Robotic Dancing: Exploring Agents that use a Stratified Perceive-Decide-Act Cycle of Interaction

James Benze and Jennifer Seitzer
Department of Computer Science
University of Dayton, Dayton, OH 45469-2160
benzejaa@gmail.com, seitzer@udayton.edu

Abstract

An autonomous agent is any intelligent entity that engages in a perceive-decide-act cycle of interaction with its environment. In this paper we present a formalism using a stratified percept chain that renders an augmented interactive cycle. In particular, we identify two kinds of agents: a "lead" agent, who gains the percepts directly from the environment, and a "follow" agent, who gains its percepts from the lead agent (as well as the environment). In this work, the lead agent procures percepts from the environment, makes decisions based on this input, and passes these new orders in the form of secondary percepts onto the "follow".

We exemplify this formalism in the application area of robotic dancing using models programmed in Alice, a programming IDE/language that facilitates the creation and visualization of autonomous agents.

Introduction

Autonomous agents achieve a level of intelligence by participating in a continual feedback loop to and from the environment. At the onset of the cycle, a percept is sent to the agent from the environment; the agent then internally makes a decision as to the action to perform next in the environment, and then lastly, the agent performs the action. In this work, I have expanded this cycle of interaction by involving two agent types: a "lead" who gets the percept directly from the environment, and a "follow" who gets its percept from the lead. This stratification of percept sources is new and affords many interesting challenges in multi-agent intelligent systems.

Stratified Percepts

In this paper we are defining "stratified percepts" to be a chain of percepts originating from the environment that are interpreted by an intelligent agent, and then received by another intelligent agent. We label the standard agent type receiving its percept from the environment as the "Lead" agent. Additionally, we identify the "Follow" agent as the agent which gets its percepts from the Lead, and executes its actions in response to the Lead's actions.

This kind of percept chain is applicable in many human interaction situations. For example, these percept chains are found in many kinds of partner dancing (Jitterbug, Lindy Hop, Waltz, Tango, etc.), however any chain-of-command domain houses many problems and applications that could benefit from the work and insights gleaned from studying percept chains. Standard organizational structure houses tiers of commands that are relayed down to lower members of the organization. For example, the Commander in Chief is the head of the American Armed Forces. Any decisions made by him must be passed down through the organization until it reaches the battlezone. Being able to simulate this effect would lead to more effective military simulations.

Obviously, when chaining stratified percepts together, one must be careful to keep the lines of communication clear. For example, consider the "Telephone game" played by children. One child at the head of a line of children whispers something in the ear of the next child. This child tries to whisper the same message to the next person, and so on until the entire line of children has been exhausted. Finally the last child announces the message aloud. This game becomes fun since rarely is message pronounced clearly between all children, and the message received by the last child is usually not even similar to original. In order to circumvent this problem, this project does not deal with a chain of intelligence agents, simply looking at the relationship between one "lead" and one "follow."

Cycle of Interaction

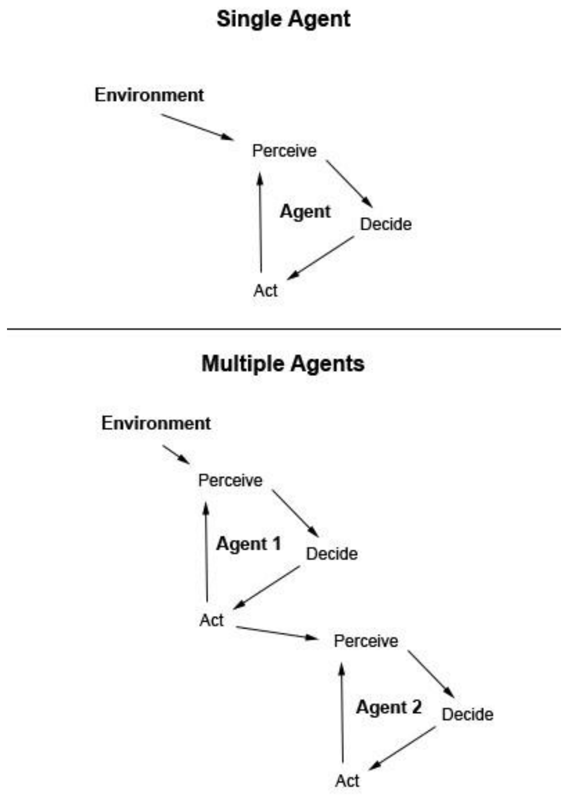

Figure 1: Single Agent vs. Multiple Agents

The actions of an intelligent agent are traditionally defined by a "Perceive-Decide-Act" cycle, as described above. When multiple agents are to be chained together, the cycles of one agent depends on the other, as demonstrated in Figure 1.

Implementation of System DANCER

We exemplify this formalism in the application area of robotic dancing using models programmed in Alice, a programming IDE/language that is geared towards an instructional computer programming and also facilitates the creation and visualization of autonomous agents. Two models of humans in Alice represent the Lead and the Follow. The two models are then made to dance in the style of East Coast Swing. The Lead interprets the "beat" given by the environment, and chooses dance moves to portray, and tries to demonstrate them to the Follow through body movement. The Follow then takes this body movement and interprets it so that she can do the correct move.

Alice

Alice was developed at Carnegie Mellon University in order to provide an easy environment to teach beginner computer science topics to introductory computer science students. However, Alice also contains an integrated graphics environment, allowing the easy placement of lighting, human models, etc with minimal effort from the developer. It was this factor that was crucial in our choice of Alice as the development environment.

One of the most challenging aspects in using Alice, however, is its lack of synchronization objects. Although Alice allows for many threads to run simultaneously (Alice is based on Java), it provides no method of protecting shared data. Because of this, creative ways had to be employed to circumvent race conditions.

East Coast Swing

East Coast Swing was chosen as our application area for several reasons: (1) the moves are relatively easy to model, (2) it is typically an introductory dance, and (3) the authors are both dancers and know East Coast Swing.

In the swing dances, such as East Coast Swing, one dancer is called the "Lead" (typically a male) and one dancer is called the "Follow" (typically a female). The Lead prepares the dance moves, and executes them at the given time. He should also make it obvious to the follow what he is doing. The Follow does her best to interpret the information given to her by the Lead (this information is also called a "lead"). This dance is therefore an excellent representative of a stratified percept chain, since it inherently requires a lead and a follow.

System Design of DANCER

In the Alice architecture, everything is contained within a global class called "World". The World, in this project, acts as the environment, and provides the original source for the percepts.

Contained inside the world are the two intelligence agents, once again designated the Lead and the Follow. Each of these two classes are created with the Alice he/she builder, and is constructed of many smaller classes, each representing a body part (forearm, neck, etc). This allows for the animation of individual body parts, so that the follow and lead are each able to move in a dancing fashion.

The Directional Light and the Bedroom classes are relatively unimportant. The directional light exists to give a visual indication to the user of the beat created by the world. The Bedroom is pure decoration, and was added so that the Lead and the Follow would be on a wooden floor (dancing is not often done on grass or sand, the two default ground textures for Alice worlds).

Finally, there are many "dummy objects" used in the system. These are invisible placemarkers that contain only a location and orientation (yaw, pitch, roll). These are used as markers for the Lead and the Follow, and provide points of reference for their movement.

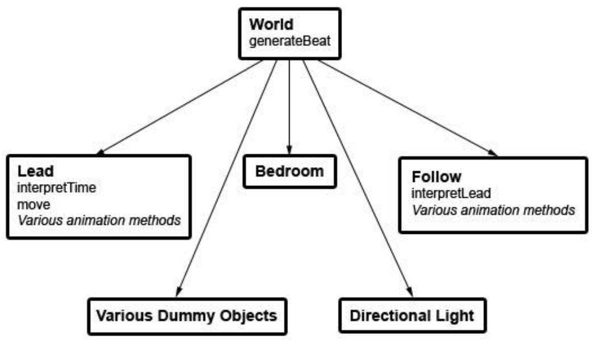

Figure 2: Classes in the East Coast Swing Enviornment

The East Coast Swing Environment

The Lexicon of Moves

In order to execute dance moves, both the lead and the follow agent had to be programmed with a vocabulary of movements that could be performed. Both the lead and the follow agents were programmed with the ability to perform the following dance moves:

- East Coast Basic
- Inside Turn
- Outside Turn
- Tuck Turn
- Repeaters (a variation on a tuck turn)

By programming these moves into the library each agent would be able to visually demonstrate its decision based on the percepts it received.

The System Algorithm and Environment

The purpose of the environment in this project is to provide the music for the lead to interpret. However, one of the limitations of Alice is a limited ability to interpret music in the project. Although an audio file could be played in the Alice environment, there was no way for any object to intelligently interact with it.

As such, a substitute for the music had to be found. Instead of playing music in the background, a flashing light was used instead. This can serve the same purpose as a musical beat, since a musical beat is just a repeated auditory impulse. A repeated visual impulse creates the same effect but through a difference sense. Since one sense is as good as any other for this project, a flashing light was deemed to be an acceptable substitute.

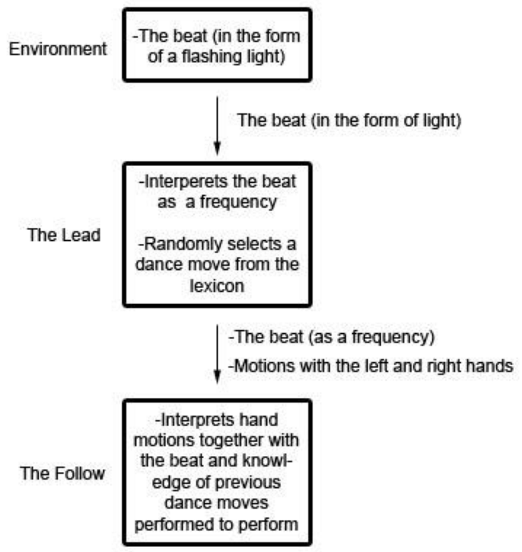
Environment
-The beat (in the form of a flashing light)
The beat (in the form of light)
The Lead
-Interperets the beat as a frequency
-Randomly selects a dance move from the lexicon
-The beat (as a frequency)
-Motions with the left and right hands
The Follow
-Interprets hand motions together with the beat and knowl-edge of previous dance moves performed to perform

Figure 3: System Algorithm

Lead

The Lead agent has two primary responsibilities: to interpret and act upon the information from the environment, and to provide a clear percept to the Follow dictating her actions.

Clear interpretation of the beat of the music can be challenging for many humans. In our system, the Lead uses Alice's internal timing function to record the time of each beat. By viewing the difference between each time, the Lead can accurately predict when the next beat would occur. This is essential in determining the speed in which to dance.

One problem was occurred due to the lack of synchronization ability in Alice. Due to other system processes running on the processor, there exists a discrepancy between the pulses of light and the time that they are recorded by the Lead agent. However, when no other projects were actively running on the system, the extra time variation was only about 0.06 to 0.09 seconds. By subtracting 0.1 seconds from each recorded beat, and by keeping the number of processes low, the recorded beat by the lead is at or slightly below the beat provided.

It was determined that having a recorded beat less than the provided beat was preferable than having the recorded beat to be greater than the provided beat. When the recorded beat is less, the lead can simply pause, and wait for the action to "catch up", whereas if the lead was too slow, he would simply be slower and slower until he has fallen noticeably behind.

To determine the specific move that the lead executes, the lead simply uses Alice's random number generator, and randomly chooses a move. It then passes on this percept to the Follow.

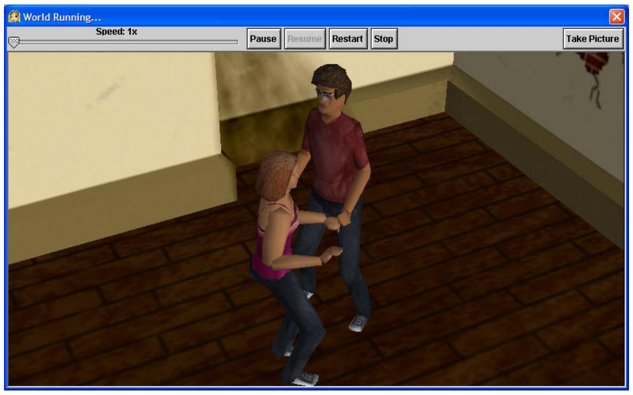

Figure 4: DANCER preparing a turn

Follow

The follow program receives its variables through a function call. There are four different variables that are passed to the function as forms of "leads": The lead from the left hand, the lead from the right hand, the beat of the music, and the distance between the two agents. The first two inputs should be obvious. These are the physical connect between the head and the follow. With is left and his right hand, he provides motions to direct the follow. The beat parameter is slightly more subtle, but in leading, the Lead pulses slightly, and in this way can transfer his knowledge of the beat to the Follow.

The distance parameter represents the translational velocity that the lead transfers to the follow. Doing certain moves the lead doesn't just cause the follow to spin, but causes her to translate as well. The distance parameter represents this type of lead.

The follow uses a nested if statement to determine further course of action. She also takes her current momentum into account while determining further courses of action. The momentum is a variable set from previous moves.

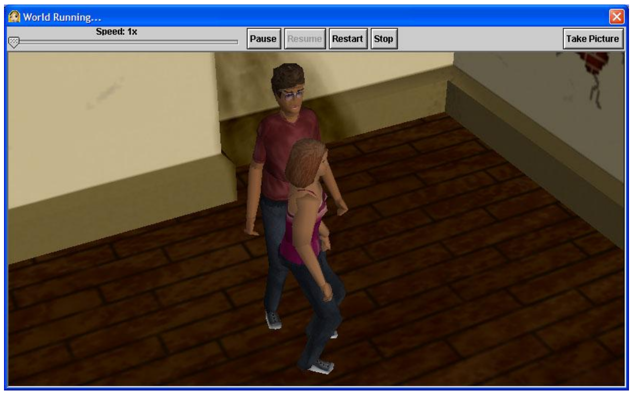

Figure 5: DANCER during a turn

Lead-Environment Lag

As mentioned before, a lack of priority functions in Alice causes the timer function not be entirely accurate. As such, the timer between the lead and the environment had to be adjusted with a 0.1 second reduction in time to account for this. In addition, causing the processor to work a large number of tasks will cause this timer to displace even more. As such, the program should only be run in parallel with as few other programs as possible. No lag exists between the Lead and the Follow, since no system clock is used, only a function call.

Multiple Threads Between Lead-Follow

Originally, the plan was to have the Lead and the Follow run in separate threads, and for the Lead to send some sort of signal containing the leads to the Follow in order to start her motion. However, the lack of synchronization elements available in Alice made this unfortunately impossible. There was too much lag between the interactions between the two agents, which caused either too great of a delay in the Follow's actions. As such, a simple function call was used to pass percepts between the two agents.

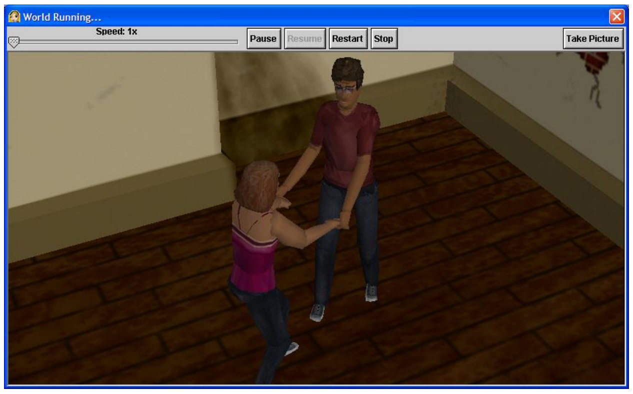

Figure 6: DANCER during the East Coast Basic

Conclusions and Future Work

An example of an interaction between two hierarchical agents was successfully modeled between the Lead and the Follow. However, many of the challenges have inspired us to probe further in future work as we describe here.

Clearer Communication between Agents

Currently, all percepts are passed directly and digitally between the Lead and the Follow, guaranteeing clear and precise communication. However, in real world scenarios, the communication between agents may become unclear. An example of this is perfectly clear in the Telephone game mentioned earlier. The specifics of a whispered message will become garbled over time, since one is difficult to hear.

Because of this potential problem methods should be implemented so that the Follow will approximate the closest action based on the decisions given. The other agents should be able to gracefully recover from a mistaken decision.

Recovery After a Delayed Response

One of the problems experienced in programming this project was that the Follow would sometimes have drastically delayed responses from the Lead. This could cause a desynchronization between the two Agents. Methods should be employed so that the Lead agent could gracefully recover from this separation.

Multiple Inputs

In this situation, each agent has a clear superior: The Lead's actions are governed by the environment and the Follow's action are governed by the Lead. However, in many situations, the Agents must receive input from multiple sources. For example, what if the Follow was able to receive input from the environment as well as the Lead? This input could be constructive (the Follow using the input musical beat in addition to the beat received by the lead to more accurately dance), or destructive (A fire alarm sounds, and the Lead ignores it. Does the follow obey input from the Lead and dance, or from the Environment, and leave the room. Both types of input will have to be considered.

Whisper-Down-the-Lane Agents

Here, an agent relationship was successfully demonstrated between a single Lead agent and a single Follow agent. This interaction could be expanded to chain of lead-follow Agents. Care would have to be taken to further reduce the risk of the other problems listed here.

Summary

Although this project accurately can portray a stratified percept chain, further testing and modeling is required should be performed in order to more accurately and effectively model this style of relationship between intelligent agents. The temporal aspect of decision making is much more important in hierarchical agent relationships, and methods of communication must remain remarkably clear in order for lower agents on the hierarchy to remain effective. Experimenting with ways to recover from these kinds of errors will vastly improve the robustness of hierarchical agents and will allow us to more accurately understand this style of interaction.

References

Alice.org, Available: http://alice.org [Accessed: March 22, 2010].

Alice—Project Kenai, Available: http://kenai.com/projects/alice [Accessed: March 22, 2010].

W. Dann, S. Cooper and R. Pausch, *Learning to Program with Alice,* Upper Saddle River, New Jersey: Pearson Education, Inc. 2006.

LEGO.com MINDSTORMS: Home, Available: http://mindstorms.lego.com/en-us/Default.aspx [Accessed: March 22, 2010].