

## Re: new here, my lang project...

**Source:** <http://coding.derkeiler.com/Archive/General/comp.object/2005-01/1034.html>

---

**From:** H. S. Lahman ([h.lahman\\_at\\_verizon.net](mailto:h.lahman_at_verizon.net))

**Date:** 01/19/05

Date: Wed, 19 Jan 2005 18:36:49 GMT

Responding to Cr88192...

>>>>>eg: an "order-0" association could probably only really associate being  
>>>>>hungry with eating. this could be helpful, eg, if it is atop a scripted  
>>>>>ai that knows to go towards any food and eat it if it is getting an eat  
>>>>>impulse. an "order-1" association would be needed to associate, eg, the  
>>>>>presence of food, being hungry, and the impulse of eating, or choosing  
>>>>>what to eat, ...

>>>>

>>>>...this seems to me to be several different interactions are involved.

>>>>

>>>>hungry -> look for food.

>>>>food sighted -> chase

>>>>food caught -> eat

>>>>satiated -> sleep

>>>>

>>>>The behavior responsibilities for looking for food, chasing it, and

>>>>eating it are likely to be largely unrelated (i.e., they are

>>>>self-contained and logically indivisible). So each activity is triggered

>>>>by a very specific change to the state of the application that maps to a

>>>>postcondition of some other <self-contained and logically indivisible>

>>>>activity. So events like 'hungry' reflect a very specific result of some

>>>>particular activity.

>>>>

>>>>

>>>>a lot of associative power would probably be needed for something like

>>>>that...

>>>>

>>>>this would not be a system of accurate or developed thought, my

>>>>expectation (as an upper limit) would be something roughly emulating

>>>>basic behaviors (maybe even if not very reliably or only approximately).

>>>>

>>>>humans seem to have an almost arbitrary-order associative ability afaik.

>>>>

>>>>Actually, this sort of thing tends to Just Work when one uses state

>>>>machines to describe behavior (particularly object state machines for

>>>>individual objects). That's because one can apply design-by-contract

>>>>(DbC) to rigorously sort out where the events should go.

comp.object: Re: new here, my lang project...

>>  
>>*Before any behavior (state action), X, can execute, some precondition in*  
>>*the overall algorithm must prevail. That may be as simple as having*  
>>*behavior Y execute first in an algorithmic sequence. Or it can be more*  
>>*complicated and include conditions on certain data being updated in a*  
>>*timely fashion. However, the precondition is usually fairly easy to*  
>>*identify.*  
>>  
>>*That precondition will always be the postcondition of executing some other*  
>>*behavior. That's because only behaviors can change the state of the*  
>>*application, so some behavior must execute to produce the precondition*  
>>*state for executing X. Therefore one has a rigorous technique for*  
>>*determining who cares about announcement events. One takes a given*  
>>*behavior, X, and determines the precondition. One finds the behavior, Y,*  
>>*whose postcondition matches that precondition. Then it is Y's*  
>>*announcement event that X cares about, so one sends the event to X when Y*  
>>*executes.*  
>>  
>>*The point here is that the sender/receiver associations aren't arbitrary.*  
>>*They are rigorously defined by the overall problem solution. [Of course*  
>>*there is a catch. This only works if behaviors are logically indivisible*  
>>*and self-contained. It is also very convenient if behaviors access the*  
>>*data that they need directly rather than having it passed to them. The*  
>>*reason is that one just needs to look at the behavior to know what state*  
>>*variables must be properly updated as part of its precondition. That*  
>>*allows one to daisy-chain behaviors to ensure the data is good when the*  
>>*behavior generating the event is good. (One can ensure data integrity for*  
>>*data that is passed around, but it becomes a major pain in a concurrent*  
>>*implementation.)]*  
>>  
>  
> *err, but I don't get how this really applies to human thought processes and*  
> *learning though (or emulating them).*

Actually, I was just referring to your comment about my treatment of events to the effect that, "a lot of associative power would probably be needed for something like that...". My point was simply that DbC provides a simple but rigorous technique for matching up announcement events with behaviors that care in order to connect the dots of the overall solution sequence.

It's not really about human thought. (One only gets into emulating human thought if the subject matter is the AI itself, as we got into later in the message.) It is more about encapsulating self-contained, logically indivisible behaviors and letting them interact to solve the problem. In theory in an OO development one could define all the objects and their behaviors by simply abstracting intrinsic behaviors of problem space entities that seem necessary to solving the problem. One could even write the object methods and unit test them prior to including any event generation or procedure calls for collaborations with other objects. Then one could then go to a higher level of

Re: new here, my lang project...

comp.object: Re: new here, my lang project...

abstraction (e.g., a UML Interaction Diagram) and apply DbC to determine the solution flow of control. Then one could backfill the methods with event generation or procedure calls from the Interaction Diagram to capture the overall solution flow of control.

In practice experienced OO developers don't do this because it is usually pretty obvious who cares about the postcondition of the method in hand that is being coded. (When one abstracts properties that seem necessary to solve the problem, one needs a megathinker anticipation of the overall solution.) But they do resort to DbC when things aren't so clear (e.g., when the receiver of the event needs data to have been updated consistently in different parts of the program).

>>>>*In this case it seems like BigCat is the single object doing everything; it looks for food, it chases food, it eats food, and it sleeps. But in practice the actual activities are likely to be delegated to other objects or even subsystems. Thus the "behavior" of looking for food may really just involve sending a message to somebody who knows where all the critters are relative to the local map. They do the actual looking and respond with a "food sighted" message. Maybe the chase behavior makes an evaluation: if the candidate food is not acceptable, it sends the "keep looking" message back; otherwise it sends a message to whoever manages chases over the terrain announcing that this Big Cat is chasing that Food. And so on...*

>>>>

>>>

>>>

>>>*well, the problem is that the ai is not all that well structured.*

>>

>>*I suspect there may be another problem. (More precisely, there is a special form of being unstructured.) Ideally the game elements should do their thing independently of whether they are player controlled or AI controlled. That is, the AI should be just as externalized as the player. The player communicates with the game via a UI, which is usually isolated in its own subsystem or layer. The same thing should apply to the AI; it should be encapsulated in its own subsystem or layer. In addition, the AI should issue exactly the same messages that the UI does to control the game artifacts. [The AI, though, needs to do a lot more accessing of data for the state of the game than the player, though, because it doesn't have a "free" display. So it may need a different interface than the UI for that sort of <synchronous> data access. But that is a one-way, synchronous data access.]*

>>

>>*IOW, the AI should be an application unto itself that encapsulates all the smarts that the player provides and it should be completely substitutable with the player UI for controlling any given game artifact. I'll bet that isn't the case.*

>>

>

> *dunno.*

Re: new here, my lang project...

comp.object: Re: new here, my lang project...

In the code you have, is the code different in any way for, say, moving a Soldier depending on whether it is the player or the AI initiating the moving (e.g., does it matter if the Soldier is German or US in Wolfenstein)? Are the player's "pieces" different than the AI's "pieces" in any significant way in what they know or do? Are the player and AI "pieces" dedicated to them (e.g., certain "pieces" are only used by the player and there is a hard-wired link to the UI in their implementation)? If the answers to any of these questions are Yes, then the AI has not been properly decoupled from the manipulation of game artifacts.

That sort of coupling is a manifestation of poor modularization. (Or poor structuring, if you will.) Ideally it shouldn't matter whether a Soldier is US or German; that is just a matter of what uniform is assigned for the graphics at startup. (An oversimplification, of course; a whole set of <data> properties is probably parameterized.) It should make no difference whether the control of movement, etc. comes from the player or the AI.

>>>largely it just that different aspects of state generate messages, which  
>>>can trigger actions, ...  
>>>what actions are triggered by what events has to be "learned", ...  
>>>  
>>>of course, divulging from just coding up what is supposed to happen is  
>>>itself dubious, but if the algo is sound, and it allows sufficiently  
>>>complex associations, then it should be possible to learn basic  
>>>behaviors, possibly near that of what can currently be done with  
>>>hard-coded ai's (possibly, though, at the cost of a lot more processing).  
>>>  
>>>an interesting test could be combining something like this with something  
>>>like genetic algorithms to try to come up with a fairly good model of the  
>>>ai (eg: you have a number of ai's competing with each other and possibly  
>>>humans). those that do best live on, and the rest are replaced by  
>>>combinations of those that did best, ...  
>>>  
>>>assuming a sufficiently good model, this may eventually result in  
>>>something that could compete effectively.  
>>>  
>>>maybe, maybe not.  
>>>  
>>>there are so many things that were not explained, or at least I haven't  
>>>got to them yet.  
>>>  
>>>something else like this should sensibly exist, but I don't know.  
>>>of course, the whole idea could be worth nothing as well...  
>>  
>>Carrying the point above further, I would expect the subsystem that  
>>encapsulates AI strategy to be a unique problem space that is exclusively  
>>devoted to emulating high level human thought. That is a daunting  
>>undertaking and I would expect the AI to have a complex scope. Thus I  
>>would actually expect it to be composed of multiple subsystems, each

Re: new here, my lang project...

comp.object: Re: new here, my lang project...

>>devoted to a particular aspect of that emulation (e.g., strategic vs.  
>>tactical algorithms; economic vs. military goals; etc.).  
>>  
>  
> this is a conventional ai, where the ai is coded to do something or think  
> some way.  
> I am talking about exhibiting abstract learning behavior.  
>  
> in the case of a learning ai, it is not as easy to divide.

I think I have to disagree with that. Whether the AI uses hard rules, heuristics, or some sort of neural net learning, the AI mechanism needs to be encapsulated. I believe that regardless of the AI mechanism, there will actually be several quite different categories of things it will have to think about and each of those categories will involve a unique set of details. That means one is likely to have a different set of rules, heuristics, or neural net configuration for each category.

It should be possible to encapsulate those separate AI mechanism contexts via subsystem encapsulation. In the end they all algorithms that operate on data. That data comes from the various game artifacts.

All one should need is a set of interfaces that update the AI when something changes or an interface that the AI can use to obtain the latest data that it needs. One would probably use the first strategy for a continuous learning AI while the second strategy might be better for hard rules or heuristics.

[Note that modularization allows one to apply the best AI approach for the context. Finding the shortest path for a point-to-point movement is probably a rules based thing. OTOH, determining how aggressive the player is would probably be a learning thing (though maybe not as exotic as a neural net).]

> I appologize if my responses aren't that great, right now I am lacking in  
> time.

So you have a Day Job? B-)

\*\*\*\*\*

There is nothing wrong with me that could not be cured by a capful of Drano.

H. S. Lahman

hsl@pathfindermda.com

Pathfinder Solutions -- Put MDA to Work

<http://www.pathfindermda.com>

blog (under constr): <http://pathfinderpeople.blogspot.com/hslahman>

(888)-OOA-PATH

Re: new here, my lang project...