Credits

• Heavily based on
Techniques to Go Through

• Decision Tree
• Finite State Machine
• Behaviour Tree
• Planning
• Steering Behaviour
• Pathfinding (1,2)
Outline for today

• Decision tree
A Very Rough Structure of Game AI

In reality, there is no clear cut.
Major Approaches

• Reactive AI
  – Computer player reacts to human player actions
    • Event-driven
    • Pull-based

• Goal-driven AI
  – Pursuing goals
    • Hierarchy of goals

• Combinations and variations
Decision Trees and Rule-Based Systems

• Many game situations can be described as  *if-then-else* cases
  
  – If see enemy *then* shoot
  – If (animal is enemy *or* neutral) *and* animal is not healthy *then* eat it
  – If animal sings and dances *then* it’s friendly

• Decision trees
• Rule-based (production/expert) systems
Decision Trees

- Simplest decision making technique
- Easy to implement and understand
- Mostly reactive AI
- Fast execution
- Can be combined with other techniques
- Can be *learned* (using machine learning techniques)
Example

Soldier decision making
• Based on perception

Attribute test

Action / Decision / Classification
Logical Connectives

• **A and B**

1. **A**
   - Yes
   - No

2. **B**
   - Yes
   - No

3. **A or B**

   1. **A**
      - Yes
      - No

   2. **B**
      - Yes
      - No

   3. 1
   4. 2
Easy to Implement

```java
if (enemy.isVisible()) {
    if (distance(player, enemy) < 10) {
        attack();
    }
    else {
        if (enemy.isOnFlank()) {
            move();
        } else {
            attack();
        }
    }
} else {
    ... ... ...
```

Hard-coded knowledge may not be a good idea
Why: Maintainability

• Why hard-coded AI is not a good idea?
  – Maintainability
    • Add an extra check “is enemy a tank?”

```java
if (enemy.isVisible()) {
  if (distance(player, enemy) < 10) {
    attack();
  } else {
    if (enemy.isOnFlank()) {
      move();
    } else {
      attack();
    }
  }
} else {
  ... ...
```

Which one would you choose to update?
Why: Tree Balancing

- The longer the branch the longer it takes to go along it

```java
if(…) {
    A;
} else if(…){
    B;
} else if (…){
    C;
} else if(…) {
    D;
    ...
    ...
    ...
```
Manageable Implementation

• Special languages
  – Overkill
  – Can be done with AI scripting approaches

• A library of (C++ / Java) classes for attributes, tests and actions
  – Somewhat similar to scene graph libraries
Extensions: Split on Other Values

- Yes/No is not an answer
  - Decide on other attributes. For example,

Possible data types:
- Boolean
- Enumeration
- 3D Vector (vector length within range, vector direction is given,...)
- ...
Variations: Random Decisions

• Completely predictable behaviour is boring

• Randomness breaks the pattern

• Coin can be biased (player psychology)
Sticking to Choice

• Marine behaviour

• Sense – Think – Act cycle navigates the decision tree every time

• Random choice every iteration will make the marine freeze
Learning Decision Trees

• Aims:
  – Better gameplay
  – Cheaper AI
  – Adaptive AI

• Not often used by game developers
  – Reproducibility and quality control
  – Increased run time
  – Can be faked
Alternatives to ML

• Pre-programmed levels of difficulty
  – Switch between behaviours

• Incremental introduction of new game entities
  – “Uncover” cleverness of AI

• Tweaking parameters at run-time
  – Reduce the number of mistakes
  – Improve aim
  – Limited form of machine learning (stats)
    • Learning user’s habits (attack from right etc.)
Faking vs Learning

- Learning (potentially) gives more options but
- With faking the AI code remains unchanged and can be tested debugged
- On the other hand, learning gives stunning results in traditional AI (not game AI).
When to Learn

• Online learning
  – While playing
  – Input from players
  – Aim: adaptive behaviour

• Offline learning
  – Before the product is released
  – Input from designers
  – Aim: finding best behaviours
Basic Techniques

• Analysing examples
  – About 75% are used to learn
  – The rest (25%) are used to test

• Reinforcement learning
  – Rewards and punishments for actions
Decision Trees from Examples

- Given: Attributes, Decisions, Examples
- Required: Construct a tree

Example: marine behaviour

<table>
<thead>
<tr>
<th>Health</th>
<th>Cover</th>
<th>Ammo</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Healthy</td>
<td>In Cover</td>
<td>With Ammo</td>
<td>Attack</td>
</tr>
<tr>
<td>Hurt</td>
<td>In Cover</td>
<td>With Ammo</td>
<td>Attack</td>
</tr>
<tr>
<td>Healthy</td>
<td>In Cover</td>
<td>Empty</td>
<td>Defend</td>
</tr>
<tr>
<td>Hurt</td>
<td>In Cover</td>
<td>Empty</td>
<td>Defend</td>
</tr>
<tr>
<td>Hurt</td>
<td>Exposed</td>
<td>With Ammo</td>
<td>Defend</td>
</tr>
</tbody>
</table>
Decision Tree Learning Algorithm

function \text{DTL}(\text{examples}, \text{attributes}, \text{default}) \text{ returns a decision tree}

\begin{align*}
\text{if } & \text{examples is empty then return default } \\
\text{else if } & \text{all examples have the same classification then return the classification } \\
\text{else if } & \text{attributes is empty then return Mode(examples) } \\
\text{else } & \\
& \text{best } \leftarrow \text{Choose-Attribute(attributes, examples)} \\
& \text{tree } \leftarrow \text{a new decision tree with root test best} \\
& \text{for each value } v_i \text{ of best do} \\
& \text{examples}_i \leftarrow \{ \text{elements of examples with best } = v_i \} \\
& \text{subtree } \leftarrow \text{DTL(examples}_i, \text{attributes}\setminus \text{best}, \text{Mode(examples))} \\
& \text{add a branch to tree with label } v_i \text{ and subtree subtree} \\
& \text{return tree}
\end{align*}

From S. Russel, P. Norvig “Artificial Intelligence: A modern approach”, Prentice Hall
Example

Attributes order: the column (random) order
### Different Order of Attributes

Attributes order: Ammo, Cover

#### Decision Tree

- **inCover?**
  - yes
  - no

- **withAmmo?**
  - yes
  - no

- **withAmmo?**
  - yes
  - no

#### Decision Table

<table>
<thead>
<tr>
<th>Health</th>
<th>Cover</th>
<th>Ammo</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Healthy</td>
<td>In Cover</td>
<td>With Ammo</td>
<td>Attack</td>
</tr>
<tr>
<td>Hurt</td>
<td>In Cover</td>
<td>With Ammo</td>
<td>Attack</td>
</tr>
<tr>
<td>Healthy</td>
<td>In Cover</td>
<td>Empty</td>
<td>Defend</td>
</tr>
<tr>
<td>Hurt</td>
<td>In Cover</td>
<td>Empty</td>
<td>Defend</td>
</tr>
<tr>
<td>Hurt</td>
<td>Exposed</td>
<td>With Ammo</td>
<td>Defend</td>
</tr>
</tbody>
</table>
Two Learnt Trees

Attributes order: the column (random) order

Attributes order: Ammo, Cover

Health does not matter!
The Order of Attributes Matters

• Pick one best splits the cases
• Bad choice may lead to overfitting: decision tree can handle given examples but not generalise from them

• First, split on the attribute that give biggest Information Gain
  – Information theory (Shannon, Weaver, 1949)
  – Numerical value of attribute based on statistics
Information Entropy

For a set of examples $S$ let

• $n_p$ be the number of examples with a *positive* outcome (e.g. Attack)
• $n_n$ be the number of examples with a *negative* outcome (e.g. Defend)

Then, the entropy (a measure of uncertainty) for this set is

$$E_S = \frac{n_p}{n_p + n_n} \log_2 \frac{n_p}{n_p + n_n} + \frac{n_n}{n_p + n_n} \log_2 \frac{n_n}{n_p + n_n}$$
Information Gain

Every attribute $A$ splits the set of examples $S$ into two subsets

- $S_A$, for which the value of $A$ is true
  - Compute the entropy $E_{S_A}$ for $S_A$
- $S_{\sim A}$, for which the value of $A$ is false
  - Compute the entropy $E_{S_{\sim A}}$ for $S_{\sim A}$

$$G_A = E_S \left( \frac{|S_A|}{|S|} E_{S_A} \right) \left( \frac{|S_{\sim A}|}{|S|} E_{S_{\sim A}} \right)$$
ID3

Pick the attribute with the highest information gain

\[ G_{\text{health}} = 0.02 \]
\[ G_{\text{cover}} = 0.171 \]
\[ G_{\text{ammo}} = 0.420 \]

<table>
<thead>
<tr>
<th>Health</th>
<th>Cover</th>
<th>Ammo</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Healthy</td>
<td>In Cover</td>
<td>With Ammo</td>
<td>Attack</td>
</tr>
<tr>
<td>Hurt</td>
<td>In Cover</td>
<td>With Ammo</td>
<td>Attack</td>
</tr>
<tr>
<td>Healthy</td>
<td>In Cover</td>
<td>Empty</td>
<td>Defend</td>
</tr>
<tr>
<td>Hurt</td>
<td>In Cover</td>
<td>Empty</td>
<td>Defend</td>
</tr>
<tr>
<td>Hurt</td>
<td>Exposed</td>
<td>With Ammo</td>
<td>Defend</td>
</tr>
</tbody>
</table>

Best choice
Learning with ID3

Attributes order: Ammo, Cover

Best outcome
Dealing with Noise

• Data often contains “noise”
  – E.g., human player decides to attack regardless of not having any ammo

• The learnt decision tree will take *irrelevant* attributes into account
  – E.g. in our example, Health was irrelevant

• **Pruning** techniques: eliminate splitting on statistically insignificant attributes
Black & White

• Most prominent example where decision trees were learnt is Black & White.
  – Creature can be trained by users
    • If the creature behaviour is “bad”, hard to retrain
  – Very positive initial reception
    • Some critics reconsidered their opinion
Decision Trees: Summary

• Advantages:
  – Simple, compact representation
  – Easy to create and understand
  – Decision trees can be learned

• Disadvantages:
  – Slightly more coding than other techniques (FSMs)
  – Learnt trees may contain errors
Expert (Rule-Based) Systems in Games

• Rule-based knowledge representation
  – Set of rules
  – Facts in working memory
  – Inference engine
  • Conflict resolution
Chaining

• Forward chaining
  – Game actions cause changes to the working memory
  – AI agent acts on the derived knowledge
  – Reactive AI

• Backward chaining
  – Pursue goals
  – Goal-driven AI
    • Other methods are more common
Example: Age of Kings

(defrule (unit-type-count villager > 0) (chat-to-all "I just made my first rule!") (disable-self); )

http://aok.heavengames.com/cgi-bin/aokcgi/display.cgi?action=ct&f=26,29,,30
Larger Rule

(defrule (building-type-count-total-total)

  (house > 0) (building-type-count-total mill == 0) (resource-found food) (can-build mill) => (build mill)

Rules are commonly used in strategy game AI