SPIR.OPS
artificial intelligence

General Presentation

GENERAL PRESENTATION

I] What does "SpirOps" means?

"SpirOps" is a contraction of the Latin words "Spiritus Operationes", which literally means "the operations of the mind".

II] What is SpirOps?

SpirOps is a development pack which brings Behavioral Artificial Intelligence (B.A.C) to the whole development team's reach; what was only a handful expert's possibility is now accessible to anyone.

SpirOps works like an interface between the dark side of A.I and the creators.

Although numerous, here are some examples of common applications of Behavioral Artificial Intelligence: In a strategy game your players will be able to confront virtual opponents; you will also have the possibility to give them independent military units.

- In a war game (FPS) your players will be able to be part of battalions composed in part of virtual soldiers and fight against such a battalion.
- In an adventure game you will be able to immerse your players in a world inhabited by all sorts of creatures and autonomous monsters having their own life. Or else (which is still very rare) make the general scenario evolve depending on the player's actions.
- In a massively multiplayer game, you will be able to create entire virtual populations, doing every day's work (peasants, bakers, millers, pastors, watchmen, and so on …) while the players will take on roles like adventurers and heroes.

We will later see that there are many other domains where B.A.C can be used and some unexpected like dynamic music generation and adaptive character animation.

SpirOps is a kit destined to "real time" applications (essentially video games), and may also be used in applications which are not, like computer graphics applications (Maya – 3DSMax – XSI).

III] What is SpirOps composed of?

The kit is composed of three major parts:

- The documentation and the tutorials describing the methodology characteristic of SpirOps.
- A graphical editor (under Windows 9x ® or more)
- A real time kernel (multi-platform).

SpirOps has a totally innovative way of handling A.I. The documentation and tutorials will help you structure your thoughts and let you build you're A.I. Step by step.

SpirOps includes a graphical editor which is the heart of the creative part. As we will see below, it is a "cross-competence" editor. This editor is also a debugger permitting to test and correct your creations.

As for the kernel, it is evidently the essential part in the execution of you A.I. It will have to be integrated in your application. It as been written in native language and optimized for real time; it thus consumes very few processing time and memory and can be used as well on a next generation console than a Mobile phone.

IV] What platforms does it work on?

The development environment works on a Windows compatible system (editor and debugger) while your applications will have to work on a platform capable of executing native code. SpirOps is therefore multi-platform. It works actually on all consoles and computers on the market as well as on all PDAs and Smartphones.

V] To whom is it destined and at which step of the production should it be used?

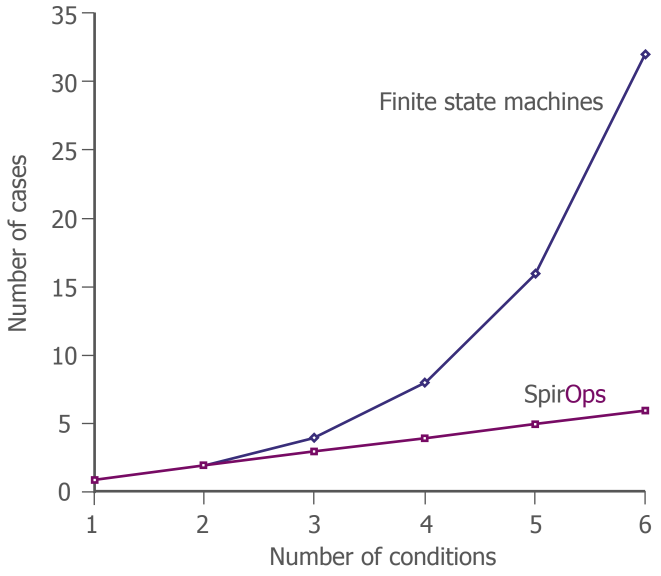
A major feature of SpirOps is that it is transversal competence wise. The editor is a centralized tool useful to the whole team (game designer – programming engineer – level designer).

Thanks to SpirOps, you will be able to structure your future game engine from the beginning even before any line of code had been written, thus avoiding many disagreements. As we will see later the intelligence of a creature is indeed directly linked to the quality of its perceptions. Therefore a creature which sees and hears nothing will be completely dumb even with the best A.I. Programmer ever. To use SpirOps from the beginning of the production will let you lay down all the essential data you will need to integrate in your game engine, which will significantly ease your work in the future.

VI] What are the key features of SpirOps?

1) A new methodology allowing a constant non-exponential complexity.

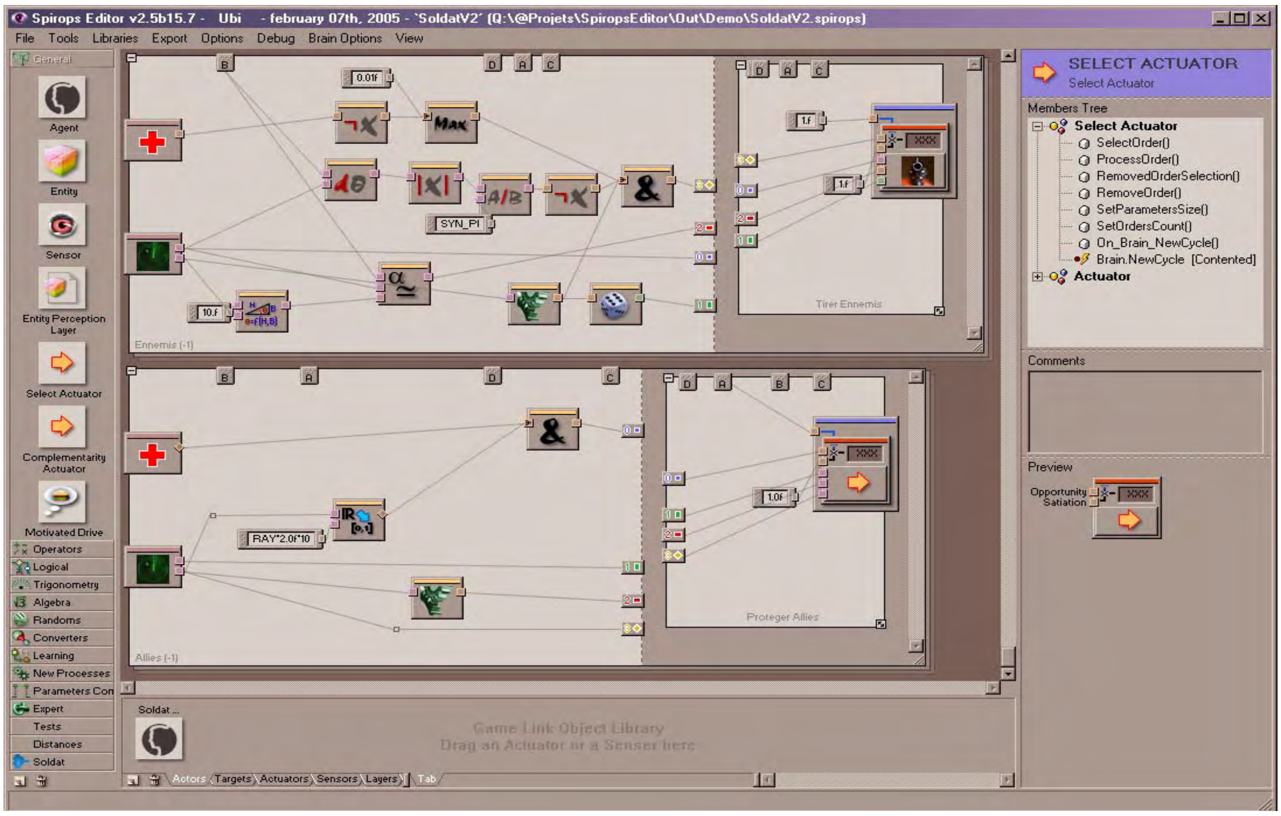
We will see below that the currently used technologies – like the finite state machines – become exponentially complex when the A.I. Acquires more rules and conditions.



Since now, this has forced programmers to limit considerably the depth of their A.I. SpirOps on the over hand allows to break this barrier.

2) An intuitive graphical development environment within reach of everyone.

The interface has the double advantage of making a complex structure much more intuitive and understandable especially when it has to be used and understood by diverse people, and be accessible to the whole team without specific knowledge (programming for example).



3) A humanly understandable technology hence easy to debug.

During the whole creation the A.I. Stays comprehensible (in contrary to "black box" systems) and is then easily analyzed step by step.

4) Visual access to the settings of the A.I. allows to easily adjust its skills.

It is very important to keep control on the "power" of the A.I. Because it must be perfectly adapted to the player's level for him to have fun.

5) An open architecture not linked to any technology in particular.

   You are free to integrate in the A.I. Created by SpirOps any technology of your choice, existing or future (Neural Networks for example). Furthermore, all the basic functionalities of SpirOps are open and overloadable at leisure.

6) Access to the full kernel source code.

   For those who have chosen the "partner" license, you will have access to the full kernel source code of SpirOps. You will therefore understand the foundations of SpirOps and eventually, if a bug is found in the kernel, find a way to bypass it while SpirOps's team is taking care of it.

7) An A.I. Generation in native language (not interpreted).

   We were very vigilant to the memory footprint and also the processing time the A.I. Of a game must take. We have then chosen to generate non interpreted A.I. Directly compiled in native language. The result gives access to very powerful A.I even on very small platforms like mobile phones. (For example: a simple watchman patrolling and attacking intruders takes less than 50 bytes in memory).

8) A system based on brick assembly allowing dynamic generation of A.I. during the game.

   The power and "modularity" of SpirOps allows you to create A.I. On the fly, for example a virtual character in an adventure game could, when acquiring a new object (like a fishing pole), see it's "brain" evolve and have new interests in fishing.

VII] What is exactly what we call "Artificial Intelligence"?

The definition of A.I. Is quite unclear and large; it can vary much between individuals. Let's say it could be: "an application which gives the impression to the user that some actions made by the computer are thought thru". Artificial Intelligence is thus not defined by a technology or a process but by a result: "The impression of intelligence" and for that all means are good. SpirOps is not limited to one or more technologies and offers the user a total liberty of choice over them. SpirOps offers rather a frame and a robust methodology as well as certain advanced functionalities like adaptation and memory. (SpirOps is said "cross-technologies").

However at SpirOps we have a more restricted definition than that exposed above, first of all we limit ourselves to artificial intelligence associated to behaviors (B.A.C.): the choices made by intelligent and autonomous agents which determine their behaviors. It is for this reason that SpirOps is historically designed mostly for virtual worlds' creators. The fields of Artificial Intelligence are vast and there are many applications; if you take a search engine like "Google", the results it gives you are "intelligent", as character, face or digital print recognition programs are "intelligent". SpirOps can adapt to these fields but will be much more effective for behaviors, field which it has been initially designed for.

Next we mean by "Artificial Intelligence", programs capable of solving problems which are not mathematically solvable or whose algorithmic solution is to complex. Thus with our limited definition of A.I., a virtual chess player would go out of the bounds of A.I. Because there exists what is called "brute force" solutions who consist in testing billions of combination (solutions much suitable against a medium human player). What we call "path-find" (search of the shortest way between two points) doesn't fit in either (in fact, there are mathematic algorithms solving these solutions in a suitable way like "A*", in addition there are no notions of "decision making").

VIII] What were the solutions used to create Artificial Intelligence before SpirOps?

Except some special cases (like "Creatures" or "Black & White"), the B.A.C in video games were created from "finite state machines". "Black & White" uses a technology which is called "reinforcement learning" and the "Creatures" use a blend of "Neural networks" and "Genetic algorithms". There are also technologies called "Expert Systems" used in games like "Close Combat 2".

Here is a description of each of these four technologies as well as a small list of their advantages and inconvenient.

1) Condition Based A.I.

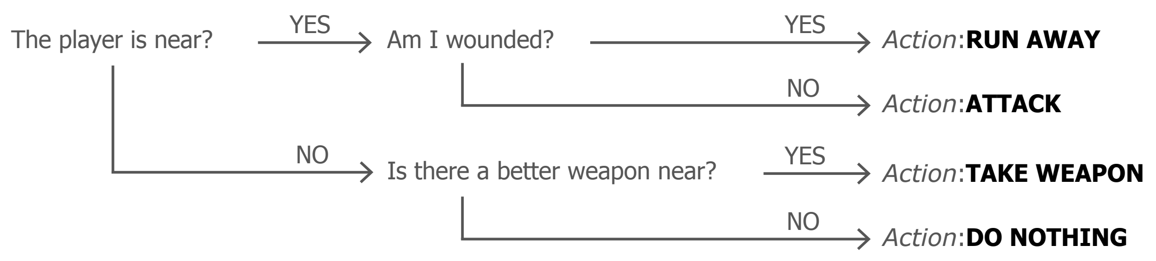
   We can divide those methods into 3 distinct technologies, which are close and share the same advantages and drawbacks:

   1. The rule based programming is the most instinctive method that programmers use often without knowing it. It consists in isolating significant conditions and then triggers actions depending on combination of those conditions.

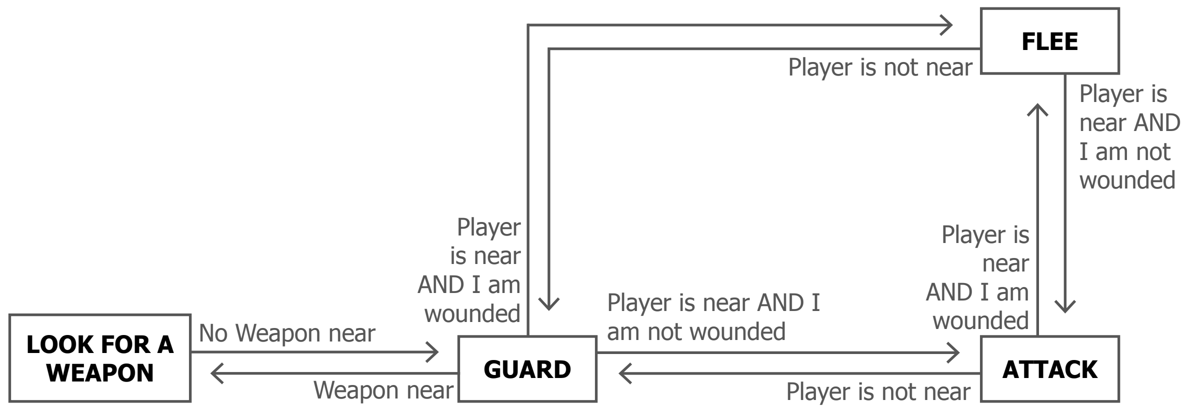
      To clarify this, let us take the example of a basic virtual enemy soldier (NPC). Let us imagine that the

A.I. controlling this soldier has for information its actual position, that of the player as well as the soldier's health. We could isolate the condition "Am I close to the player?" and the "Am I hurt" one. One rule of triggering could then be: "IF I am near the player AND I am not hurt THEN attack the player".

2. The decision trees are quite similar, except that conditions are organized into some kind of tree (as shown below). At each node of the tree, the condition dictates the next embranchment. The leaves of the tree correspond to actions of the character.



3. The Finite State Machines works a little differently. The A.I. is divided into some smaller A.I. which are taking care of a unique sub problem. These parts become the states. The transitions between states are managed by conditions.


FLEE
Player is not near
Player is near AND I am not wounded
Player is near AND I am wounded
Player is near AND I am wounded
LOOK FOR A WEAPON
No Weapon near
Weapon near
GUARD
Player is near AND I am not wounded
Player is not near
ATTACK

The great quality of these methods is that it is instinctive for all programmers and accessible to everyone. But they possess 4 main defaults:

1. The complexity of programming A.I. grows exponentially when the A.I. handles new conditions, and it becomes very difficult and even impossible to program them. If we examine our "simplistic" example of the soldier we find that only a fourth of the cases have been taken care of; In fact, the soldier still doesn't know what to do when the player is far from him (should he roam ? , search for the player ?) nor what he should do when he is hurt (should he run ? should he call for backup ?) and not anymore what he should do when he is hurt and far from the player (should he heal himself ? go back to the base ?). The number of cases to guess is in powers of two which gives 4 cases with 2 conditions but more than a billion after 20 conditions (all cases having to be programmed AND tested).

2. Secondary objectives are not taken into account, the thinking is simplified. For example, in the previous decision tree, when the soldier runs away (because he is hurt and near the player) he will not think of taking the weapon even if he walked over it during its escape.

   Of course as programmers we could say that it is not too bad, but the players seeing that will definitely say that the soldier is dumb; which is the worst judgment to have upon an A.I.

3. The modification of an element of the virtual world can cause huge amount of reprogramming for A.I. Even in being extremely methodical, adding a condition of late can mess with a great part of the A.I. In fact it will be necessary to "inject" the impact of this new condition on all the other existing conditions. Which means a thorough analyze of all the code lines of the A.I.

4. Some would say that the virtual world should be defined at the beginning of the production and never change afterwards; our little experience of industry showed us that in practice, this was hardly ever true. You have to convert your data into conditions. Most of the data that describe the world and the entities in it, are based on values, integer or scalar. For example, the distance between the player and the soldier. This distance must be converted into one or several conditions (Near, Far etc.). This conversion used thresholds (which might be fuzzy thresholds) that are complex to set and tweak.

2) Neural Networks.

The networks are too complex to be described here but one could say that they inspire themselves from the working of biological neurons.

B.A.C. Requires complex neural networks to work well in virtual worlds (like the dynamic recurrent neural networks). These neural networks are very complex to design. You will need an "expert" (engineer or scientist) in your team. The results obtained are tricky to tune, and their adaptation capabilities tend to make them unpredictable.

Moreover it is a technology that is called "black boxed". Its internal structure is very hard to understand and to tweak, you will only be able to judge from the result and it is very – even impossible – to analyze step by step its working (even for its creator).

Its principal attractiveness is its very seducing "magical" feel, due to its adaptability that in some cases has done miracles.

3) Genetic algorithms.

In two words, it is like breeding plants wishing to obtain a plant with the qualities of both its genitors, except that we do that with A.I.

So we create A.I. With the same structure but possessing different settings; after a testing period we chose which one gives the best results and we mix together their settings in order to get better A.I. And so on for "generations" of A.I.

Genetic algorithms are algorithms that work very well in games like "Creatures" but who seemed to have been made for this technology (in those games, you had among other things to breed creatures having their own A.I).

Nothing guaranties that the children A.I. Will be better than their parents. So the results are quite unpredictable. The whole breeding process is lead by a function that measures the fitness of the A.I. If this function is not well done, the whole system will tend to nothing interesting.

4) Learning by reinforcement.

These algorithms are found very easily in games that use them ("Creatures", "Black & White", "Dogz" or "Catz"). If you have seen one of these games you will understand instantly what it is. The idea of punishing or rewarding an A.I. Whether it has made "good" or "bad" choices (as we would do for a pet). Sometimes reward is given by the user, this leads to supervised reinforcement learning. Sometimes the A.I. Understand itself if the consequences of its actions are good or bad. The A.I. Rewards itself. This leads to unsupervised reinforcement learning.

The idea is rather simple and the major problem for the A.I. Is to "understand" which actions lead to such a treatment in the moments preceding.

The genetic algorithms and the reinforcement learning allow the A.I. To evolve during time and thus adapt. Our opinion is that it is what actual video games A.I. Lack. In SpirOps we took those notions of adaptation and we made them generic and predictable, to adapt to any problematic. A SpirOps's A.I. Can benefit from memories, have feedback on its effectiveness and thus acquire what we call "experience" in order to improve and adapt.

In conclusion, no existent technology seems to match the requirements of B.A.C. Neural Networks and Genetic Algorithm are too complex to design and tune, condition based systems tend to limit the complexity of A.I. And become difficult to maintain. SpirOps is our response to those limitations. This is why we qualify SpirOps as a universal behavioral artificial intelligence conception tool.

IX] What other tools exists that could be used in addition to SpirOps?

The tools claiming to be A.I. Development kits are RenderWare A.I., A.I. Implant and PathEngine. Those tools do not deal with Behavioral Artificial Intelligence but rather with other branches of A.I. Related to character movement in a 3D universe like path find or mob animation ("flocking").