Introduction to Artificial Intelligence

Unit # 15

Important Facts

- Games themselves have become a huge industry (it's estimated that games gross more than movies).
- Much of the development of AI for classical games focused on brute force search relying on high-performance computing.
- Video game AI differs greatly in that little of the CPU is available for the AI (as little as **10%,** since the majority of the CPU is tied up with the physics and graphics engines).
- Video game environments (such as can be found in real time strategy games or first-person-shooters) provide a useful test-bed for the application and visualization of AI techniques.

USA Today Article

- http://www.usatoday.com/tech/gaming/2007-09-24-a-i_N.htm

AI is A-OK in new games

Posted 9/25/2007 12:09 AM | Comments 5 | Recommend 12

E-mail | Save | Print | Reprints & Permissions | RSS

Enlarge Microsoft Game Studios

A Covenant "brute" attacks in 'Halo 3.'

By Mike Snider, USA TODAY

Our video-game enemies are smart — and getting smarter. The artificial intelligence that guides in-game characters today leads to far more natural actions and realistic friends and foes than in the past. "As graphics improvements top out, artificial intelligence will (drive) game innovation," says University of California-Santa Cruz professor Michael Mateas. A look at AI evolution:

Mixx it
Other ways to share:
Digg
del.icio.us
Newsvine
Reddit
Facebook
What's this?

USA Today Article (Cont'd)

Enlarge Activision

A medic tends to the wounded in 'Quake Wars.'

Enlarge Activision

Halo 3

Players will notice more enemies aiming to thwart the Master Chief — and more artificially intelligent marines to help him along in this new campaign.

REVIEW: 'Halo 3' lives up to the hype

Where battlefields in previous *Halo* games might have had 15 or so combatants, *Halo 3* will feature as many as 40 enemies and allies fighting intelligently. Typical *Halo 2* enemies or squad mates had at most about 50 behaviors; in *Halo 3*, programmers have upped that to as many as 70. Each character has more rules attached to each behavior — up to 10,000, compared with 10 in the past.

FIND MORE STORIES IN: World War II | Russians | Nazi | Microsoft Xbox | Middle Eastern | Halo | Windows PCs | Software | Spartan | Call of Duty | Covenant | Master Chief | John Dean | Isla | Modern Warfare | Jason West | University of California-Santa Cruz | Bungie | Infinity Ward | Michael Mateas | Legendary Edition | Quake II

The computer-controlled characters use their AI brains to calculate all these rules in milliseconds just as humans' can, and their actions are harder to predict. In the past, a Covenant "brute" (an alien attacker) would likely seek cover when you fired on him. Now the enemy can deploy a bubble shield, hold its ground, assess the situation and perhaps find a method of attack.

Technologies/Skills Required for Development of Video Games

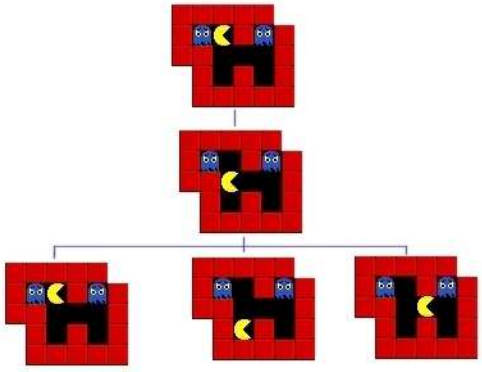
- Computer Science
  - Computer Graphics
  - **Artificial Intelligence**
  - Networking
  - Human-Computer Interaction
  - Software Engineering
- Visual Arts
- Music

Overview of Courses on AI in Video Games

- http://www.cse.lehigh.edu/~munoz/CSE348/
- http://homepage.cs.uri.edu/faculty/hamel/courses/2010/spring2010/csc481/
- http://www.cs.ucf.edu/~kstanley/syllabus_cap4053spring09.html
- http://www.bowdoin.edu/~smajerci/courses/cs380/syllabus.html
- Many more

Game Trees

- Is this tree looks familiar?
- How can we describe the best move?



Ms Pac-Man Competition

- http://cswww.essex.ac.uk/staff/sml/pacman/PacManContest.html

Best Score Yet

The best score yet of any agent that I've run on my machine is:

20640 (Mehrabian and Khosravi)

Entry	Authors	Affiliation
Southern Maine	Fitzgerald, Kemeraitis and Bates Congdon	University of Southern Maine
Handa	Hisashi Handa	Okayama University, Japan
Dortmund	Piatkowski, Neugebauer and Naujoks	University of Dortmund
Gan	Gan, Liu and Bao	n/a
Flensbank	Flensbak and Yannakakis	IT University of Copenhagen, Denmark
Leandro	Liu and Gomez	Universidad Nacional de Colombia
Ruck	Kashifuji, Oda, Matsumoto, Hirono and Thawonmas	Ritsumeikan University, Kyoto, Japan
Pacool	Khosravi and Mehrabian	Sharif Univ. of Tech., Tehran, Iran
Wirth	Wirth and Gallagher	University of Queensland, Australia
Shirikawa	Ando, Shirakawa, Otsuka, Takahashi, Totsuka and Nagao	Yokohama National University, Japan
Jabbari	Pante Jabbari	Sharif Univ. of Tech., Tehran, Iran

Path Finding

- The object of path-finding in many games from first or third-person-shooters, to real-time strategy games is identifying a path from point A to point B.
- In most cases, multiple paths exist from point A to point B, so constraints may exist such as shortest path, or least cost.
- Consider, for example, two points separated by a hill. It may in fact be faster to go around the hill, than going over it, but going up the hill could give some advantage to the player (say an archer with an enemy opponent on the opposite down-slope).

Search Algorithms

- The landscape upon which we're to plot a route is a graph of nodes representing waypoints.
- Each edge of the graph has a given cost (for example, plains could have an edge cost of one, where inclines could have an edge cost consistent with its slope).
- Considering path-finding as search through a graph of nodes with weighted edges, the **A*** search algorithm is ideal for this application.
- It is optimal compared to DFS and BFS, and can give us the optimal path.

Problem with A*

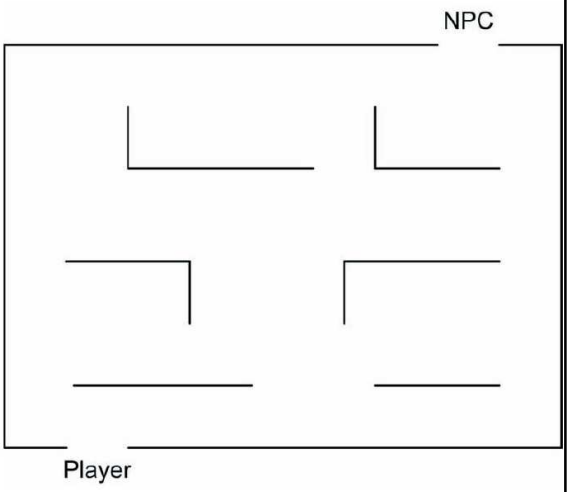
- The problem with A* is that it's a very computation-intensive algorithm.
- Considering the number of agents in a real-time strategy game that need to move around the map, the amount of time taken by A* would be multiplied.
- As the AI in a real-time strategy game would also need to support high-level goal planning and economic strategy, path-finding is but one element that should be optimized.

Table Lookup

- Any map can be reduced to a graph, where the nodes are the places that can be visited, and the edges are the paths between the nodes.
- Reducing a map in this way does a couple of things.
- It potentially reduces a map with an infinite number of points into a graph with fewer points.
- The edges, or the paths between nodes in the graph, define the various ways that we can travel around our graph (and our map).

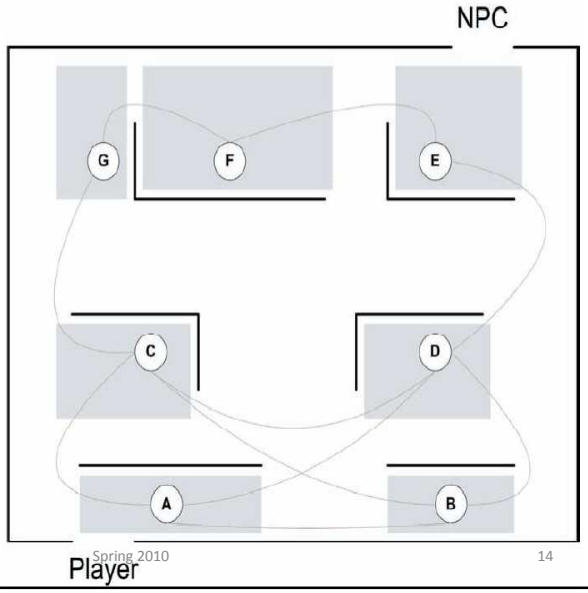
Map

- We'll assume here, for the sake of simplicity, that the NPC always sees the player.
- In a more complicated system, the player would need to be in the NPC's field-of-view (FOV) in order for the NPC to identify the player's presence.
- Read "AI Cheating" on Wikipedia



Converting Map into Graph

- We'll implement two basic strategies.
- If our NPC is healthy, we'll take an offensive strategy to attack the player.
- If the NPC is not healthy, then we'll take a defensive strategy.



Offensive Strategy

- The rows represent the current location of our NPC, while the columns represent the current location of the player.
- If the NPC is at node E, and the player is around node A, then the NPC will use the offensive strategy table and move to node D.
- If the player then moves from node A to node C, we use the table again for the NPC at node D, and the player at node C, which results in the NPC moving to node C (on the attack).

Player (Opponent)

NPC	A	B	C	D	E	F	G
A		B	C	D	D	C	C
B	A		C	D	D	C	C
C	A	B		D	D	G	G
D	A	B	C		E	E	C
E	D	D	F	D		F	F
F	G	E	G	E	E		G
G	C	C	C	C	F	F	

Offensive Strategy

Defensive Strategy

- The defensive strategy is one of taking some time to heal by avoiding the player.
- Take, for example, the NPC at node D and the player again around node A. The lookup table returns a move from node D to node E, essentially putting distance between us and the player.
- If the player then moved to node D, the lookup table would return node F, moving away from an eventual move by the player to node E.

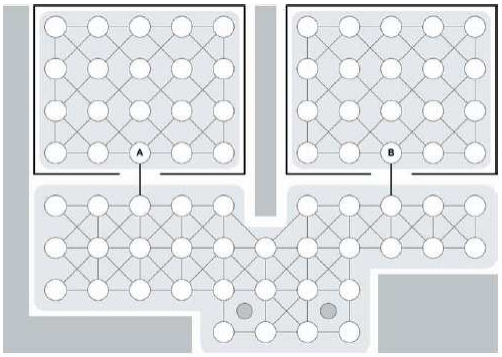
Player (Opponent)

NPC	A	B	C	D	E	F	G
A		C	D	C	-	B	B
B	D		G	C	A	-	-
C	D	G		G	A	B	D
D	E	E	E		A	B	B
E	-	F	-	F		D	D
F	E	G	E	G	G		E
G	F	-	F	-	C	C	

Defensive Strategy

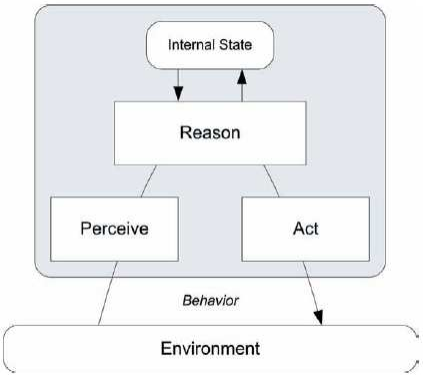
Multiple Maps

- In large environments, it can also be possible to segregate a map into multiple connected maps, each having funnel points for which an NPC may travel.
- Rather than including a single lookup table for all points, three separate lookup tables would exist.
- If our NPC agent was in the left room, it could use the lookup table to determine which path to take to a given node.
- If the destination were outside of the room, it would by default move to node A, and from there, the external lookup table would take over for routing the NPC to its destination.

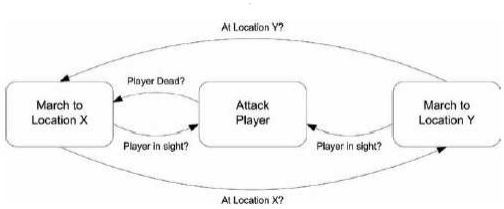


NPC Behavior

- The behavior of an NPC can't be considered in isolation, because behavior is ultimately grounded in the environment.
- The NPC must be able to *perceive* its environment (see, hear, etc.).
- With the information perceived from the environment (as well as internal state information such as motivation), the NPC can *reason* about what should be done.
- The result of reasoning is potentially an intentional act, which is performed as an *action*



Static State Machines



- The NPC marches between two locations, guarding some entry from the player.
- When the player is in sight, the NPC fights to the death.
- If the NPC dies, then the state machine is no longer active.
- If the NPC defeats the player, it begins marching again toward location X.
- It is possible to add transition probabilities to give the NPC a small element of randomness.

Layered Based Architectures

- If the NPC's health is low, it should head to the infirmary.
- If the NPC is low on ammo, it should head to the armory to reload.
- If more than one player appears to attack, should it fight, or rush to the guard house to pull the alarm?

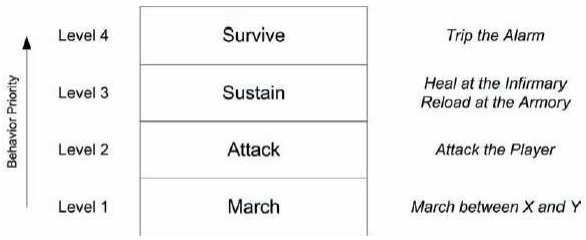
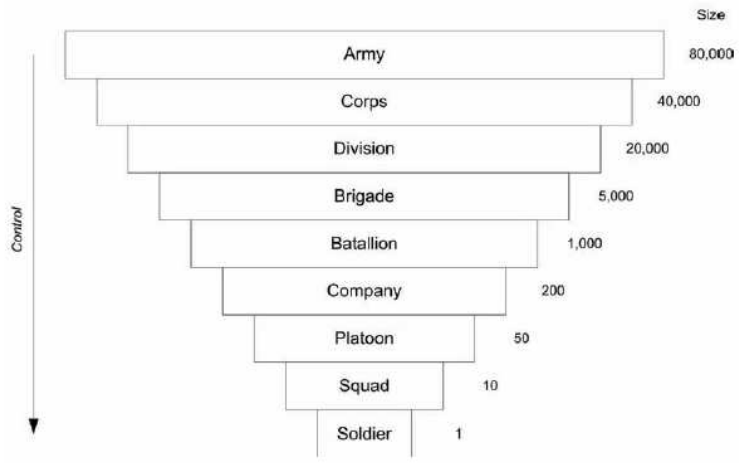


FIGURE 4.19: Behavior layers for the simple NPC.

Team AI

- In many games, there's not just a single NPC soldier that we're fighting against, but an entire army that must work together in harmony with a single or handful of goals in mind.
- The control can exist at a number of levels within a hierarchy, from the squad leader directing troops to flank an enemy in the field using real-time battlefield information, to the general managing his entire army to implement the higher-level military strategies.
- Managing an overall army can entail a large number of problems, from scheduling and production, to resource allocation, to the overall strategy and tactics of the force in the field.

Organizational Structure of a Simplified Army

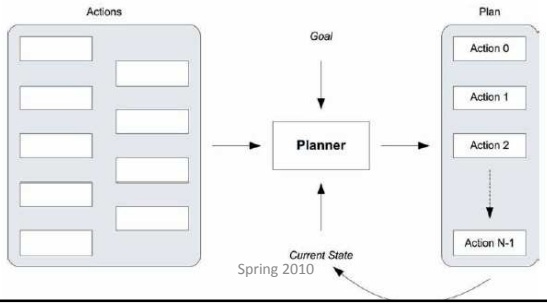


Strategic and Tactical Levels

- The problem here can be simplified greatly by minimizing the number of levels or organization.
- First, the individual soldiers simply follow orders, and unless they are routed, can fight to the death.
- These individual units can be programmed using finite state machines, or other simple mechanisms for behavior.
- At the highest level is the Strategic AI. This layer of the AI has the global view of the battlefield, troop strengths, available resources, etc.
- Next is the Tactical AI whose job is to implement the strategy provided from above.
- At the lowest rung is the individual soldier, on whose collective shoulders that overall strategy relies.

Goals and Plans

- A goal is a condition that is desired to be satisfied. Example goals include taking an enemy position, destroying a bridge, flanking an enemy, etc.
- To reach a goal, we must perform a series of *actions*, which are independent steps within a plan.
- The *plan,* then, is a set of actions that when performed in the given order, achieve a goal.



Goals and Plans

Goal: Eliminate(Player)

Prerequisites:

Covered_Position(Player)

Alive(Player)

Plan:

Action-1: Identify_Flanking_Position(NPC_1)

Action-2: Covering_Fire(NPC_2)

Action-3: Move_to_Flanking_Position(NPC_1)

Action-4: Fire_at_Player(NPC_1)

Action-5: Rush_Player_Position(NPC_2)