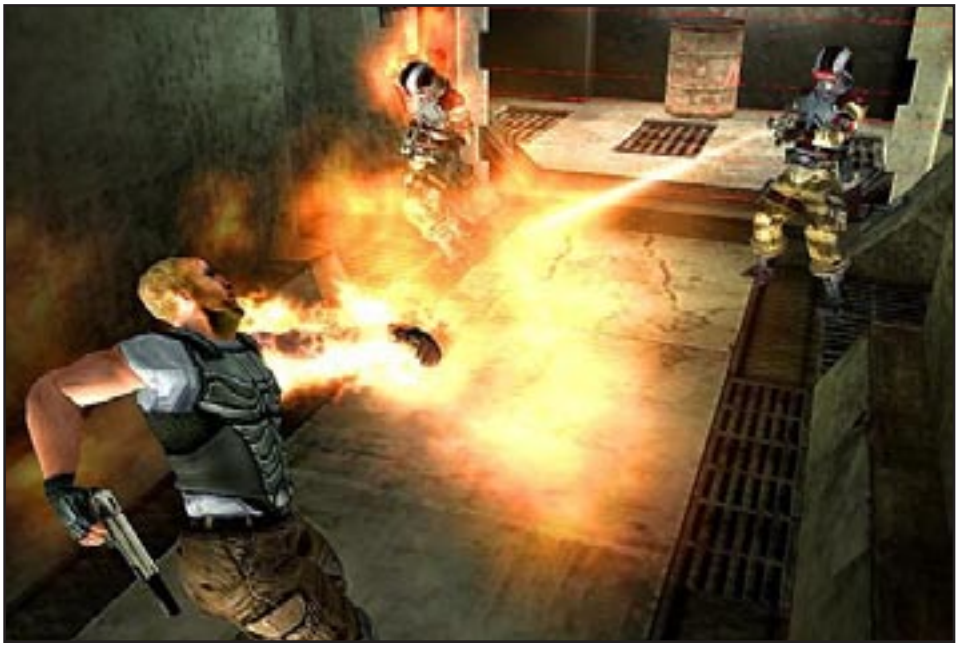
AI.implant™ for Games: case study of Midway Games

A leading developer/publisher of video games, Midway Games is a pioneer of the electronic entertainment industry. Starting with pinball machines, Midway then branched out into video games where it revolutionized the market which such breakthrough titles as Pong, Defender and Spy Hunter. Following in the footsteps of such blockbuster titles as Ready 2 Rumble Boxing, NFL Blitz and the infamous Mortal Kombat series, Midway decided to develop new game properties featuring brand new characters, engaging new storylines, and original game play.

Originally conceived in 2001, PSI–Ops: The Mindgate Conspiracy (previously known as ESPionage) was to become the first in a new line of such games. Combining the best elements of the action adventure genre with the fast pace of first person shooter games, PSI–Ops invited characters to assume the character of Nick Scryer, an elite American soldier formed in the covert Mindgate Program, in his efforts to combat a terrorist organization known as The Movement. Through the use of such PSI powers as telekinesis, remote viewing and mind control, PSI–Ops would deliver on Midway's goal of bringing innovative game play to the mix.

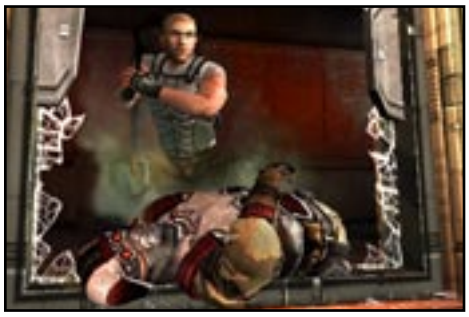


PSI–Ops hero Nick Scryer blasts two AI.implant guards working for The Movement.

In creating PSI–Ops, the design team at Midway sought to build an experience over a series of game levels, with many of them featuring action over multiple floors. Via Nick's character, players would be required to weave through complex rooms, go up stairs and down ladders, and even jump across holes freshly

made by explosions. In order to pursue the player, The Movement's soldiers—computer–controlled characters based on artificial intelligence programming—would need to be able to do the same.

To navigate their environment, these soldiers would first need to understand how it was laid out. They would also need to know where they could and could not go, and they would need the ability to determine the best, most efficient path that would enable them to achieve their goal of stopping Nick Scryer. They would have to do so in the most realistic manner possible so as to maintain the suspension of disbelief so critical in both engaging and keeping players immersed in a game.



Nick Scryer disposes of an AI.implant guard by throwing him through a window.

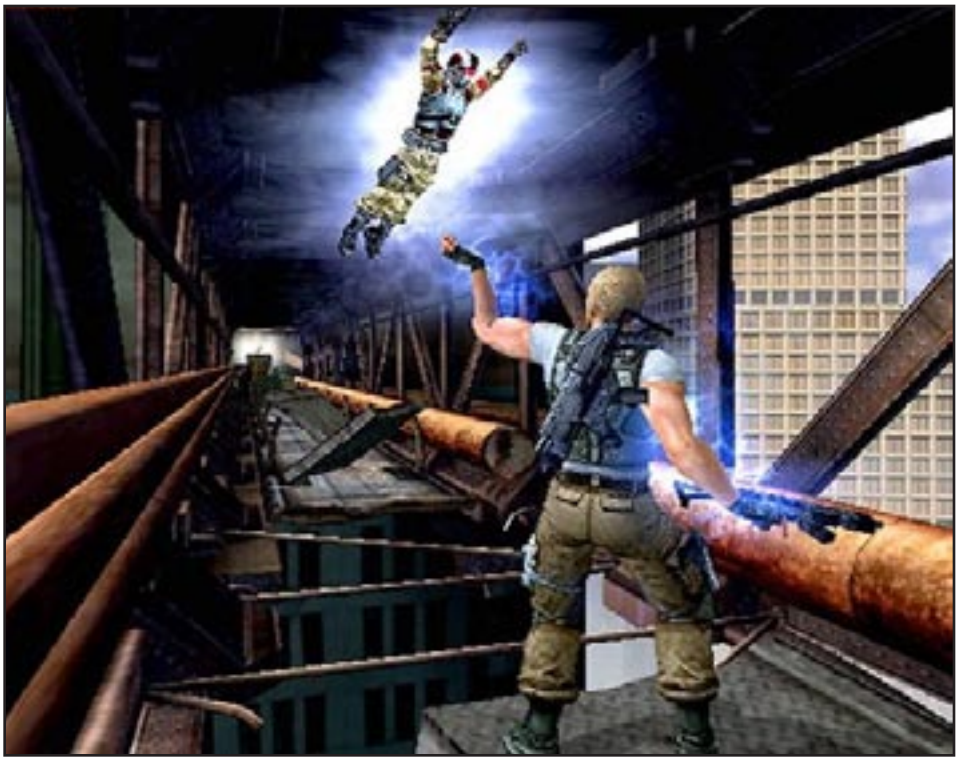
In addition to enabling The Movement's foot soldiers to do their job well, Midway's development team needed a solution that would allow them to author the pathfinding information for the game levels relatively quickly to meet the aggressive development schedule, but to also handle the inevitable design changes as consumer and tester feedback was integrated into the game. As a result, Midway turned to BioGraphic Technologies' AI.implant™ for Games, the award–winning, artificial intelligence–based 3D tool and runtime engine to use in conjunction with its Alias Maya®–based pipeline .

The need for good pathfinding

In putting in place the infrastructure necessary to enable non–player characters (NPCs) to interact with main player characters and the game world in a realistic fashion, game developers turn to a concept called pathfinding. Broadly defined, the goal of pathfinding is to get the computer–controlled character where he wants to go, in the most efficient way possible, while avoiding obstacles that may slow or stop their progress.

While apparently an innocuous technical element—done right, pathfinding is never really noticed by the player—pathfinding can actually make or break a game; it directly affects the user's experience and perception during game play. Bad pathfinding is the bane of many games as it destroys the suspension of disbelief so critical to a successful game experience and, aside from jumpy graphics, is generally considered to ruin the game. Examples of bad pathfinding include NPCs getting stuck in and unable to get themselves out of corners; not reacting properly when confronting the main player (e.g., they see the character but don't appear to react); or non–natural robotic movement whether it be paths that are too straight—characters neglecting to cut corners—or unrealistic movement like walk or run cycles that are out of context.

Though at first simple, the concept of a best path itself changes constantly as the NPC must not only calculate—while moving—the appropriate path among multiple options through doorways, corridors, and wide open spaces strewn with random objects, but must do so while taking into account the need to avoid dynamic obstacles such as vehicles or other NPCs. Additional considerations, specifically in the case of PSI–Ops, included the need for soldiers patrolling a given area to choose a path that enabled them to cover/survey the most territory or to choose a path that provided the most coverage from the main player character—and his bullets and PSI powers in the case of PSI–Ops—to ensure its safety and survival.



The Movement's AI.implant–powered guard is help up by Nick Scryer.

The pathfinding status quo

Pathfinding itself is composed of two elements: a representation of the world marked up in a format that an NPC can understand; and the basic logic—the pathfinding algorithm that tells the NPC how to use that representation to navigate the world—plus additional information about how to deal with dynamic obstacles while respecting the NPC's strategic imperatives.

World mark–up: manual or automated

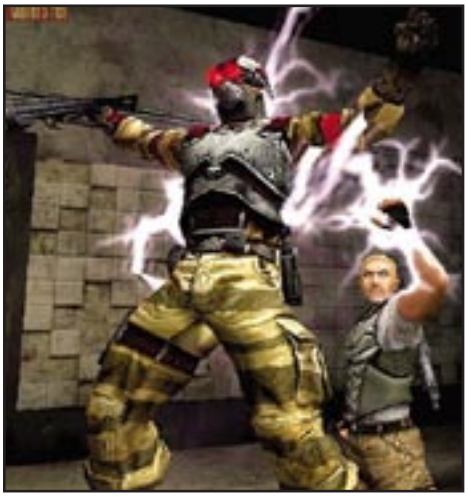
The need to perform world mark–up has led to two general approaches. Under the first approach, mark–up is performed by hand by the person assigned to perform the level editing function. It entails manually creating navigation paths separate from the underlying geometry of the level, or selecting existing geometry and tagging it with the information required by the pathfinding algorithm to do its job. While this approach can be effective for simple levels, it doesn't scale well. Indeed, it tends to break down in the face of the complex levels that characterize the latest generation of action adventure games such as PSI–Ops. As well, the manual mark–up approach tends to bog down in the face of the multiple level revisions that occur throughout the development process almost continuously until the game is locked down.

The alternative to manual markup has been the development of custom tools. In response to the reality of ever more complex game levels and the constant revision of them during the game's production, many development studios have developed their own tools, which automate the process of producing pathfinding mark–up. While these have proved to be an effective solution for some game development efforts, they have not always been the most efficient. The reasons for this are generally two–fold: one, in order to develop and maintain these tools, programming resources and funds are diverted away from the

core game development, an expensive effort that leaves other programmers to pull up the slack and reduces funds available for more customer–critical areas of the game like artwork, audio and storyline. Second, these tools—often skunk works efforts done on programmer initiative—usually take the form of separate stand–alone applications, which may not be easily understood by non–technical, more artistically inclined level editor personnel. As a result, the very people they are designed for are often unable to use the tools, implying either unplanned or unbudgeted training time or that expensive programmers perform the level–editing functions themselves. Furthermore, from the start these tools need to be designed and programmed so that the output they generate works across the game's various deployment platforms, implying additional engineering effort.

A combination of resource usage—map size and complexity, the number of NPCs, the presence of dynamic objects—and the technical constraints of the underlying deployment platform—CPU clock speed, memory heap, among others—determine in large part the technical approaches to pathfinding that can be considered. Among the most important considerations of all is that the calculation of character AI not cause the graphics frame rate to drop. Since so much of the processing time and resources of a game console are taken up already by graphics (models and textures), physics and game logic, the effective AI solution also needs to be lightweight, both in terms of the runtime code—the AI mark–up and logic—required to perform the pathfinding routines and the actual library size, since space is at a premium on console game discs.

Another AI.implant guard is taken out. Nick Scryer: 5. The Movement: 0.



How AI.implant helped

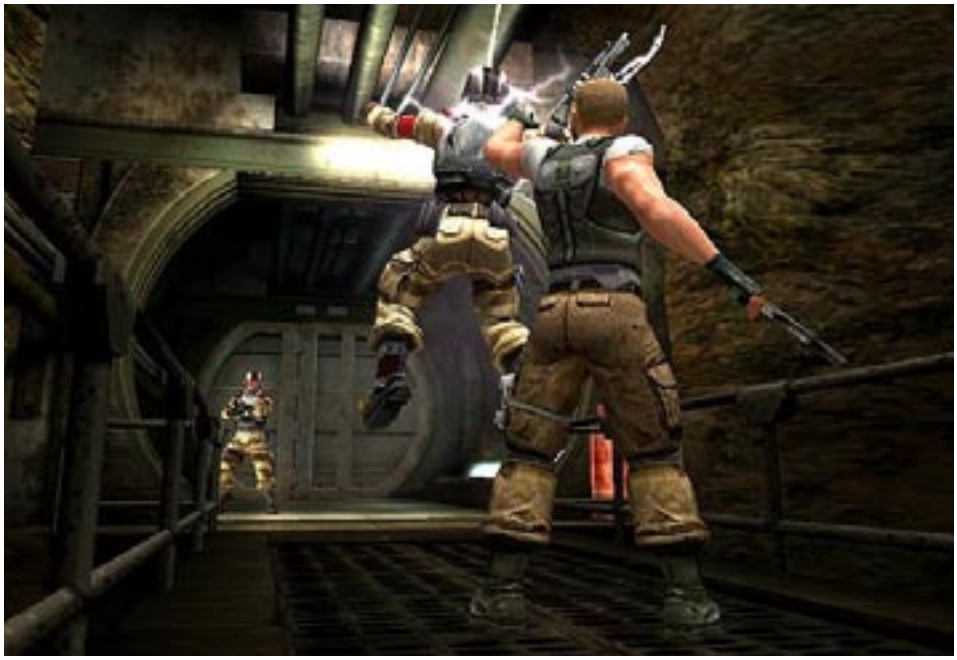
Faced with the technical challenge inherent in producing fully marked–up levels on a timely basis in the context of constant change, and not wanting to go to the trouble of building expensive in–house tools or having to train their employees on them, Midway turned to AI.implant for Games. Featuring both an authoring plug–in for the popular Maya art package on which Midway had standardized for the production of PSI–Ops, and a runtime engine for the PlayStation 2, Xbox, and GameCube* game consoles, AI.implant offered Midway the benefits of a cross–platform tool integrated into its development pipeline without the associated cost of developing and maintaining the tool in–house, thus freeing artists and programmers to do what they do best.

Offering both the ability to mark up the game world and create the logic to tell the NPC how to navigate based on that information, AI.implant provided the PSI–Ops team with a complete package for developing the great pathfinding it needed to both meet and exceed its customers' expectations—through fluid realistic movement while maintaining frame rates on all deployment platforms and the NPCs' ability to follow Nick Scryer anywhere he went.

After building the levels in Maya, Midway's team of level editors turned to AI.implant to create the waypoint networks. Once a given level had been marked up, Midway then used AI.implant to give The Movement's digital soldiers the ability to avoid obstacles—props, Nick Scryer and other characters—dynamically. Again using the AI.implant menu in Maya, Midway's game developers were able to build up the core behaviors—including avoidance of

** In the case of PSI–Ops, this meant the PlayStation®2, Xbox®, and GameCube® platforms. Prior to shipping PSI–Ops, Midway made a business decision to drop support for the GameCube platform. The game subsequently shipped as planned on the PlayStation 2 and Xbox platforms.*

dynamic and static obstacles, the ability to seek to a target (Nick Scryer), and the use of cover points—that would enable the soldiers to effectively guard the game levels. The NPCs also had the ability, on hearing the sounding of the alarm, to flock to the point where Nick Scryer was spotted. These abilities were not hardwired to the scene; the underlying geometry or design of the level could change without impacting the ability of the digital characters to react properly. From a creative standpoint, this meant that Midway's game designers could focus their efforts on the game design, knowing full well that AI.implant would be able to handle their level changes efficiently and rapidly when the time came to update the mark–up.



Nick Scryer sends a signal to the other AI.implant guard that he is next on the list.

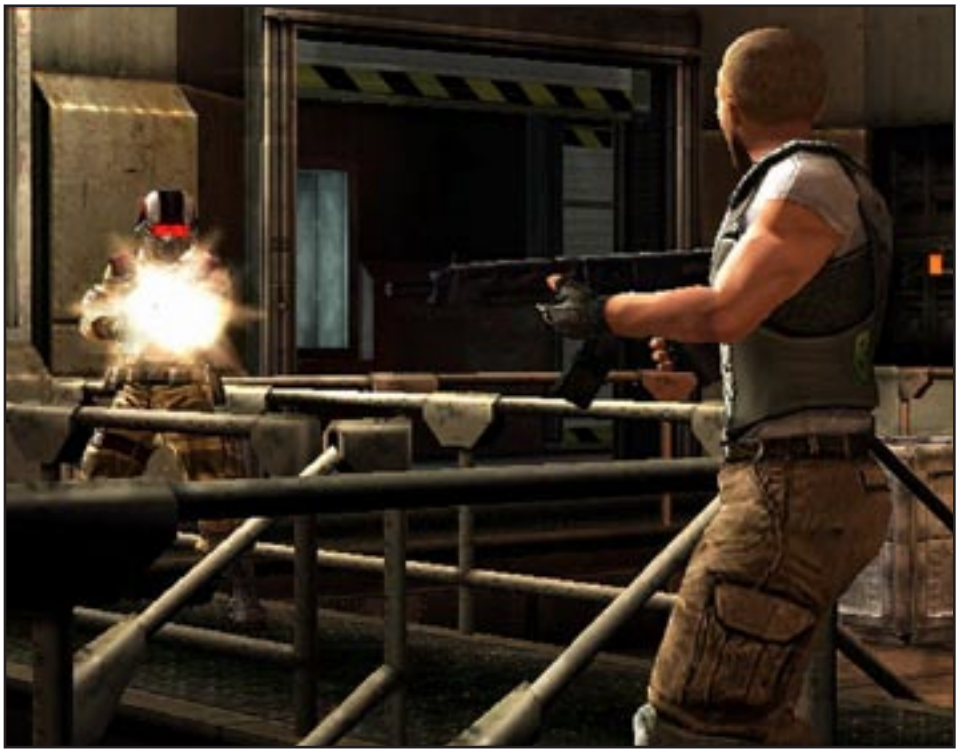
Due to AI.implant's integration with Maya, the PSI–Ops level design team was then able to play back these AI behaviors within Maya in real–time. NPCs could be placed at various points in the level to test for impact on game play such as the use of cover points and the size of the sections of the level that guards were expected to cover. This integration between AI.implant and Maya was a major factor in the enhanced productivity of the team; instead of having to export out the pathing mark–up and logic to the production game engine each time they wanted to review their work, they could preview it directly on their desktop. This not only saved time for the level editors but enabled the core programmers to focus on their work as well.

Result

By using AI.implant to achieve its pathfinding needs, Midway was able to successfully deliver on its goal of delivering compelling realistic supporting characters in a highly original storyline. Through its integration with the Maya workflow, AI.implant enabled the PSI–Ops production team to focus its efforts on the rich game concept and not on building or maintaining expensive development

tools. AI.implant also enabled Midway to achieve its goal of a game that looked and performed equally well on both the PlayStation 2 and the Xbox. In addition to a lightweight runtime library and efficient pathing markup, AI.implant provided the ability to effectively scale the underlying pathing information. Finally, through its support for meta–networks—a network of networks, AI.implant allowed Midway's level editors to divide the game levels into sub–networks, saving precious console memory and CPU power that could be used elsewhere to further differentiate PSI–Ops.

"Because we were developing original IP, we didn't have a bank of legacy assets and tools we could call upon to reduce our development risk", said Jason Blochowiak, Lead Programmer, PSI–Ops: The Mindgate Conspiracy. "AI.implant not only gave us the confidence we needed to successfully complete the PSI–Ops project, it gave us the authoring ease–of–use and runtime performance we needed to achieve our goal of delivering original and engaging game play for our customers."



A hostile AI.implant guard finally gets the best of Nick Scryer.

Bio|Graphic Technologies

Epilogue

Soon after Midway shipped PSI–Ops, BioGraphic Technologies introduced a second approach to pathfinding, based on navigation meshes (NavMeshes for short). The transition to a navmesh architecture has enabled BGT to build additional features which strongly benefit the creative freedom of game designers, including support for fully dynamic pathfinding, the ability to embed information in the mesh, sharing of geometry, and the addition of new tools for AI authoring and debugging.

Starting with version 2.5 of AI.implant, launched in September 2004, all pathfinding is now dynamic. In addition to improved support for dynamic obstacle avoidance, AI.implant now offers developers the ability to create fully constructive and destructive worlds. With the capacity to turn navmeshes on (e.g., a hole is blown in a wall and a new path opens up) or off (e.g., an alleyway is blocked by rubble from an exploding building), this new version opens up a world of creative options for level designers.

In designing this new release of AI.implant, BGT's engineers also added the ability to embed information in the underlying mesh. Known as 'blind data', this user–defined information can be specified on a per–vertex or per–cell basis, opening up a whole world of possibilities. This information can be subsequently queried by a NPC to both influence its pathfinding and to achieve other forms of spatial reasoning. For example, a series of cells can be tagged as a 'sidewalk' and the pathfinding algorithm instructed to keep the character on the sidewalk. Alternatively, a cell could be tagged as a weapons cache, and the character programmed to load up on weapons whenever near such a cache (and short on ammunition.)

In an additional nod to performance, AI.implant v2.5 is now able to use the underlying level geometry previously created for physics (collisions) or rendering purposes negating the need to mark up the level separately. This results in significant savings of memory and improved overall performance at runtime.

Finally, BGT also added new tools for both authoring AI and testing. The first tool, AI.DE™ is a development environment specifically designed for authoring AI. It enables programmers to work with AI.implant for the first time without the need for a separate seat of Maya or 3ds max. Presenting the same familiar interface of the AI.explorer interface used in Maya, AI.DE offers the ability to preview AI behaviors locally using an OpenGL–based renderer; in addition, it offers the ability to write, test and debug scripts written with the AI.scriptT™ language that was also launched with AI.DE.

The addition of Arena™, a runtime demo game engine, enables game developers to now see the performance of their AI on the target platform (PlayStation 2, Xbox, GameCube, Windows, Linux) without the need to export out to and integrate with the production game engine. Through an integration with AI.DE via TCP/IP, it is possible to connect to Arena running on the target platform, pause it, export out the AI state at that moment and open it in AI.DE; the AI can then be modified and sent back to the console to test the changes. The time saved from not having to roundtrip to the production engine and the tight integration between AI.DE and Arena can be spent on additional iterations that improve the overall playability of the game's NPCs.