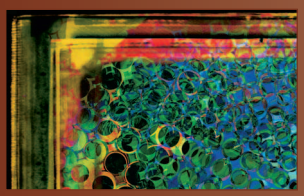
Game Smarts

Michael van Lent, ICT



AI advances will move games beyond battles and sports to explore complex social interactions.

Artificial intelligence, often referred to by the acronym AI, has been a part of videogames since their early days. While the earliest videogames such as *Spacewar!* and *Pong* pitted players against each other, there soon followed single-player games in which the player competed against computer-controlled adversaries. Computer opponents, including everything from the ghosts in *Pac-Man* to the most devious virtual general in the latest strategy game, show only some of the roles AI plays in today's games.

In both military-themed strategy and team sports games, the AI not only fills the role of the opposing commander or coach but also that of the individual units or team members the player and strategic opponent control. This is particularly challenging in sports games because the human playing the game will likely know a great deal about each real-life athlete and have specific expectations the virtual athlete must satisfy.

In more story-centric game genres, such as role-playing and adventure games, the player generally controls a single character, usually the hero, while the AI controls the rest of the characters who play out the story. These *nonplayer characters* (NPC) can range from bit parts, such as the vendor who sells better weapons, to central roles such as the main antagonist and recurring allies. Cortana from Bungee's *Halo* game series provides a particularly interesting example of an NPC ally because she is actually an artificial intelligence within the game world. Thus, we have today's game AI providing the behavior logic for a futuristic, advanced AI.

AI AND GAMES

For game developers, AI has come to mean the broad range of techniques used to generate the behavior of these opponents, battlefield units, teammates, NPCs, or anything else that acts in the game with simulated intelligence. A few of these techniques, such as finite state machines and the heuristic A* search algorithm, have proven themselves in many games over the years. At the most basic level, the finite state machines implemented in games consist of

- the several states a character can be in,
- a set of conditions for when to change states, and
- a chunk of code to implement the character's behavior for each state.

For example, an evil alien might have three states: hunt, fight, and flee. In the fight state, the alien might move toward the player while firing its laser cannon. If, however, our alien nemesis's health state drops below 25 percent while in the fight state, it will transition to the flee state and run back to the mothership. Finite state machines can effectively decompose the full range of a character's behavior into independent chunks with simple logic to transition between them. However, as the complexity of a character's behavior increases, the number of states can explode.

In these examples, the AI-controlled alien moves toward the player to fight and away from the player to flee. Both of these behaviors require the game's AI to calculate a path from the alien's current position to a good attack position while avoiding walls and other obstacles in the virtual environment. This problem, called *path planning*, is one of the most common challenges in game AI.

Path planning is needed when a squad of AI infantry must move into an attack position, when an AI running back needs to get downfield, and when an AI sidekick needs to follow the player through a maze of rooms and doorways.

The A* search algorithm forms the basis for how most games compute the path an AI character will take to get from point A to point B. The A* search maintains a list of partial paths and continually expands on the partial path with the shortest combination of distance explored so far and estimated distance to the goal.

With certain restrictions, A* search is theoretically an optimally efficient search algorithm. But, because game developers can tightly control the problem's context, several interesting variations of A* search have been identified that, while not faster in the theoretical sense, can be more efficient for the set of problems encountered in a specific game.

Path planning has been a primary focus of AI specialists in the game industry for many years. Today's games can calculate paths for hundreds of units using only a small fraction of the available processing power, which leaves the bulk available for other demands.

RAISING THE BAR

While solutions like finite state machines and A*-based path planning have been successful so far, AI for games is rapidly becoming a much harder problem. Each new generation of games has larger and more detailed environments as well as greater numbers of AI-controlled characters and units, each with increasingly complex actions.

For game developers, just maintaining the same level of AI behavior in the face of this increasing complexity poses a major challenge. Path planning was a fairly straightforward problem in early games like id's *Doom*, which, from the AI perspective, had a 2D world with only a few obstacles—none of which moved. Today's games have fully 3D worlds with numerous mobile objects, not to mention stairways, ladders, teleport stations, and the like. In these games, the basic problem of moving to a target position without getting stuck takes on a whole new level of complexity.

In addition, players have become more demanding, looking for increasingly complex interactions with the game's characters and between those characters and the environment. Each new game must have some novel twist on the usual set of AI actions. Taking cover behind crates and barrels is followed by throwing grenades back at the player before they explode, which in turn is followed by coordinated team attacks and so on.

To add to the pressure, the quality of a game's artificial intelligence has increasingly become a topic of discussion in game reviews and Web forums. Where games were originally developed completely in-house, many companies are now building atop licensed game engines such as the Unreal engine, Gamebryo, and Crytek's upcoming CryENGINE.

While this trend started with first-person shooter (FPS) games, it has now spread to a much wider range of genres. For example, Bethesda's *Elder Scrolls IV: Oblivion* role-playing game takes advantage of Gamebryo, as does Sierra's real-time strategy game *Empire Earth II*. The massively multiplayer game *Lineage II* uses the Unreal engine.

The quality of a game's artificial intelligence has increasingly become a topic of discussion in game reviews and Web forums.

This shift is significant because it gives every game developer access to the state-of-the-art graphics these engines provide without having a world-class 3D graphics programmer on staff. Since everyone now can more easily have good graphics, game developers are looking to other areas, such as AI, to differentiate themselves from their competitors.

FINITE BENEFITS

Increasingly complex games, players' ever higher demands, and the need to rise above the crowd are forcing game developers to start exploring new AI techniques. Fittingly, a few have turned to the research community for inspiration: Researchers coined the term artificial intelligence at an academic conference held at Dartmouth College in 1956. This gives the academic field of study almost a 20-year head start on using AI in games.

AI researchers first described the A* search algorithm in 1968, four years before *Pong*. However, while there is some overlap in problems explored and solutions used, research AI and game AI are not the same thing. Finite state machines form part of any comprehensive list of game AI techniques. Yet they are not considered part of the artificial intelligence field, belonging instead to computation theory. Following the A* search algorithm's path, game developers are starting to explore techniques from several AI research subfields, including automated planning and machine learning.

To date, most AI techniques used in games have been either reactive, as is the case with finite state machines, or scripted to use essentially the same strategy or set of behaviors repeatedly. Game AI has typically been unable to think multiple steps ahead—a capability necessary to generate interesting strategies on the fly such as feints and ambushes.

Automated planning has the potential to make this kind of reasoning possible in games. At the most basic level, the subfield of automated planning studies techniques for finding a plan that will achieve a goal.

A planning problem consists of a description of the current situation, a description of the goal situation, and a set of actions that might change a situation into a slightly different situation. A planning algorithm then finds a sequence of actions, called a *plan*, that, when executed in order, will transform the current situation into the goal situation.

Vivendi's *F.E.A.R.* FPS uses the goal-oriented action planning technique (GOAP) to rapidly generate short sequences of actions to achieve goals that can't be achieved with a single step. In effect, GOAP treats the problem of getting from the current situation to the goal situation as a path planning challenge. The current situation is the starting location, the goal situation is the target location, and actions move the character closer to or further from the goal. GOAP can use the same A* search routine for both path planning and finding plans to achieve goals.

Gamers and critics widely considered *F.E.A.R.* one of the best games of 2005 and almost every review singled out the AI opponents' cunning as a strong point. Hopefully, the success of this early attempt to bring automated

planning into games will encourage more developers to follow.

TEACHING THE MACHINE

A second subfield of research AI, *machine learning*, has started making inroads into games. It has the potential to let AI characters improve with experience and adapt to individual players.

The two machine learning techniques most commonly discussed in the games context are inductive and reinforcement learning. Lionhead's strategy game *Black & White* and its sequels use both techniques in combination.

In *Black & White*, the game gives each player a pet creature that plays the role of an AI-controlled ally. The player can teach the creature how to behave by rewarding good behavior with strokes and punishing bad behavior with slaps. The combination of inductive and reinforcement learning lets the creature generalize positive or negative responses to individual actions into a general set of guidelines for what actions should and shouldn't be taken.

Another example of machine learning in games, the Drivatar technology (http://research.microsoft.com/mlp/forza/), features prominently in Microsoft's car racing game *Forza Motorsports*. The technology lets the player train an AI-controlled driver to drive in his or her style. While machine learning can be—in one game developer's words—"scary voodoo," it has the potential to move game AI beyond preprogrammed behavior.

We can glimpse what the future holds for artificial intelligence in games by studying the freely available interactive drama *Façade* (www.interactivestory.net/). This title moves game AI beyond the "kill or be killed" mindset and into the realm of interpersonal, social interaction. The setting is a dinner party at the apartment of Grace and Trip, an attractive couple who you quickly learn are in the final stages of breaking up. The "game" plays out like an interactive soap opera where what you say and do will change the course of the story and the lives of Grace and Trip.

Videogames are often criticized for being too violent. However, as game AI becomes more sophisticated and lets game characters have deeper social interactions with the player, many more nonviolent games will appear, and these games will attract whole new audiences.

Michael van Lent is a research scientist at the ICT. Contact him at vanlent@ict.usc.edu.

Editors: Michael Macedonia, macedonia@computer.org; and Michael van Lent, vanlent@ict.usc.edu