AI.implant™ for Games: great game AI has a name

Introduction

Artificial intelligence (AI) has been a part of video gaming for as long as there have been video games. Before the advent of two–player or multiplayer games, video game players had to play against the machine (computer or game console) or not play at all. To create opponents and other characters worthy of their names, game developers have had to develop from scratch the logic and routines to drive non–playing characters (NPCs) that could think for themselves and react in real time to the player's movements. These rather primitive routines were the basis of all video games for the first 20 years of their existence.

The importance of quality game AI should not be understated as it has a direct impact on a game's playability. In many cases, AI is as important as, if not more than, the game's visuals, and can make or break a game's success. In fact, game players will often judge a game by the quality of the NPCs, specifically their ability to interact with the main game character in a realistic manner, whether that means fighting with the main character or working with him to accomplish a mission.

While 3D graphics have evolved tremendously over the years with the development of standardized libraries like OpenGL and Direct3D, the same could not be said of artificial intelligence. Until quite recently, developers continued to write their AI routines from scratch — a very labor intensive and time–consuming undertaking. In addition, the AI routines that were developed were not that smart and tended to be very context–specific, too specific in fact to reuse and port across game projects, an important consideration in this time of rising costs, limited budgets and strict development timeframes.

Though the quality and depth of AI have improved immensely in intervening years and NPCs have in some cases acquired realistic and lifelike behavior, the widespread use of good game AI, let alone great game AI, remains problematic. While games such as *Half–Life 2*, *Grand Theft Auto: Vice City* and *The Sims* have raised the bar, a generalized solution that enables all game developers to achieve great AI, while meeting their budget constraints and development timeframes, has until recently remained elusive.

Introducing AI.implant for Games

Realizing that there was an opportunity to create a software foundation that would give game developers a head start in their game's AI development, and provide the added benefit of portability, BGT BioGraphic Technologies under-

took the development of AI.implant™ for Games (AI.implant), a product akin in many ways to an OpenGL for artificial intelligence.

Led by Dr. Paul Kruszewski, a pioneer in the field of procedural animation and simulation, the team of engineers at BGT set about to define an AI vocabulary and grammar, thus creating a solid basis for great game AI. The BGT team would go on to successfully accomplish their mission, launching AI.implant for Games at the 2002 Game Developers Conference.

Strong feature set

By using AI.implant, developers gain access to an advanced, well–defined and battle–tested API, which enables them to quickly add immersive AI to their games, resulting in the most realistic, interactive games being made today. AI.implant provides a number of core features, including intelligent navigation and decision–making, that enable game developers to take their game AI to new heights. As well, it offers additional features not traditionally associated with AI, like animation control which can drive the game's animation engine to create cinema quality visuals during runtime

Pipeline integration

AI.implant was developed with a keen understanding of the pipeline and workflows inherent in game development. Indeed, recognizing the trend to empower artists and level editors to not only create game assets but actually implement some initial game functionality (e.g., Havok Reactor for 3ds max), AI.implant was designed to be tightly integrated with game authoring tools such as Discreet's 3ds max™ and Alias' Maya®. AI.implant leverages both packages' intuitive approaches to authoring game content, making it easy for artists and level editors to access such traditional AI features as path finding (navigation) and decision–making.

Once AI mark–up and behaviors have been put in place and tuned to the desire (or ability) of the artist or level editor, AI–related information can be exported as an AI.implant ACX file to be read by the game engine and the AI.implant solver. At that point, additional, more advanced game AI function-

Sample scenario

We will consider the various features of AI.implant for Games within the context of a simple scenario.

In this scenario, the main character must enter and navigate through a secret compound littered with boxes and architectural columns, and defended by a number of computer–controlled (autonomous) guards.

These guards have been instructed to defend the compound against any intruder, to take action when necessary, including firing upon the intruder, to insure their own survival by taking cover, and to alert other guards if surprised or overwhelmed.

ality, whether through scripting, or C++ code, can then be implemented by the designated AI programmer. This rapid prototyping and validation of AI functionality dramatically speeds up the game development process.

Rapid world mark–up

AI.implant allows users to quickly tag objects and world elements with AI behaviors to make them autonomous or non–autonomous characters, and then preview those same behaviors on–the–fly from directly within the level editor (3ds max or Maya). Consequently, artists and level editors can achieve game–like interactivity and functionality in real time without the need to export assets and information out to the game engine.

Navigation / path finding

One of the most basic activities in developing game AI is determining how and under what circumstances NPCs travel around the game level. Before autonomous game characters can move around effectively, they need to know which areas of the level can be visited and which paths they can take to visit them. Barriers translate the game level into terms that the autonomous characters can understand.

Autonomous character
A character that has a brain controlled by AI.implant and has some ability to improvise its actions and interact with other objects in the world.

Non–autonomous character
A character that is not controlled by AI.implant but can still move and interact with other objects in the world.

AI solver
The container for managing the autonomous and non–autonomous characters and other objects in the world. The AI solver can be created in either 2D for simple terrains or 3D to allow characters complete freedom of movement.

Overall game level mark–up with AI.implant can be performed from directly within the level editor by simply clicking on surfaces and assigning appropriate AI properties using the AI.implant menu. In the case of our sample scenario, the compound floor, walls, and columns as well as boxes would be marked up as non–autonomous objects. While they do not have behaviors per se, except in the case of sliding doors, and cannot think for themselves, they interact with autonomous characters (the guards) by providing information about their size, position, etc.

Special areas such as cover points, useful for the guards to hide from or to attack the main player from, and retreat points, areas for the guards to head for when running away, can be designated just as easily. In our sample scenario,

we designate many of the boxes as cover points, while areas at the back of the compound or near remote doors can be considered retreat points.

Once the barriers are in place, we describe the means which autonomous characters can use to make their way around. To act realistically, NPCs must be able to move about the game level without banging into walls or other objects (boxes, columns, doors, etc.). AI.implant makes the generation of these paths, otherwise known as waypoint networks, easy. In fact, AI.implant offers the ability to generate waypoint networks across an existing game level from within the level editor with the simple click of a mouse.

Once the waypoint network markup is in place, characters can begin moving around the game level. In our sample scenario, guards will move about the compound, patrolling it in case intruders should try to infiltrate.

Decision making

Once the relevant areas and means of moving throughout the game level have been defined, it becomes necessary to determine when such movement or interactions with other characters should occur. In our scenario, a single guard will normally patrol the compound in a regular manner, making the rounds on a clockwork–like basis to ensure that all is in order. When an intruder (the main character) is detected, the guard will react. After initially firing upon the intruder, the guard will retreat to the back of the compound to sound the alarm, and invite his colleagues to join him in defending the base. The group of guards will now alternate between now openly attacking the intruder and hiding behind cover points, from where they will continue firing upon the intruder, so as to minimize their own chances of getting shot. If they conclude the intruder is likely to overpower or kill them, they will attempt to run away.

This autonomous character is determined to protect The Company's valuable intellectual property with his life.

In our sample scenario, the guards' behaviors have been defined by the developer, using a combination of behaviors, some of which are supplied with AI.implant and others that are game–specific and thus written by the developer. These abilities and behaviors can be viewed as building blocks, which can be used individually or combined together with others to create even more advanced behaviors and interactions within the game. In our sample scenario, we are making use of the Seek, Strafe, Flee and Hide behaviors with the order and intensity in which they occur defined by the game developer.

Sense. Think. Do.

Once the guards' behaviors are in place, we must enable the character to interact with the world and make decisions based on those interactions. The AI.implant decision model is based on a series of rules, which are in turn driven by environmental stimuli during runtime.

'**Sense**' refers to the general ability of characters to sense their environment. In our example, the guard can "see" intruders in the immediate vicinity, can hear footsteps nearby, and can smell the burning odor of a recently fired gun, or the stench of caked blood on a deceased colleague. These sensors, in turn, trigger reactions. For example, a junior guard seeing his leader shot down by the player character may consider running away to a retreat point, or may choose to take cover before changing to sniper mode.

Once a sensor has been triggered, the character must '**Think**' about the situation at hand, then choose a course of action. The actual decision model that the NPC uses can be based on a decision tree, a finite state machine (FSM), scripting or a combination of any of the three. For example, complex decisions may require triggering scripts from decision trees located themselves within state machines. AI.implant provides a visual interface that allows you to easily author, modify and debug the decision model. In our example, the guard will consider whether it should stay put and keep firing at the intruder, run for a cover point and keep shooting, or sound the alarm and retreat to the back of the compound.

Having chosen a course of action, the character must now '**Do**'. Working with the game's animation engine, AI.implant selects the appropriate animation based on the course of action and the available bank of animations. If the guard character does decide to run for cover, AI.implant will tell the animation engine to play the walk, run, brake and crouch animations, in successive order as the guard leaves for and gets to his destination. It can also provide the animation engine with additional information such as blending and scaling to ensure that feet don't slide, that transitions among animations are smooth and don't "pop". In addition, AI.implant also offers the ability to prioritize behaviors as well as change their intensities. For example, it may be more important for the guard, who is being fired upon by the main character, to seek cover rather than return fire. These priorities and intensities can also be defined, previewed and fine–tuned from within 3ds max or Maya.

Animation control

AI.implant goes beyond simply providing the game's AI functions to actually enhancing the overall look and feel of the game. By leveraging AI.implant's built–in animation control system, the developer can actually drive the game's character animation system for NPCs as well as the main player character, resulting in incredibly fluid, film–quality animation that further enhances the onscreen appearance of the game's characters. AI.implant intelligently feeds the game's animation engine, providing the information necessary to select, scale and blend animation clips. These can be based on movement–related information such as the character's speed or whether he is turning, or they can be based on state information like whether the character has drawn a weapon or is dying. As a result, characters transition smoothly among their various animations. Indeed, characters can change cleanly from an idle to a walk to a run and then a braking animation, or pull out and fire a weapon while running, with no visible "popping" — the erratic, unrealistic movement that might otherwise occur. This very powerful feature is again made available to the level editor or developer from within the intuitive 3ds max or Maya interface.

AI.implant also includes a sophisticated animation marker system that allows the developer to trigger external events. For example, it can instruct the game engine to play a bang sound when the character fires a gun or match the character's feet to steps, ensuring that they always touch the ground. Rather than requiring the developer to set them up elsewhere in the game development pipeline, AI.implant enables the developer to implement these advanced animation functions as he is doing the other AI–driven animation routines.

Other features

Modular design

Ai.implant features a modular design that enables game developers to extend and enhance the functionality of AI.implant to create new types of AI interaction. The AI.implant architecture allows for the creation of custom AI objects, including commands, sensors, and behaviors, and makes it easy to push them through the pipeline. The game developer can override, replace or extend the supplied programming classes, or simply cut down on game AI functionality with the goal of reducing binary size or minimizing memory usage. For example, game developers can use the collision detection functionality built into AI.implant or swap it out in favor of their own or that of other middleware (e.g., Havok, Renderware, Gamebryo).

Cheaper than Dobermans and twice as tough; with AI.implant for Games telling them what to do, these guards have got the place covered.

"Halt! Who goes there!?!"

Data sharing

AI.implant has the ability to share data with the game engine and other middleware products, enabling the developer to save on runtime resources such as memory and CPU time, as well as reduce duplication of effort during game development. As an example, the barrier information generated when marking up levels for the Havok physics SDK can be used by AI.implant.

Broad platform support

AI.implant features support for all major game console platforms (PlayStation®2, Xbox™ and GAMECUBE®) as well as PC–based computing platforms (Windows and Linux). As such it offers developers the ability to develop once on one platform and deploy their code on all other supported platforms. While the AI.implant solver library has been ported to each underlying game platform, the ASCII–text based ACX file format is platform independent, making it easy and pain–free to develop AI.implant–based games on multiple platforms.

This autonomous character thought he had a well–developed sense of fashion. In the end, his AI.implant–driven colleagues thought otherwise.

Integration with 3rd party middleware

AI.implant can be easily integrated with other third–party middleware including popular graphics, physics and networking engines, among others. BGT has relationships with such leading companies as Criterion Software (RenderWare), Numerical Design (Gamebryo), Havok (Havok 2) and Quazal (Net–Z). As well, BGT has first–hand experience with a number of these products, having previously performed various integrations with these companies' game engines. Consequently, when developers integrate AI.implant with any of these products, they can look forward to prompt technical support, concise answers and a minimum of finger pointing. As an example, in the case of the Midway Games' action/adventure title *ESPionage*, AI.implant was successfully integrated with the Havok 2 physics SDK. The Institute for Creative Technologies at the University of Southern California has integrated AI.implant with Unreal Engine 2003 for a military training application.

Battle–tested and supported

Now at version 1.6.1, AI.implant has also been field–proven by a number of respected game developers. Foremost among them is Midway Games of Chicago, a top–10 games publisher which integrated AI.implant into its ESPionage title for the PlayStation®2, Xbox and GAMECUBE platforms. Working with the AI.implant technical support team, Midway was able to add advanced AI functionality to its game on time and on budget.

Conclusion

Today's games are no longer about just the latest 3D graphics. What makes the difference between a triple–A title and plain old shelfware is the quality of the game play. Games like *Half–Life 2*, *GTA: Vice City* and *The Sims* have raised the bar, changing customer expectations forever. Sub–par game AI with flaccid lifeless opponents is no longer an option. In this time of tight budgets and project timeframes, only AI.implant makes it so easy to develop, test and deploy great game AI.

By integrating tightly with the game development pipeline, and with leading tools such as 3ds max and Maya, AI.implant simplifies the process of adding AI functionality to a game. Implementing pathfinding and decision making is easy with AI.implant, enabling game developers to rapidly move on to designing and tweaking rich character interactions. And by going beyond core AI functionality, and providing a sophisticated animation control system, AI.implant makes games look better too. AI.implant offers all of this powerful functionality and more, while enabling developers to implement advanced game AI, on time and under budget.

With AI.implant for Games, great game AI finally has a name.

Bio|Graphic Technologies