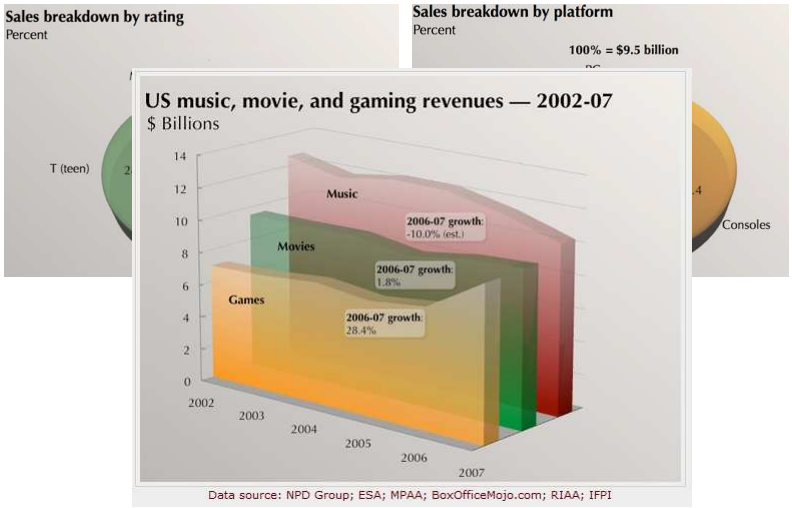
Video Game AI

Video Game AI

- Classical Games
  - Focus on optimal players using computationally expensive search techniques
- Video Game AI
  - Refers to games such as First Person Shooters, Real Time Strategy, Role Playing, Action/Arcade Games
  - Differs greatly from classical AI since relatively little CPU available for the AI
    - Focus on believable characters, behaviors
    - **Rarely rises above the level of finite state machines**

Video Game Industry



January, 2008

Video Game AI

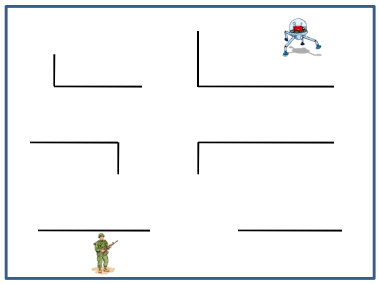
- Growing area for research
  - E.g. Doom AI players
  - Starting to see AI game engines
  - Several research efforts in believable characters
- Most video games though use simple algorithms to reduce the CPU load
  - Some techniques
    - Movement and Path Finding
    - Reasoning
    - State Machines
    - Layered Architecture
    - Team AI
    - Planning

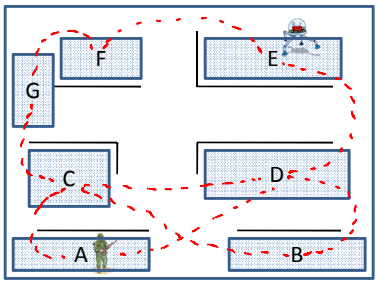
Movement and Path Finding

- Many games require the AI to go from point A to point B
  - Generally multiple paths, obstacles, scenarios
- Easy enough – we already covered A*
  - Optimal compared to DFS and BFS
  - Convert problem into a graph and run A*, problem solved?

Simple Solution to Pathfinding

- Reduce map to a graph with fewer edges, only along select routes, to simplify the problem





If player within the rectangle then position is simply referenced as the rectangle
Assume global knowledge of the game board

Movement Strategies

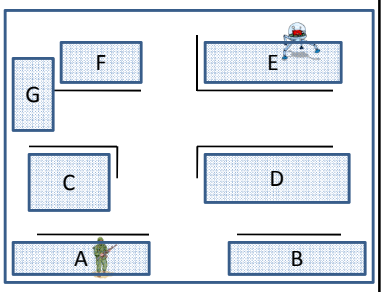
- Can encode attack/defense strategies using a simple lookup table on the simple graph
- Offense
  - If healthy use the attack strategy
- Defense
  - If not healthy take the defensive strategy

Offensive Lookup Table

Human

AI

	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	

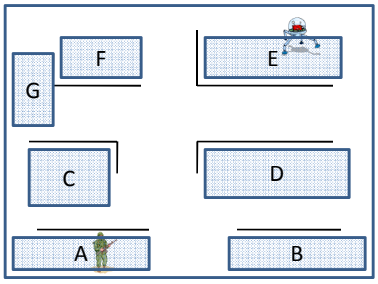


Defensive Lookup Table

Human

AI

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

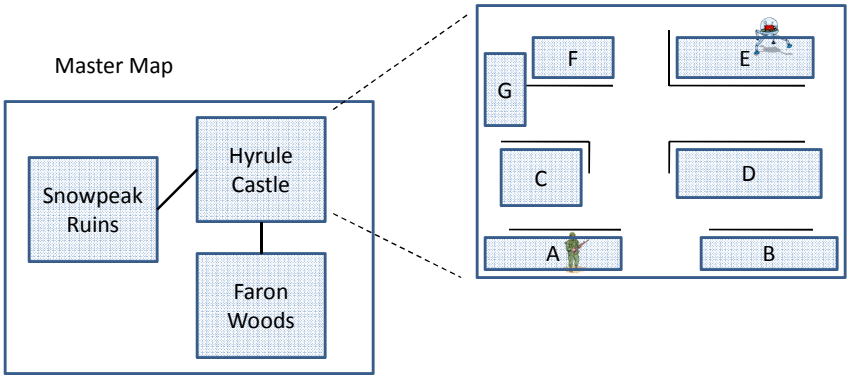


Lookup Table Strategy

- Provides for reactive behaviors
  - AI reacts to the movement of the player
- Stateless
  - No state kept between lookups
  - AI simply uses its position and the players position to determine the next move
  - Fast, efficient, no search required
  - Can add unpredictability by randomly selecting moves with some probability

Scaling Up

- For large environments, can make a map of maps

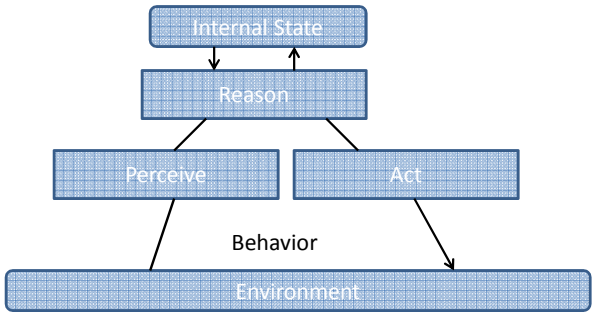


Lookup Table Problems

- Tends to fall apart if the environment changes over time
- Can sometimes result in bizarre behavior at the thresholds between zones and because of the vast simplification
- Video example: http://www.ai-blog.net/
  - Direct link: http://www.youtube.com/watch?v=lw9G-8gL5o0

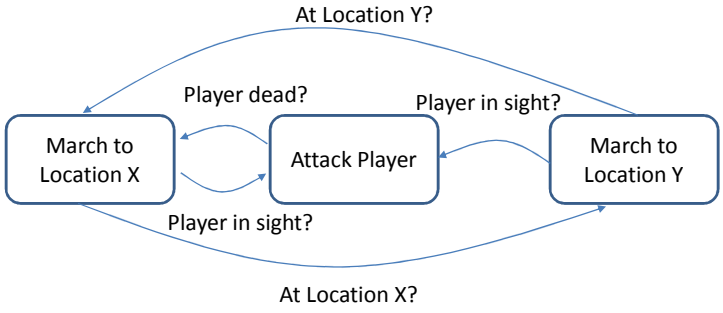
Agent Behavior and Environment

- AI's behavior ultimately grounded in the environment
  - Environment may change, AI should reason about the environment to perform some action, affecting environment
  - Closed loop of reasoning



Basic Reasoning Abilities – State Machines

- Perhaps simplest and most common is the state machine; example with 3 mental states for a sentry guarding two locations



Finite State Machine

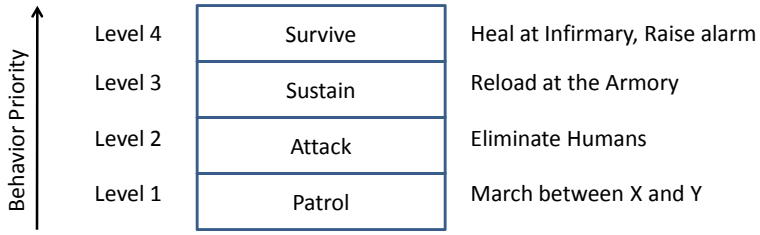
- Could exhibit more intelligence with more states, more transitions
- Simple to write and easy to debug
- Predictable but can add transition probabilities for element of randomness

Basic Reasoning Abilities – Layered Behavior Architecture

- Previous example only marched, attacked
  - What if other things on the AI's mind?
    - If health low, go to the hospital
    - If low on ammo, go to the armory
    - If outnumbered, stay and fight or run for the alarm?
  - One solution to handle the conflict in action selection is Rodney Brooks' **subsumption architecture**
    - Implemented for robot systems

Subsumption Architecture

- Responsibilities confined to isolated layers, but a layer subsumes (takes priority over) another if the need arises
- Within each layer we might implement the behavior as a FSM



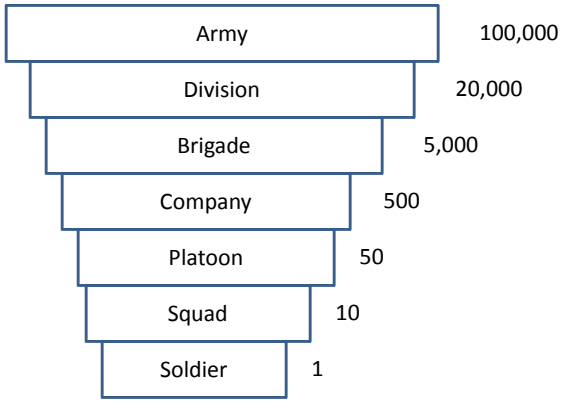
Basic Reasoning Abilities – Other Approaches

- Machine Learning – will discuss shortly\
  - Rote learning
  - Neural networks
  - Genetic algorithms
  - Others
- Usually in the context of a scripting language for character behavior

Team AI

- Often an entire army that must work together, how to manage the multitudes of agent AI's?
- Use the same organizational hierarchy as real life
  - Soldiers implemented as finite state machines or other simple mechanism; might always follow orders
  - Higher-level units subdivided at higher strategic levels

Team AI Hierarchy



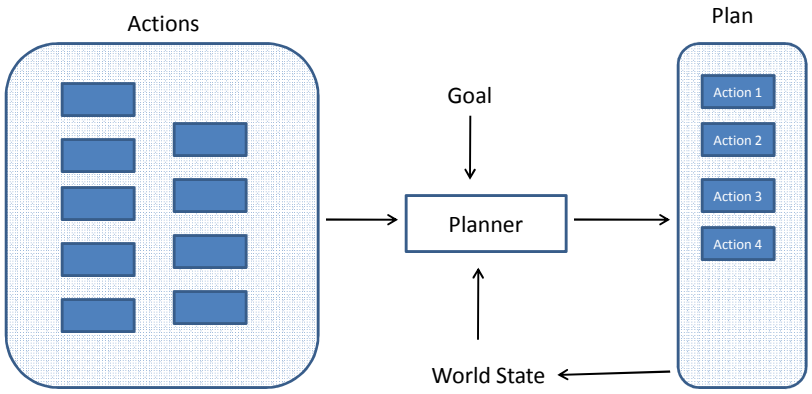
Hierarchical Control

- One mechanism is to define a goal and then create plans to reach that goal
  - Goal might be a win-situation or intermediate goal closer to a win
    - E.g. taking an enemy position, destroying a bridge, flanking the enemy, etc.
  - To reach goal must perform a series of actions
  - Actions may have preconditions for the action to be performed and postconditions as a result
    - E.g. planting a charge requires the demolitions expert, action is to plant the charge, postcondition is the target is destroyed
  - The plan is the set of actions that when performed in the given order achieves the goal

Sample Plan

- Goal Eliminate(Player)
  - Preconditions
    - Covered_Position(Player)
    - Alive(Player)
  - Plan
    - Action 1: Identify_Flanking_Position(AI_1)
    - Action 2: Fire_At_Player(AI_2)
    - Action 3: Move_To_Flanking_Position(AI_1)
    - Action 4: Fire_At_Player(AI_1)
    - Action 5: Rush_Player(AI_2)

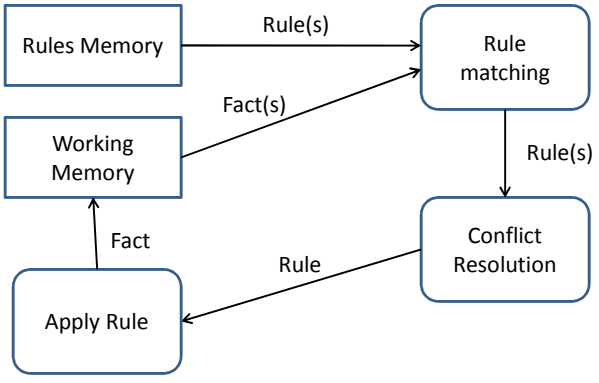
Planner



How do we come up with the Plan? Search!

Rule-Based AI

- In a similar manner, we can also create a rule-based AI system



Rule Example

Working Memory
(opponent-1 size 1000)
(opponent-2 size 100)
(army size 1000)

Rule
(rule "Attack weaker opponents"
(< (?opponent size) 500)
(> (army size) 500)
--> (attack ?opponent))

After applying the rule to our working memory, we get: (attack opponent-2)

Rule-Based System

- RBS could remove facts as driven by consequents to reflect the dynamic nature of the game
- With a sufficient rule/fact base, can make appropriate actions, useful for RTS games
  - E.g. first, focus might be to build bases and collect resources because military rules not able to fire
- Difficult to code the rule base; requires some CPU time to apply the rule engine with a large number of rules
  - Does allow strategy to be decoupled from the game engine, permit later tweaking by game developers

References

- Jones, Tim. Artificial Intelligence, A Systems Approach, Infinity Science Press
- http://www.gameai.com/
- Game AI, State of the Industry: http://www.aiwisdom.com/bytopic_stateoftheindustry.html