3D Computer Game Programming

AI for Games

Excerpted from Introduction to Game Development, ed. S. Rabin, 2010 and other game articles

Artificial Intelligence (AI)

- Intelligence embodied in a man-made device
- Human level AI still unobtainable

Game AI

- What is Game AI? (definition)
- How is it different from other AI fields?
- What are common AI techniques used in game?

What is Game AI?

- What is considered Game AI?
  - Pathfinding?
  - Is it any Non-Player Character (NPC) behavior?
    - A single “if” statement?
    - Scripted behavior?
  - Animation selection?
  - Automatically generated environment?
  - …

Possible Game AI Definition

Inclusive view of game AI:

"Game AI is anything that contributes to the perceived intelligence of an entity, regardless of what's under the hood."

Goals of an AI Game Programmer

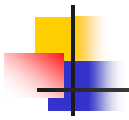
Different than academic or defense industry.

Goal: to create both entertaining and challenging opponent while shipping the product on time.

1. AI must be intelligent, yet purposely flawed.
2. AI must have no unintended weaknesses.
3. AI must perform within the CPU and memory constraints.
4. AI must be configurable by game designers or players.
5. AI must not keep the game from shipping.

1. AI must be intelligent, yet purposely flawed.
   - Opponents must present a challenge.
   - Opponents must keep the game entertaining and fun.
   - Opponents must lose to the player in a challenging and fun manner.

2. AI must have no unintended weaknesses.
   - There must be no “golden paths” to defeating the AI every time in the same way.
   - The AI must not fail miserable or look dumb.



3. AI must perform within the CPU and memory constraints.

- Most games are real time and must have their AIs react in real time.
- Game AI seldom receives more than 10 to 20 percent of the frame time.

4. AI must be configurable by game designers or players.

- Designers must be able to adjust the difficulty level, tune the AI, and occasionally script specific interactions.
- If the game is extensible, players can tweak or customize the AI.

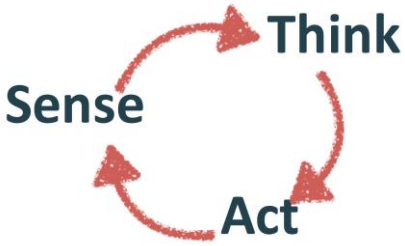
5. AI must not keep the game from shipping.
   - The AI techniques employed must not put the game at risk.
   - Experimental techniques must be proved early in the development cycle during preproduction.

Game Agents

- In most games, the purpose of AI is to create an intelligent agents, aka Non-Player Character (NPC).
- NPC may act as an
  - Opponent
  - Ally
  - Neutral character

Game Agents

- Continually loops through the "**Sense-Think-Act" cycle**
  - This cycle is a simple conceptual framework for organizing intelligent behavior.
  - Optional learning or remembering step.



Sense-Think-Act Cycle: Sensing

- The game agent must have information about the current state of the world.
- Agent can have access to perfect information of the game world
  - Game World Information
    - Complete terrain layout
    - Location and state of every game object
    - Location and state of player
- But isn't this cheating? Also, it may be expensive/difficult to tease out useful and pertinent info. Thus, game agents are usually given limitations.

Sensing: Enforcing Limitations

- Enforce human limitations
- Limitations such as
  - Not knowing about unexplored areas
  - Not seeing through walls
  - Not knowing location or state of player
- Can only know about things seen, heard, or told about
- Must create a sensing model – how to perceive the world.

Sensing: Vision Model

- The following steps approximate human vision.
  - An agent gets a list of pertinent objects or agents. (e.g. use raycasting)
  - For each object, calculate:
    1. Is it within the viewing distance of the agent?
    2. Is it within the viewing angle of the agent? (usually dot product is used)
    3. Is it unobscured by the environment?

Sensing: Vision Model

- The vision model in the previous slide does not detect if just a portion of an object is visible.
- Isn't vision more than just detecting the existence of objects? What about recognizing interesting terrain features such as hiding spots and high-risk areas.
  - Designers can mark them.
  - Develop an algorithm to discover them from the world representation.

Sensing: Hearing

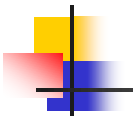
- Allowing agents to sense through hearing.
  - If the player drops an object, a guard behind a wall can notice.
  - If the player starts wildly firing his gun, agents who can't see the player might rush to the scene (the player's location.)
- Hearing is commonly modeled through event-driven notifications.
  - If the player performs an action that makes a noise, the game will compute where the noise might travel to and inform any agent within that range.

Sensing: Communication

- Agents might talk amongst themselves!
  - Guards might alert other guards
  - Agents witness player location and spread the word
- Model sensed knowledge through communication
  - Similar to hearing model, event-driven can be used to model this communication.
  - When agents within vicinity of each other, the information will be sent directly from one agent to the other agents.

Sensing: Reaction Times

- When modeling sensing, we have to build artificial reaction time.
- Agents should NOT see, hear, communicate instantaneously.
- Players notice!
- Build in artificial reaction times
  - Vision: ¼ to ½ second
  - Hearing: ¼ to ½ second
  - Communication: > 2 seconds



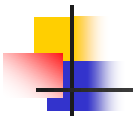
Sense-Think-Act Cycle: Thinking

- Sensed information gathered
- Must process sensed information

- Two primary ways in which an agent makes a decision in a game:
  - Process using pre-coded expert knowledge (hardcoded if-then rules with randomness introduced to make agents less predictable)
  - Use a search algorithm to find a near optimal solution

Thinking: Expert Knowledge

- Many different systems for encoding expert knowledge:
  - Finite-state machines
  - Production systems
  - Decision trees
  - Logical inference
- Encoding expert knowledge is relatively easy.
  - Write a series of if-then statement.
- Problems with expert knowledge
  - Not very scalable.
  - But for most agents solve narrow problem domains, limited expert knowledge is sufficient.



Thinking: Search

- Employs search algorithm to find an optimal or near-optimal solution.
- A* pathfinding common use of search.

- The most common use of search is game agent navigation -- planning where the agent should move next.

Thinking: Machine Learning

- If imparting expert knowledge and search are both not reasonable/possible, then machine learning might work.
- Examples:
  - Reinforcement learning
  - Neural networks
  - Decision tree learning
- Not often used by game developers
  - Requires deep knowledge and years of experience to make them work
  - Often doesn't outperform other techniques in terms of performance, robustness, testability, ease of programming, and ease of tuning.

Thinking: Avoid Flip-Flopping

- Must prevent flip-flopping of decisions
  - If a decision is reevaluated every frame, the agent will be paralyzed in a perpetual moment of indecisiveness.
- Must make a decision and stick with it
  - Until situation changes enough
  - Until enough time has passed

Sense-Think-Act Cycle: Acting

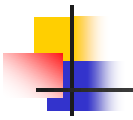
- Sensing and thinking steps invisible to player
- Acting is how player witnesses intelligence
- There are numerous agent actions, for example:
  - Change locations
  - Pick up object
  - Play animation
  - Play sound effect
  - Converse with player
  - Fire weapon
  - …

Acting: Showing Intelligence

- Adeptness and subtlety of actions impact perceived level of intelligence $\Rightarrow$ places enormous burden on asset generation (variety and aesthetic quality of the animations, sound effects, and dialogs.)
- Agent can only express intelligence in terms of vocabulary of actions

Acting: Showing Intelligence

- Assets convey hidden work (sensing and thinking) to the player to make it look intelligent
  - For example, if the agent concluded that it will inevitably die in near future
    - One way – the agent sits and dies. => the player perceives a dumb agent.
    - The other way – the agent shouts “Oh, no” as it is about to die. => the player perceives a smart agent who comprehends the situation.



Extra Step in Cycle: Learning and Remembering

- Optional step
- Not necessary in many games
  - Agents don't live long enough
  - Game design might not desire it
- In game in which the agent is persistent a little bit longer, it might be useful.

Learning

- Remembering outcomes and then generalizing and predicting future outcomes.
- Simplest approach: gather statistics
  - If 80% of time player attacks from left Then expect this likely event.
- Adapts to player behavior

Remembering

- Remember hard facts (past observances) and use them in the “Think” step.
- For example,
  - Where was the player last seen?
  - What weapon did the player have?
  - Where did I last see a health pack?
- Memories should fade
  - Helps keep memory requirements lower
  - Simulates poor, imprecise, selective human memory

Remembering within the World

- All memory doesn’t need to be stored in the agent – can be stored in the world
- For example, a smart terrain:
  - Agents get slaughtered in a certain area
  - Area might begin to “smell of death”
  - Agent’s path planning will avoid the area

Making Agents Stupid

- It is very easy to destroy player
  - Make agents faster, stronger, more accurate
- Sometimes necessary to dumb down agents, for example:
  - Make shooting less accurate
  - Make longer reaction times
  - Engage player only one at a time
  - Change locations to make self more vulnerable

Agent Cheating

- Players don't like agent cheating
  - When agent given unfair advantage in speed, strength, or knowledge
- Sometimes necessary
  - For highest difficultly levels to provide a supreme challenge to the player
  - For CPU computation reasons
  - For development time reasons
- Don't let the player catch you cheating!
  - Consider letting the player know upfront

Common AI Techniques

- Common AI Techniques include
  - A* Pathfinding
  - Behavior Tree
  - Command Hierarchy
  - Dead reckoning
  - Emergent behavior
  - Flocking
  - Formations
  - Influence mapping
  - Level-of-Detail AI
  - Manager task assignment
  - Obstacle Avoidance
  - State Machine
  - Subsumption architecture
  - Terrain analysis
  - …

Common AI Techniques: A* Pathfinding

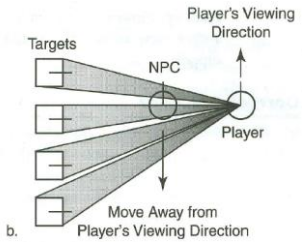
- It is a directed search algorithm used for finding an optimal path through the game world
- A* pathfinding
  - The environment must first be represented by a data structure where movement is allowed.
  - Given a start position and goal position, A* returns a list of points that defines the move path.
- A* is regarded as the best
  - Guaranteed to find a path if one exists
  - Will find the optimal path
  - Very efficient and fast

Common AI Techniques: Command Hierarchy

- It is a strategy for dealing with decisions at different levels
  - From the general down to the foot soldier
- Modeled after military hierarchies
  - General directs high-level strategy
  - Foot soldier concentrates on combat
- It is often used in real-time strategy or turn-based games where there are typically three easily identifiable levels of decisions: overall strategy, squad tactics, and individual combats.
- It is also useful when a large number of agents must have an overall coherency.

Command Hierarchy Example

- Squad Tatics
  - Move the squad to in the direction of less threat by the player



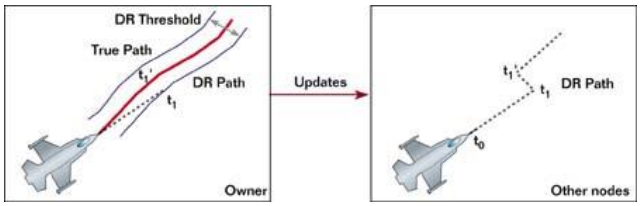
- Individual NPCs
  - Stay close to the team

Common AI Techniques: Dead Reckoning

- Method for **predicting object's future position** based on current position, velocity and acceleration
- Works well since movement is generally close to a straight line over short time periods
- Game Examples:
  - In a FPS game, before firing, predicting position of moving targets
  - In a sports game, predicting position of other player.
- Also used to mitigate the effects of networked game lag and improve player experience

Dead Reckoning Example

- For a multiplayer networked game, when a vehicle is created on a computer A, A sends out information to all the computers on the network to generate and render a vehicle.
- With the use of dead reckoning, the vehicle state will be updated one every five seconds instead of every update.



Common AI Techniques: Flocking

- Example of emergent behavior
  - Simulates flocking birds, schooling fish
- Developed by Craig Reynolds (SIGGRAPH'87 paper)
- Each creature follows three classic movement rules:
  1. Separation – avoid local flockmates
  2. Alignment – steer toward average heading
  3. Cohesion – steer toward average position
- The group behavior emerges from the individual rules.

Common AI Techniques: Flocking

- Games usually use flocking to control background creatures such as birds and fish.
- Inspired other movements such as formation and swarming.

https://bryanduggan.org/tag/unity3d/

Common AI Techniques: Formations

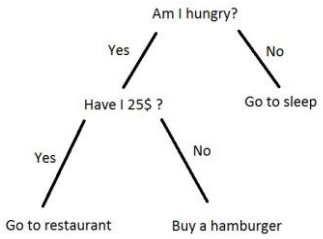
- Group movement technique
  - Mimics military formations
- Similar to flocking, but actually distinct in that each unit guided toward formation position
  - Flocking doesn't dictate goal positions
- In games, formation can be used to organize the movement of ground troops, vehicles, or air-craft. The game *Age of Empires 2* pioneered several key techniques for formations.

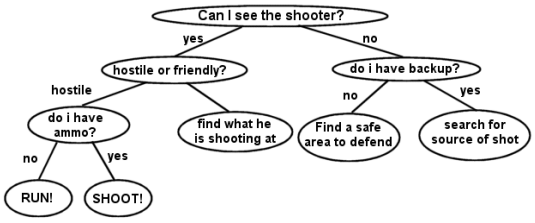
Promising AI Techniques

- Show potential for future
- Generally not used for most games
- They include
  - Bayesian Networks
  - Blackboard Architecture
  - Decision Tree Learning
  - Filtered Randomness
  - Fuzzy Logic
  - Genetic Algorithm
  - N-Gram Statistical Prediction
  - Neural Networks
  - Production Systems
  - Reinforcement Learning
  - Reputation System
  - Speech Recognition
  - …

Promising AI Techniques: Decision Tree Learning

- Constructs a decision tree based on observed measurements from game world
  - For example, inputs are health and ammunition of a bot predicting the probability of the bot surviving an engagement with the player.





Promising AI Techniques: Decision Tree Learning

- Best known game use: **Black & White**
  - Creature would learned what to eat in the world based on feedback from the player and world
  - Then, a decision tree is created to reflect what the creature has learned from experiences.
  - The creature can then use the decision tree to decide whether certain objects can be used to satisfy his hunger.

Example	Allegiance	Defense	Tribe	Attack
1	friendly	weak	Celtic	no
2	enemy	weak	Celtic	yes
3	friendly	strong	Norse	no
4	enemy	strong	Norse	no
5	friendly	weak	Greek	no
6	enemy	medium	Greek	yes
7	enemy	strong	Greek	no
8	enemy	medium	Aztec	yes
9	friendly	weak	Aztec	no

Promising AI Techniques: Production Systems

- Formal rule-based system
  - Captures expert knowledge in the form of rules
  - Consists of database of rules and facts and inference engine to decide which rules trigger – resolves conflicts between rules
- Example
  - *Soar* was experimented with Quake 2 bots. Needed 800 rules for competent opponent.

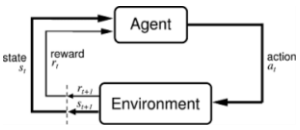
  - Microsoft's Sports Group experimented with a production system to create team sports games.

Latest Updates in GameAI

- 2016 - Google DeepMind's AlphaGo (version: Lee) defeated Lee Sedol 4–1. Lee Sedol is a 9 dan professional Korean Go champion who won 27 major tournaments from 2002 to 2016.

- 2017 - An OpenAI-machined learned bot played at The International 2017 Dota 2 tournament in August 2017. It won during a 1v1 demonstration game against professional Dota 2 player Dendi.

- 2018 – Many game studios train and develop NPCs using Deep Learning and Reinforcement Learning and deploy them to real games.



Summary

- Game AI is different from many other AI fields.
- The goal is to create intelligent agent that results in engaging and enjoyable experience for the player.
- The goal is not to beat the player, but rather to lose in a fun and challenging way.
- Most games are populated by agents that sense, think and act. Advanced agents might also learn and remember.
- There are many common and promising AI techniques. There is no single solution.