Machines

• Basic FSM

from “Artificial Intelligence for Games” by I. Millington & J. Funge
Finite State Machines

• Basic FSM

How to introduce unpredictability?

from “Artificial Intelligence for Games” by I. Millington & J. Funge
Finite State Machines

• Hierarchical FSM

How it changes to react to “re-charge now” alarm?

from “Artificial Intelligence for Games” by I. Millington & J. Funge
Finite State Machines

- Hierarchical FSM

from “Artificial Intelligence for Games” by I. Millington & J. Funge
Finite State Machines

• Hierarchical FSM

What if an additional alarm?

Upper bound on # of states for N alarms?

from “Artificial Intelligence for Games” by I. Millington & J. Funge
Finite State Machines

• Hierarchical FSM: strict hierarchy with only global alarms

  *State: \[\text{State at Level } i, \ldots, \text{State at Level 1}\] (for \(i\) active FSMs)*

  *All triggers get acted upon by FSM at level \(i\)*

  *Whenever FSM at Level \(i\) exits, FSM at level \(i-1\) becomes dominant*

---

from “Artificial Intelligence for Games” by I. Millington & J. Funge
Finite State Machines

- Hierarchical FSM: strict hierarchy with additional direct transitions

*Direct transitions between levels allow to leave the source state*

*from “Artificial Intelligence for Games” by I. Millington & J. Funge*
Finite State Machines

- Hierarchical FSM: strict hierarchy with additional direct transitions

Direct transitions between levels allow to leave the source state. When GetPower is done, Clean Up is entered from scratch.
Finite State Machines

- Hierarchical FSM: strict hierarchy with additional direct transitions

More complex example:

![Diagram of a finite state machine with states and transitions]

from “Artificial Intelligence for Games” by I. Millington & J. Funge
Finite State Machines

• Combining FSM and decision trees

from “Artificial Intelligence for Games” by I. Millington & J. Funge
Basic Decision-making Mechanisms for this Class

• Decision Trees

• Finite-state Machines

• Basic Behavior Trees
Basic Behavior Trees

• Became very popular after Halo 2 game [2004]
Basic Behavior Trees

- Became very popular after Halo 2 game [2004]
- Especially, when coupled with graphical interfaces to edit them

Screenshot of GameBrains GUI
Basic Behavior Trees

• Collection of simple tasks arranged in a tree structure
• One of the main advantages: Tasks and sub-trees can be reused!!!
Basic Behavior Trees

• Type of tasks: *Conditions, Actions and Composites*
• Each returning either success or failure

“*Condition*” task tests for a condition

*Examples of behavior trees that consist of Conditions tasks only:*

<table>
<thead>
<tr>
<th>Door open?</th>
<th>Health level OK?</th>
<th>Enemy close-by?</th>
<th>...</th>
</tr>
</thead>
</table>
Basic Behavior Trees

- Type of tasks: *Conditions, Actions and Composites*
- Each returning either success or failure

“*Action*” task alters the state of the game

Examples of behavior trees that consist of Actions tasks only:

- Move to room
- Find a path
- Play audio sound
- Talk to the player
- ...

Basic Behavior Trees

• Type of tasks: *Conditions, Actions and Composites*
• Each returning either success or failure

*Condition and Action tasks are always at the leafs of the tree “Composite” task sequences through them*
Basic Behavior Trees

• Type of tasks: *Conditions, Actions and Composites*
• Each returning either success or failure

*Condition and Action tasks are always at the leafs of the tree
“Composite” task sequences through them*

*Two types of Composite tasks:*

*Selector returns as soon as
the first leaf task is successful*
Basic Behavior Trees

- Type of tasks: Conditions, Actions and Composites
- Each returning either success or failure

Condition and Action tasks are always at the leafs of the tree “Composite” task sequences through them

Two types of Composite tasks:

**Selector** returns as soon as the first leaf task is successful

**Sequencer** returns as soon as the first leaf task fails

- Attack
- Flee
- Declare defeat

- Door open?
- Move (into room)
Basic Behavior Trees

• Type of tasks: *Conditions, Actions and Composites*
• Each returning either success or failure

*Condition and Action tasks are always at the leafs of the tree*

*Composite* *task sequences through them*

*Two types of Composite tasks:*

*Selector* returns as soon as the first leaf task is successful

*Sequencer* returns as soon as the first leaf task fails

Tasks are often parameterized

- Attack
- Flee
- Declare defeat

- Door open?
- Move (into room)
Basic Behavior Trees

• Type of tasks: *Conditions, Actions and Composites*

• Each returning either success or failure

*Example:*

- **Door open?**
- **Move (into room)**
- **Move (to door)**
- **Open door**
- **Move (into room)**

What does it do?
Basic Behavior Trees

- Type of tasks: *Conditions, Actions and Composites*
- Each returning either success or failure

**Example:**

1. **Door open?**
2. **Move (into room)**
3. **Move (to door)**
4. **Door unlocked?**
5. **Open door**
6. **Door locked?**
7. **Barge door**

What does it do?
Basic Behavior Trees

• Type of tasks: *Conditions, Actions and Composites*
• Each returning either success or failure

*Example:*

```
Door open?    Move (into room)    Move (to door)
  └──┬─┐       └──┬─┐            └─┬─┘
    │         │                │
Door unlocked? Open door Door locked? Barge door
```

*Can the tree always be structured so that sequence and selector levels alternate?*
Basic Behavior Trees

• Type of tasks: *Conditions, Actions and Composites*
• Each returning either success or failure

Example:

How to reduce predictability of the behavior?
Basic Behavior Trees

- Behavior trees with **Order Randomization for some Sequencers and Selectors**

**Example:**

![Diagram](image)

*How to reduce predictability of the behavior?*

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