What is Artificial Intelligence?
- “The study of computational systems that exhibit intelligence.”
- Theories and computational models.

What is Intelligence?
“It’s whatever people do.”
- Leads to the “Turing Test”: If a person can’t distinguish its behavior from a human, then it is intelligent.
- This is not rigorous—“I know intelligence when I see it”.

Behaving in a rational manner:
- Use available knowledge to maximize goal achievement.
- Often leads to optimization techniques.

Demonstrating complex capabilities:
- Problem solving
- Learning
- Planning,...
Roles of AI in Computer Games

Roles of Game AI: Complex behaviors not specified by player.

- Nonplayer Opponents: Realistic attack behavior.
- Nonplayer Teammates: Coordinated supportive behavior.
- Support and Autonomous Characters: Crowd/flock behavior.
- Commentators/Instruction: Does the player need a hint?
- Camera Control: Finding the best viewpoint.

What AI is not: There is not a clear distinction, but...

- Determined by physical laws: E.g., path of a tennis ball.
- Purely random: E.g., which block falls next in Tetris.
- Direct response to game rules/user inputs: E.g., shoot gun when space-bar hit, scripted instruction sequences.

AI versus Animation

AI System: AI determines what to do and the animation does it.

AI drives animation: AI system decides what action to take.
Animation renders the result.

Scenario 1: AI system issues orders: "move from A to B."
Animation system does the rest.

Scenario 2: AI system controls everything, down to the animation clip to play.

Which scenario?
- Depends on the nature of the AI system and the nature of the animation system.
- Is the animation system based on motion capture, or physics, or something else?
- Does the AI perform collision avoidance? Does it do detailed planning?
Properties of a Good AI System

Goals:

- **Goal driven**: AI system decides what to do, and then figures out how to do it.
- **Responsive**: AI system responds immediately to changes in the world.
- **Smart**: (knowledge intensive) AI system knows a lot about the world and how it behaves, and uses this knowledge in its own behavior.
- **Consistent**: Embodies a believable, consistent character.
- **Efficient**: Low CPU and memory usage.
- **Practical**: Fast and easy development.

Tradeoffs:
- Unfortunately, many of these goals conflict fundamentally.

Overview

- **AI Basics**
  - Agents
  - Rule-based Approaches
  - Finite-State Machines
**General Structure: AI Update Step**

**Sensing:**
- Determines the **state** of the world.
- **May be simple:** State changes all come by message.
- **Or complex:** Figure out what is visible, where your team is, etc.

**Thinking:**
- What to do **next**, given world state/goals.
- The **core** of the AI system.

**Acting:**
- Tells the animation system what to do.
- Translate low-level goals into **specific motion** (velocities, joint-angles).
- Can be messy/tedious.

---

**General Structure: Polling or Event Driven?**

**Polling:**
- The AI gets called at **fixed time** intervals.
- **Senses:** Look to see what has **changed** in the world. For instance:
  - Queries what it can see.
  - Acts on it.

**Event Driven:**
- Acts in response to **world events**.
  - Events sent by **message**. (E.g. invoke a callback function.)
- **Examples:**
  - You heard a sound.
  - A threat just entered your field of view.
  - You have just been hit.

**Real systems are a mix:**
- Something just changed (event)...so do some sensing (polling).
AI Techniques in Games

Basic problem:
- Given the state of the world, what should the agent do next?

Range of (real-time) solutions in games:
- Rule-based systems
- Finite-state machines
- Decision trees
- Neural networks
- Fuzzy logic

Wider range of solutions in the academic world:
- Planning systems
- Logic programming
- Genetic algorithms
- Bayes nets.
- Currently, too slow for games (but this may change in time).

Two Measures of Complexity

Execution Complexity:
- How fast does it run as more knowledge is added?
- How much memory is required as more knowledge is added?
- Determines the run-time cost of the AI.

Specification Complexity:
- How hard is it to write the code?
- As more knowledge is added, how much more code needs to be added?
- Determines the development cost, and risk.
Overview

- AI Basics
- **Agents**
- Rule-based Approaches
- Finite-State Machines

Agents: Goals of an Opponent

Provide a challenging (but flawed) opponent:
- Should be **beatable** (but not too easily).
- Should be **entertaining** and fun.

Not too challenging:
- Should not be **superhuman** in accuracy, precision, sensing, ...

No unintended weaknesses:
- No "golden path" to defeating opponent every time.
- Must not fail miserably or look **dumb** (e.g., getting lost).

Not be too predictable:
- Through randomness.
- Through **multiple**, fine-grained **responses**.
- Through **adaptation** and learning.
Agents: What are They?

Definition:
"An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future." (Stan Franklin and Art Graesser)

Structure of Agent Action:
- Sensing: perceive features of the environment.
- Thinking: decide what action to take to achieve its goals, given the current situation and its knowledge.
- Acting: doing things in the world.

Issues:
- Thinking can make up for limitations in sensing and acting (e.g., extrapolating future motion).
- The more accurate the models of sensing and acting, the more realistic the behavior.

Agents: Sensing Limitations & Complexities

Sensing Issues:
- Limited sensor distance/field of view: Can sense local environment.
- Obstacles: Agent cannot see through walls.
- Improving the view: Detecting and computing paths to doors.
- Realistic behavior: Sensitivity to motion, fooled by camouflage.
- Reaction time: Sensed information is not processed instantly.

Random noise in sensors:
- Mimicking human limitations.

Different Sensors:
- Sound: Omni-directional, gives direction, distances, speech, ...
- Vision: Limited field of view, 2.5D, color, texture, motion, ...
- Smell: Omni-directional, chemical makeup.

Integration:
- Need to integrate different sources to build complete picture.
**Agents: Simple Action Strategies**

**Random motion:** ("unintelligent" AI)
- Roll the dice to select when and where to move.

**Regular pattern:**
- Follow invisible tracks: E.g., Galaxian.

**Tracking/Pursuit Strategies:**
- **Direct Pursuit:** Move toward prey’s position.
  (E.g., heat-seeking missile)
- **Lead Pursuit:** Move toward a spot ahead of prey.
- **Interception:** Move to where prey is expected to be.

**Evasion Strategies:**
- **Direct Evasion:** Move directly away from (slow) pursuer.
- **Side-step:** Head perpendicular to (fast) pursuer’s current bearing.
- **Weave:** Every N seconds move X degree off pursuer’s bearing.

---

**Overview**

- AI Basics
- Agents
- **Rule-based Approaches**
- Finite-State Machines
**Rule-based approaches**

**Rule-Based Systems:** Popular because:
- Familiar.
- Predictable and, hence, testable.
- Many game designers don’t know much about AI. :-)

**Rules specify the action depending on circumstances:**
- **Deterministic:** One action for each situation.
- **Random:** Multiple possible actions in the same situation.
- **Weighted:** Certain actions have a higher probability.
- **Based on State/Environment Information:** Take the direction that leads you to the opponent.

**Example:**
- A simple rule-based system for moving a ghost agent in Pac-Man.

<table>
<thead>
<tr>
<th>State</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>Go ahead</td>
</tr>
<tr>
<td>Blocked</td>
<td>Turn Right</td>
</tr>
<tr>
<td>Blocked, Open</td>
<td>Turn Left</td>
</tr>
<tr>
<td>Blocked, Blocked, Blocked</td>
<td>Go backwards</td>
</tr>
</tbody>
</table>

**Note:**
- Easy to add randomness, if we like.
- Does not consider the location/state of the player.
Overview

- AI Basics
- Agents
- Rule-based Approaches
- **Finite-State Machines**

Finite-state machines (FSMs)

**Finite-State Machines:**
- A (finite) set of possible states that the agent can be in.
- Connected by transitions that are triggered by events or changes in the world or user input.
- Normally represented as a directed graph, with the edges labeled with the transition event.

**Ubiquitous:**
- Almost all computer game AI systems support FSMs.

**Déjà vu?**
- FSMs are used in language processing (regular languages).
- You might have seen them in formal language theory (CMSC 452).

**Issues:**
- Provide a clean graphical way to describe state transitions/actions.
- Uniform table/graph structure.
Quake-Bot Example

Types of behavior to capture:
- If you don’t see an enemy, wander randomly.
- When see enemy, attack.
- When hear an enemy, chase enemy.
- On dying, re-spawn.
- Extras: If health is low and seeing an enemy, retreat.

Finite-state machines (FSMs)

Actions: On entering or exiting a state we invoke a callback function.

Problem: There is no transition from Attack to Chase.
Finite-state machines (FSMs)

Possible Fix: Create an extra attack state if sound is heard.

Problem: This can result in the proliferation of a huge number of states.

Solution: Rather than creating new states, make limited use of additional variables.

Events:
E=Enemy Seen
S=Sound Heard
D=Die

Wander
- E, - S, - D

Attack
E, - D

Attack-S
E, - D, S

Chase
S, E, - D

Spawn
D

Hierarchical FSMs

Expand complex states into their own FSM.

Use a stack to save current state.

Wander
E / E

Pick-up
Powerup

Start

Chase
S / S

Die

Spawn
Nondeterministic FSMs

Add multiple transitions and associated probabilities. These are essentially Markov chains.

Efficient FSM Implementation

**Basic Implementation:**
- Compile into an array indexed by [state-name, event].
- Transition: state-name_{i+1} ← array[state-name_i, event]
- Switch based on state-name to call execution logic.

**Hierarchical FSMs:**
- Create array for every FSM.
- Have stack of states:
  - Classify events according to stack.
  - Update state which is sensitive to current event.

**Nondeterministic FSMs:**
- Have array of possible transitions for every (state-name, event) pair.
- Choose one at random, based on transition probabilities.
Summary

Summary:
- AI Basics
- Agents
- Rule-based Approaches
- Finite-State Machines