9-1-2005

Deceiving Intelligence

Chris Pruett

Follow this and additional works at: http://commons.pacificu.edu/inter05

Recommended Citation

This Article is brought to you for free and open access by the Interface: The Journal of Education, Community and Values at CommonKnowledge. It has been accepted for inclusion in Volume 5 (2005) by an authorized administrator of CommonKnowledge. For more information, please contact CommonKnowledge@pacificu.edu.
Deceiving Intelligence

**Rights**

Terms of use for work posted in CommonKnowledge.

This article is available at CommonKnowledge: [http://commons.pacificu.edu/inter05/35](http://commons.pacificu.edu/inter05/35)
Deceiving Intelligence

Posted on September 1, 2005 by Editor

By Chris Pruett <c_prueitt@efn.org>

At my day job as a programmer at video game development studio Vicarious Visions, one of my common tasks is creating artificial intelligence systems. These systems typically control the enemies and obstacles that the player must overcome in order to progress to the end of the game. AI is an extremely wide academic field, but my day-to-day work rarely has much to do with actually creating intelligence artificially; rather, my goal is only to create the appearance of intelligence.

Academic AI study is mostly the study of systems that match patterns. The human brain is a big pattern matching machine, and we learn by training our brain to recognize new patterns, even in situations where the data at hand is incomplete or partially incorrect. Using software technologies such as neural networks, it is possible to build systems that are similar in structure to the brain but are far less complex. The problem with this approach is that learning is an aspect of human beings that is highly dependent on our surrounding environments; the brain can only learn new patterns in specific ways if new patterns are presented appropriately. Thus software models of the brain often have difficulty learning how to produce the correct response to a certain input; it is possible for neural network brains to grow up to be stupid.

However, AI in video games is an almost entirely different field. Academic research is applicable to a certain degree, but often technical limitations alone rule out the use of real AI systems in the video game world. Instead, game programmers like myself are charged with creating systems that fool the player into believing that their opponents are real people, or at the very least, that they are interacting with something more than simple software constructs. The one and only goal of the game AI programmer is to produce an experience that is fun for the player, and so in some respects my job is much easier than that of an academic AI researcher.

Rather than create something that can actually evaluate its environment and make a choice about the best course of action, it is often sufficient for game AI programmers to simply enumerate every possible scenario and tie it to a specific, predefined action. Almost all game AI operates on elementary Pavlovian stimulus-response devices; very rarely is there any code that
might be considered a model of a real thought process. For example, if the game design calls for a troll that lives under a bridge and eats the occasional passer-by, there is no need for the AI programmer to actually model the troll’s hunger level and subsequent decision process in software. A simple counter (1C; eat every 5th pedestrian 1D;) or random number generator (1C; for every pedestrian that walks by, let there be a 1 in 5 chance that the troll will eat them 1D;) will suffice. Since the troll is such a simple device, adding more complexity to him will not make the game any more fun, and thus there is no need to spend more time to develop a more realistic system.

As a game AI programmer, I can get away with all sorts of trickery that creates the illusion of intelligence. My job is simply to make sure that my AI system is not too repetitive (predictable patterns are boring), is sufficiently life-like (a task which often falls on the animator rather than the programmer), and helps to construct an environment that is fun for the player to interact with. Most of what I do amounts to software smoke and mirrors, but as a game AI programmer, deception is part of my job description.

However, as games become more complex my job description is going to change. The majority of technical innovation in the game industry is centered around improving visual fidelity, and while a lot of work is done in producing ever-prettier pictures, it has become clear that convincing movement, animation, and behavior of computer-controlled characters is quickly becoming a requirement. Creating convincing behavior can be accomplished given the industry’s tried-and-true methods of fooling the player, but the level of complexity is quickly rising. At some point, it may actually become easier to create smart software than to try to cover every possible game scenario.

For example, let us return to the bridge troll. Say we give the player the freedom to cross the bridge safely any way he can. The player can choose to sneak across the bridge, jump down into the riverbed and fight the troll, or perhaps travel upstream, blow up a dam, and flush the troll away. This is a pretty open-ended scenario for the player, and the troll must react believably to whatever the player tries to do. In the past, game developers have simplified such situations by creating only one or two pre-determined solutions and simply forcing the player to work out what was the right thing to do. But lately games like Grand Theft Auto 3 have shown us that allowing the player to solve problems on their own given a wide variety of tools and a highly interactive environment can be extremely enjoyable and a much more rewarding play experience.

Even this example is somewhat simplified. If the player decides to fight the troll, how will the troll defend itself? A club? Perhaps a rapier? Can we have trolls that are skilled in martial arts in addition to trolls who’s main maneuver is to bonk opponents over the head with a heavy stick? If the player floods the riverbed, can the troll swim? If so, where does he swim to? Is he angry at the player when he reaches shore? How does this affect his behavior?

Clearly, giving the player total freedom would require us to create artificial intelligence systems that are closer to real intelligence than just a set of responses to specific stimuli. Given the state
of the industry at the moment, the troll problem described above would probably be solved with a very general purpose stimulus-response system: hints on how to react to being in water, or how to wield a weapon to brain an opponent. At the lowest level, the decision process would probably still boil down to a list of deterministic responses to predefined sensory input. In other words, our troll would have about as much actual intelligence as the animatronic Pirates of the Caribbean marauders.

And at the moment, Pirates of the Caribbean is good enough. The game industry has yet to break out of its quest for visual perfection, but eventually the graphics arms race will slow and programmers like myself will need to find new ways of creating seemingly intelligent game characters. As game environments become more dynamic and less deterministic (both through innovative game design and the application of better technologies, such as physics simulation), the AI interacting with the player will also need to grow and adapt. The work that academia is producing now will likely be increasingly relevant to the game industry in the near future. However, even as new technologies are integrated into the game industry, the role of AI in games as an instigator of enjoyment will remain the same.

This entry was posted in Uncategorized by Editor. Bookmark the permalink [http://bcis.pacificu.edu/interface/?p=3149].