Perception, Learning, Planning and Near-Future Game AI

Chris Darken

Seoul International Games Symposium
May 11, 2006
MOVES Institute

MOVES = MOdeling, Virtual Environments, and Simulation
Housed at the Naval Postgraduate School
Most students are military officers (from US and 30 other countries)
Defense-oriented research and education
America's Army

- A polished full scale game developed at a school!
- Used Epic's Unreal game engine
- Purpose was Army marketing
- Distributed freely
- At peak, the third most popular US online game
Delta3D Game Engine

- Open Source
- Built in-house at MOVES
- www.delta3d.org
- Used in house to build training and analysis games
- AI facility under construction!
Outline

• Game AI as we see it, and its problems
• How we want to address those problems
  – More and better AI *perception* of who and what are near it
  – Autonomous *learning* about the game world
  – Selecting actions by *planning*
Game AI is all about the actions taken by the character. Are they consistent with the role of the character, or jarringly inconsistent?
Where do we look for problems?

Not just here...

perception

But also here!
Game AI is Nearly Blind
Half-Life 2 Example
Lines of Sight (LOS)
LOS Problems

This target is invisible to LOS.

This target, however, is obvious!
Multiple-Trace LOS

- Tracing lines of sight to multiple locations on (or near) the target is a partial solution.
Mini-Renders (Pursel thesis 2004)

- A mini-render is a low-resolution render is done from the AI character's point of view
- Mini-renders are not for display; they are for the AI only
- A second, false-color render determines which pixels are the figure and which the ground
- This allows GPU computation of
  - The visible area of a target
  - The visible fraction of a target
Problems with Visible Area

• Visible area computation can solve both LOS errors
• But it does not solve all problems
• Consider targets in
  – Camouflage
  – Smoke
  – Deep shadow
• We are investigating adaptations of the ACQUIRE target detection model for these cases
The ACQUIRE Family

Tight View → Segmentation Mask → Brightness → Texture

Correia thesis, 2005
Experimental Testbench

We are currently performing experiments to tune and expand ACQUIRE to more closely match human performance in virtual environments.
Knowledge of Level Geometry: Ground Truth

- Car
- House
- Rock
The Level Designer Adds Waypoints ("Path Nodes")*

*This description is specific to Unreal-style engines. An alternative approach is to use “navigation meshes”.
Game AI Perception: Waypoints Only
Waypoint Navigation

• Game AI traditionally navigates from point A to B using “waypoints”

• Find the nearest waypoint that can see A (call it WA)

• Likewise WB is the nearest seeing B

• Find the shortest path from WA to WB on the waypoint graph using A* search

• Move!
Example: Navigation

The light green half-circles are visible to the green character.
Example: Navigation
Waypoint-Based Hiding

- How can waypoints be used to enable the game AI to hide (take cover)?
- Typically, waypoints that might be good hiding places are manually tagged by the level designer.
- When the AI wants to hide, the nearest tagged waypoints are checked against the current enemy positions.
- The AI can now move to the best waypoint.
Example: Hiding

The light red half-circles are visible to the red character.
Example: Hiding
Dense Waypoint Graphs

- The current trend is to use very many waypoints to ensure that the game AI has good nearby choices.
- Placing the waypoints is requires a lot of labor by the level designer.
- For this reason, we are investigating automatic waypoint placement by “level exploration.”
Level Exploration

- For each waypoint, and each possible movement
  - Try movement
  - Check distance to closest stored point
  - If above threshold, add to the list of waypoints
Advantages

• Automatic level annotation
  – For Unreal-style engines, placing and annotating waypoints is the level designer's job

• Consistency and flexibility
  – Can use the actual movement and collision models that will be used in the game, even physics-based movement

• Automatic level testing
  – Can find and tag places where player becomes stuck or falls off the level
Alternative: The Sensor Grid Algorithm for Cover Finding

- A run-time approach to finding hiding places
- Designed to work even with dynamic environments
- Documented in:
Action Selection

• How does the game AI decide what to do?

• The most important approaches are
  – Using a scripting language (rather than C) to speed development
  – Using a Finite State Machine (FSM) to organize the code

• Problem: FSM's get large and confusing!

http://ai-depot.com/FiniteStateMachines/FSM-Framework.html
Planning

A declarative, rather than procedural, approach to constructing game AI: a runtime search process is involved

Used in FEAR, a successful AAA title that has received much positive comment on its AI (Orkin, 2006)

Our architecture is based on FEAR's
Operators

Plans are comprised of operators, which contain:

- **Preconditions**: must be satisfied before the operator can be applied
- **Effects**: The results of using the operator
- **Interrupts**: Failure tests that trigger replanning (not used at right)
Simple Planning Example

Initial State

Goal State
Simple Planning Example

Initial State

Op 1

Op 4

Op 3

Goal State
Cleared Hot Helicopter Operators

**SetFlyToWaypoint**
- **Precond:** HaveWaypoint
- **Interrupt:**
- **Effect:** SetWaypoint

**FlyToWaypoint**
- **Precond:** SetWaypoint
- **Interrupt:** xShouldStop
- **Effect:** AtWaypoint

**Hover**
- **Precond:**
- **Interrupt:** HaveWaypoint
- **Effect:**
The operators interact with a small FSM (Finite State Machine)
Planner Details

Planner searches states via forward A*
Can handle variables
Planning is computation intensive (intractable), so plans are typically cached
Not fully integrated with game engine yet, so only cached plans supported at present
Plan monitor supports “interrupts” that force replanning
There is plenty of “glue code” around the planner and state machine
Learned Anticipation

An *autonomously-acquired* ability to *recognize* a situation and to *predict* what is likely happen next
Summary

Built a MUD (networked combat-oriented game) discrete-event simulation
Built first-ever prototype agents that learn completely via a simulation event stream


Perceps

2.34 + location Bob meadow

(A: agent action,
+: begin sensing,
-: end sensing,
E: all other events)
Percept History

[11.3870000839, 'A', 'look', 'spock85']
[11.3870000839, '+', 'location', 'pitchfork74', 'Paperville3']
[11.3870000839, '+', 'pitchfork', 'pitchfork74']
[11.3870000839, '+', 'location', 'wand75', 'Paperville3']
[11.3870000839, '+', 'wand', 'wand75']
[11.3870000839, '+', 'location', 'spock85', 'Paperville3']
[11.3870000839, '+', 'spock', 'spock85']
[29.1619999409, 'A', 'get', 'spock85', 'spock85']
[29.1619999409, 'E', 'missing', 'spock85']
[39.5670001507, 'A', 'health', 'spock85']
[39.5670001507, 'E', 'feisty', 'spock85']
[50.4430000782, 'A', 'e', 'spock85']
[50.4430000782, 'E', 'go', 'spock85', 'east']
[50.4430000782, '+', 'location', 'pitchfork74', 'Paperville3']
[50.4430000782, '+', 'pitchfork', 'pitchfork74']
[50.4430000782, '+', 'location', 'wand75', 'Paperville3']
[50.4430000782, '+', 'wand', 'wand75']
[50.4430000782, '+', 'location', 'spock85', 'Paperville3']
[50.4430000782, '+', 'spock', 'spock85']
[50.4430000782, '+', 'location', 'spock85', 'The_Western_Waste40']
[50.4430000782, '+', 'spock', 'spock85']
[80.0460000038, '+', 'location', 'Troll82', 'The_Western_Waste40']
[80.0460000038, '+', 'Troll', 'Troll82']
[90.0499999523, 'E', 'miss', 'Troll82', 'spock85']
[91.0510001183, 'E', 'miss', 'spock85', 'Troll82']
[93.0540001392, 'E', 'miss', 'Troll82', 'spock85']
[94.055999943', 'E', 'miss', 'spock85', 'Troll82']
[94.055999943', 'A', 'unequip', 'spock85', 'spock85']
[94.055999943', 'E', 'missing', 'spock85']
[96.0590000153, 'E', 'hit', 'Troll82', 'spock85']
[96.0590000153, 'E', 'feisty', 'spock85']
[97.059999428, 'E', 'hit', 'spock85', 'Troll82']
[97.059999428, 'E', 'feisty', 'Troll82']
[99.0629999638, 'E', 'hit', 'Troll82', 'spock85']
[99.0629999638, 'E', 'tired', 'spock85']
[100.06400013', 'E', 'miss', 'spock85', 'Troll82']
[102.067000151, 'E', 'hit', 'Troll82', 'spock85']
[102.067000151, 'E', 'wounded', 'spock85']
[103.069000006, 'E', 'hit', 'spock85', 'Troll82']
[103.069000006, 'E', 'feisty', 'Troll82']
[105.072000027', 'E', 'hit', 'Troll82', 'spock85']
[105.072000027', 'E', 'faltering', 'spock85']
[106.072999954', 'E', 'hit', 'spock85', 'Troll82']
One “Situation”

[11.3870000839, 'A', 'look', 'spock85']
[11.3870000839, 'A', 'location', 'pitchfork74', 'Paperville3']
[11.3870000839, 'A', 'location', 'wand75', 'Paperville3']
[11.3870000839, 'A', 'location', 'spock85', 'Paperville3']
[11.3870000839, 'E', 'get', 'spock85', 'spock85']
[29.1619999409, 'A', 'missing', 'spock85']
[39.5670001507, 'A', 'health', 'spock85']
[39.5670001507, 'E', 'feisty', 'spock85']
[50.4430000782, 'A', 'e', 'spock85']
[50.4430000782, 'A', 'go', 'spock85', 'east']
[50.4430000782, 'E', 'location', 'pitchfork74', 'Paperville3']
[50.4430000782, 'E', 'location', 'wand75', 'Paperville3']
[50.4430000782, 'E', 'location', 'spock85', 'The_Western_Waste40']
[80.046000038, 'A', 'location', 'Troll82', 'The_Western_Waste40']
[90.0499999523, 'E', 'miss', 'Troll82', 'spock85']
[91.0510001183, 'E', 'miss', 'spock85', 'Troll82']
[93.0540001392, 'E', 'miss', 'Troll82', 'spock85']
[94.055999943, 'E', 'miss', 'spock85', 'Troll82']
[94.055999943, 'A', 'unequip', 'spock85', 'spock85']
[94.055999943, 'E', 'missing', 'spock85']
[96.0590000153, 'E', 'hit', 'Troll82', 'spock85']
[96.0590000153, 'E', 'feisty', 'spock85']
[97.0599999428, 'E', 'hit', 'spock85', 'Troll82']
[97.0599999428, 'E', 'feisty', 'Troll82']
[99.0629999638, 'E', 'hit', 'Troll82', 'spock85']
[99.0629999638, 'E', 'tired', 'spock85']
[100.06400013, 'E', 'miss', 'spock85', 'Troll82']
[102.067000151, 'E', 'hit', 'Troll82', 'spock85']
[102.067000151, 'E', 'wounded', 'spock85']
[103.069000006, 'E', 'hit', 'spock85', 'Troll82']
[103.069000006, 'E', 'feisty', 'Troll82']
[105.072000027, 'E', 'hit', 'Troll82', 'spock85']
[105.072000027, 'E', 'faltering', 'spock85']
[106.072999954, 'E', 'hit', 'spock85', 'Troll82']
Example Situation

Combat Events

Health Status Messages

[['102.067000151', 'E', 'wounded', 'spock85']
[['103.069000006', 'E', 'hit', 'spock85', 'Troll82']
[['103.069000006', 'E', 'feisty', 'Troll82']
[['105.072000027', 'E', 'hit', 'Troll82', 'spock85']
[['105.072000027', 'E', 'faltering', 'spock85']
[['106.072999954', 'E', 'hit', 'spock85', 'Troll82']

THE MOVES INSTITUTE
NAVAL POSTGRADUATE SCHOOL
<table>
<thead>
<tr>
<th>X1</th>
<th>Action</th>
<th>X2</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3870</td>
<td>'A', 'look', 'spock85'</td>
<td>50.4430000782</td>
<td>'E', 'go', 'spock85', 'east'</td>
</tr>
<tr>
<td>11.3870</td>
<td>'A', 'location', 'pitchfork74', 'Paperville3'</td>
<td>50.4430000782</td>
<td>' ', 'location', 'pitchfork74', 'Paperville3'</td>
</tr>
<tr>
<td>11.3870</td>
<td>'A', 'location', 'wand75', 'Paperville3'</td>
<td>50.4430000782</td>
<td>' ', 'location', 'wand75', 'Paperville3'</td>
</tr>
<tr>
<td>11.3870</td>
<td>'A', 'spock', 'spock85'</td>
<td>50.4430000782</td>
<td>' ', 'spock', 'spock85'</td>
</tr>
</tbody>
</table>

The moves include looking around, checking the location, walking towards the pitchfork, and continuing towards the wand. The player eventually approaches the location, equips the spock, and moves towards the east. This sequence of moves is consistent with the expected behavior of the game, as the player navigates through the environment and interacts with the objects present.
History -> Overlapping Situations

[11.3870000839, 'A', 'look', 'spock85']
[11.3870000839, '+', 'location', 'pitchfork74', 'Paperville3']
[11.3870000839, '+', 'pitchfork', 'pitchfork74']
[11.3870000839, '+', 'location', 'wand75', 'Paperville3']
[11.3870000839, '+', 'wand', 'wand75']
[11.3870000839, '+', 'location', 'spock85', 'Paperville3']
[11.3870000839, '+', 'spock', 'spock85']
[29.1619999409, 'A', 'get', 'spock85', 'spock85']
[29.1619999409, 'E', 'missing', 'spock85']
[39.5670001507, 'A', 'health', 'spock85']
[39.5670001507, 'E', 'feisty', 'spock85']
[50.4430000782, 'A', 'e', 'spock85']
[50.4430000782, 'E', 'go', 'spock85', 'east']
[50.4430000782, '-', 'location', 'pitchfork74', 'Paperville3']
[50.4430000782, '-', 'pitchfork', 'pitchfork74']
[50.4430000782, '-', 'location', 'wand75', 'Paperville3']
[50.4430000782, '-', 'wand', 'wand75']
[50.4430000782, '-', 'location', 'spock85', 'Paperville3']
[50.4430000782, '-', 'spock', 'spock85']
[50.4430000782, '+', 'location', 'spock85', 'The_Western_Waste40']

[50.4430000782, '+', 'location', 'spock85']
[80.0460000038, '+', 'location', 'Troll82', 'The_Western_Waste40']
[80.0460000038, '+', 'Troll', 'Troll82']
[90.0499999523, 'E', 'miss', 'Troll82', 'spock85']
[91.0510001183, 'E', 'miss', 'spock85', 'Troll82']
[93.0540001392, 'E', 'miss', 'Troll82', 'spock85']
[94.055999943, 'E', 'miss', 'spock85', 'Troll82']
[94.055999943, 'A', 'unequip', 'spock85', 'spock85']
[94.055999943, 'E', 'missing', 'spock85']
[96.0590000153, 'E', 'hit', 'Troll82', 'spock85']
[96.0590000153, 'E', 'feisty', 'spock85']
[97.0599999428, 'E', 'hit', 'spock85', 'Troll82']
[97.0599999428, 'E', 'feisty', 'Troll82']
[99.0629999638, 'E', 'hit', 'Troll82', 'spock85']
[99.0629999638, 'E', 'tired', 'spock85']
[100.06400013, 'E', 'miss', 'spock85', 'Troll82']
[102.067000151, 'E', 'hit', 'Troll82', 'spock85']
[102.067000151, 'E', 'wounded', 'spock85']
[103.069000006, 'E', 'hit', 'spock85', 'Troll82']
[103.069000006, 'E', 'feisty', 'Troll82']
[105.072000027, 'E', 'hit', 'Troll82', 'spock85']
[105.072000027, 'E', 'faltering', 'spock85']
[106.072999954, 'E', 'hit', 'spock85', 'Troll82']
Recognizing Situations
The Situation Multigraph
A Better Multigraph

- hit
- hit
- hit
- wounded
- faltering
- null
- feisty
Graph Invariant

A graph invariant is any function (in the strict mathematical sense) of a graph.

Thus two graphs that are identical must have the same value of any invariant. However, different graphs may nonetheless have the same value of an invariant.
Coloring Situations By Invariant
Coloring Situations By Invariant
Relatively Few Comparisons Remain To Be Done

<table>
<thead>
<tr>
<th></th>
<th># compares</th>
<th>total runtime (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>no invariant</td>
<td>2,092,801</td>
<td>189.2</td>
</tr>
<tr>
<td>with invariant</td>
<td>16,054</td>
<td>9.4</td>
</tr>
</tbody>
</table>
Variable-Order Markov Models

Based on the same algorithmic technology used for file compression

“Context tree” data structures are used to reduce the number of comparisons

Statistical tests determine how many percepts are used to make any given prediction
Bottom Line Improvements

Increased context-awareness* by a factor of 2
Improved accuracy from 75.1% to 89.4%
Improved speed by 20X

* In terms of the number of recent percepts taken into account
Extending the Horizon: Iterated Prediction

Predicted percept added to percept history, as if it actually happened.
Action Selection: Planning

Choose Action → Predict → Chosen action is then added to percept history, as if the agent perceives itself taking the action. Consequences of the action are predicted.
Intriguing Trends Influencing Near-Future Game AI
Games for Therapy & Education

• The game industry has a big budget

• The education and healthcare industries are quite a bit bigger

• How much education can we do through games?

• Educational games often need to “do AI right”

http://www.seriousgames.org
http://www.seriousgamessource.com
Parallel Processing Abounds

- Multiple cores and graphics coprocessors (GPU's) with capabilities far beyond graphics are the new norm.
- Are multiple processors, multiple GPU's, or physics coprocessors next?
- The total processing cycle budget will rapidly increase.
- Some AI (e.g. learned anticipation) fits great!
Rising Game Development Costs

- Opportunities for AI to help
  - Reducing traditional game AI cost with learning
  - Automation of level design and testing
Summary: Our View of Near-Future Game AI

● Who sees who
  – Trace multiple rays, or do mini-renders, at least off-line

● How to move and where to hide
  – AI can learn this for itself by exploring

● General action selection
  – World model learning combined with planning
For Contact Info and Full Paper References...

• See my website:
  http://www.nps.navy.mil/cs/cjdarken/