A Game AI Approach to Autonomous Control of Virtual Characters

Kevin Dill
Lockheed Martin
Burlington, MA 01803
kevin.dill@lmco.com

ABSTRACT

There is a strong need to develop Artificial Intelligence (AI) for virtual characters which are:

- **Autonomous** – able to function effectively with little or no human input at runtime
- **Reactive** – aware of and responsive to the evolving situation and the actions of the trainees
- **Nondeterministic** – the viewer should never see exactly the same thing twice
- **Culturally Authentic** – act as a person of the portrayed culture would
- **Believable** – maintain immersion by acting in a believably human way

This could greatly reduce the training costs, increase accessibility, and improve consistency.

As one aspect of the Future Immersive Training Environment Joint Capabilities Technology Demonstration we created the "Angry Grandmother," a mixed reality character portraying the elderly grandparent of an insurgent whose home is entered and searched by the trainees. She needed to be believable, culturally authentic, nondeterministic, and reactive within the limited scope of the scenario. In addition, she needed to be capable of autonomy, but also responsive to direction from the instructor/operator.

The last 10 years have seen a dramatic improvement in the quality of the AI found in many video games; in our opinion, game AI technology has reached a level of maturity at which it is applicable to immersive training. Accordingly, we built an AI which combines Behavior Trees (BTs) and utility-based approaches. This approach is a descendant of that used in several extremely successful video games, including the Zoo Tycoon 2 franchise, Iron Man, and Red Dead Redemption.

This paper will present the AI architecture which we used for the Angry Grandmother, compare and contrast it to relevant game AI approaches, and discuss its advantages particularly in terms of supporting rapid development of autonomous, reactive characters, but also in terms of enabling that crucial dichotomy between autonomy and operator control.

ABOUT THE AUTHORS

Kevin Dill is a staff software engineer at the Lockheed Martin Advanced Simulation Center. He is a recognized expert on Game AI and a veteran of the game industry, with seven published titles under his belt. He was the technical editor for Introduction to Game AI and Behavioral Mathematics for Game AI, and a section editor for AI Game Programming Wisdom 4. He has taught classes on game development and game AI at Harvard University, Boston University, and Worcester Polytechnic Institute.

A Game AI Approach to Autonomous Control of Virtual Characters

Kevin Dill
Lockheed Martin
Burlington, MA
kevin.dill@lmco.com

BACKGROUND

Infantry immersion training reproduces as accurately as possible the environment to which the trainees will be deployed. The intent is to recreate not only the physical environment (the sights, the sounds, the smells), but also the language, culture, and behaviors of the locals. This approach has become increasingly popular in recent years, largely due to its effectiveness in preparing Marines and soldiers for contemporary combat operations. It is conducted at a number of locations, one of which is the Infantry Immersion Trainer (IIT) at Marine Corps Base Camp Pendleton.

The IIT is a recreation of a small town similar to what would be found in Iraq or Afghanistan. Since its inception, it has relied heavily on human roleplayers to act as the residents of that village. This has not been entirely satisfactory. While the roleplayers do an excellent job of providing realistic, culturally authentic immersive characters, they require expenditures of thousands of dollars each day. In addition, there are certain vital roles that can't easily be portrayed with live roleplayers. Appropriate personnel may simply not be available (as is often the case with the elderly), or there may be legal and moral prohibitions against using them (as is the case with children). Finally, human roleplayers can deliver inconsistent results. While they are typically quite good, they may do or say the wrong thing at the wrong moment. We ultimately hope to reach a point where the consistent reproducibility of technological training experiences will prove superior to the varying quality seen with human roleplayers.

To be successful, a technological alternative must at a minimum provide effective training, hitting key training objectives, just as a live roleplayer does. It must also provide the same level of realism and cultural authenticity, so as to avoid any significant loss in the trainees' sense of "being there." Finally, a successful solution should address the cost issue, reducing manpower requirements. Thus, solutions which simply replace roleplayers with operators are not viable.

With all of this in mind, our virtual characters need to have at least the following characteristics:

- They need to be ***autonomous***, able to operate with little or no human input.
- They need to be ***reactive***, which is to say that they are visibly aware and responding to the ongoing situation and the actions of the trainees.
- They need to be ***nondeterministic***. Their actions should *random but reasonable*, so that they are not predictable or repetitive while still ensuring that every action makes sense.
- They need to be ***culturally authentic***, acting as a person of the portrayed culture would.
- They need to be ***believable***. The sense of immersion must be maintained, so the characters need to look and feel like a real human.

The Future Immersive Training Environment JCTD

The overall goal of the Future Immersive Training Environment (FITE) Joint Capability Technology Demonstration (JCTD) was to demonstrate that "by simulating the immersive conditions of the combat environment… we can improve tactical and ethical decision-making by small unit leaders, increase small unit operational effectiveness, improve individual resiliency, and reduce casualties" (Muller 2010) Under Spiral 2 of this JCTD we developed a wide variety of technologies that were integrated into the IIT, many of which helped to reduce the reliance on roleplayers. The demonstration consisted of a series of five training scenarios, which were used for live training of Marines prior to their deployment to Afghanistan.

This paper focuses on a virtual character from the fourth scenario in the series. In this scenario, the trainees receive credible intelligence that an insurgent stages regular illicit meetings in his home at the edge of the village. Consequently, the trainees (with Afghan soldiers attached) isolate and enter the insurgent's home with the intention of conducting a search and detaining any enemy forces present. Unfortunately, the only occupants are an elderly woman and her grandson, both of whom are hostile but not violent.

The Angry Grandmother

The Angry Grandmother ("Granny") character, shown in Figure 1, is intended to introduce trainees to a non-

cooperative character who challenges their interpersonal skills (through the medium of an attached interpreter), forces them to consider relevant cultural training, tests their knowledge of detention criteria and tactical questioning procedures, creates non-kinetic complications during the conduct of a house search, and stresses their judgment regarding when and how to contact higher headquarters in an ambiguous situation.

An elderly woman and a child are ideally suited for this role, putting the trainees in an ambiguous situation that blends a known hostile location with belligerent, but obviously nonthreatening, occupants. These roles cannot be portrayed age appropriately by live roleplayers. Consequently, we created a mixed reality character, which is to say a life-size character, projected on the wall, who appears to be in the room with the trainees. She responds much as a live actor would. Trainees can talk to her, threaten her, shoot her, order her to surrender, question her about her grandson, and so on. The ultimate goal is that interacting with her should be no different from interacting with a human.

Previous mixed reality characters at the IIT have AI that is heavily scripted, with minimal input from the environment aside from the ability to detect shots fired, making them most appropriate for use in kinetic "shoot or no-shoot" scenarios requiring relatively short interactions with the trainees. In contrast, the Angry Grandmother is expected to remain active for ten minutes or more. Over this time, she needs to remain visually and behaviorally plausible, culturally appropriate, and responsive to the ongoing situation and to a broad set of possible trainee actions. Further, she needs to act in such a way as to create the complex training experience described above. Finally, the IIT's operators have many demands placed on their attention. As a result, she needs to be able to act autonomously and still present a credible performance. At the same time, any AI is imperfect, and her sensing mechanisms are fairly limited, so she also needs to remain responsive to operator control.

During a typical run through the scenario, Granny begins by calling out, asking the Marines to leave as they knock on her door. When they enter, she screams and flinches back, and then proceeds to berate them, telling them that her family is innocent, that Americans are unwelcome, and that they should leave her home. She continues in this vein as the Marines search, although her demeanor gradually changes as they get closer to finding something, becoming less angry and more anxious. She responds appropriately if threatened or shot at, but she is too worked up to be rational, so if the Marines try to talk to her then she just yells louder.

Figure 1: The Angry Grandmother and her grandson.

If the Marines find hidden contraband, Granny will begin crying inconsolably. She continues to weep until the Marines force her to surrender, at which point she kneels on the ground with her hands on her head. The Marines will then typically call in for instructions, and be told to question her and then let her go.

In Dill and Dreger (2011) we gave an overview of the systems that are involved in Granny's operation. This paper will focus specifically on her AI architecture.

AI ARCHITECTURE

The basic framework of our architecture is a modular, hierarchical decision making approach, similar to the popular Behavior Tree (BT) architecture, called the Component Reasoner. It can support many approaches to decision making, but we rely primarily on a utility-based approach called the Weight-Based Reasoner.

The Component Reasoner

BTs, as realized in the context of game AI, were first proposed by Damian Isla at the 2005 Game Developer's Conference. Since then they have exploded in popularity, largely replacing Hierarchical Finite State Machines (HFSMs) as the AI technology of choice for many genres, particularly first-person shooters (FPSs).

A BT consists of a hierarchy of *selectors*, each of which chooses among several options. These options can be concrete (i.e., something the character will actually do), or they can contain another selector which makes its own decision. Control works its way down through the tree of selectors until it arrives at a concrete option.

BTs have two major advantages. First, the hierarchical approach is extremely powerful, avoiding spending processing time on irrelevant decisions, and is a natural way to structure the AI such that independent decisions

are decoupled. Second, the options are modular, which is to say that a given option can appear multiple places in the tree. This prevents the need to re-implement functionality every place that it is used.

By design, BTs rely on simple Boolean discriminators for their selectors. This simplifies implementation, but puts a limit on how well the AI can examine the subtle nuance of a situation before making a decision (Dill 2008). More generally, it has been our experience that there are often cases where a more complex approach to decision making should be used for a particular decision, while retaining simplicity elsewhere (Dill, Rabin, & Schwab, 2010). This was validated by a recent Game Developers Conference panel (Dawe et. al. 2010), which discussed the advantages and disadvantages of several of the most popular AI approaches and concluded that each of them works well for some situations but poorly for others.

Thus we want a framework which retains the hierarchy and modularity of the BT's structure, but allows us to employ complex decision makers where appropriate while retaining support for simple selectors elsewhere. The Component Reasoner does this by using *reasoners* rather than selectors to make each decision.

The difference between a reasoner and a selector is subtle but important. Selectors are expected to use simple logic, such as taking the first valid option, selecting each member of a sequence in order, or using fixed probabilities (assigned a priori, not at runtime) to make their decision. In contrast, a reasoner is allowed to be arbitrarily complex. The only requirements are that it be configurable via XML and that it support a standard interface. Thus, nearly any approach to decision making could be implemented as a reasoner.

The advantage of this approach is that it allows us to select the most appropriate approach for each decision being made. For decisions that are highly deterministic we can use a BT-style selector. For decisions that require us to weigh the relative advantages of several possibilities, a utility-based approach will work well (Dill and Mark 2010). For decisions which require us to build complex sequences of actions, Goal Oriented Action Planning (Orkin 2004) or Hierarchical Task Network Planning (Gorniak 2007) might be used. In a situation where the AI needs to learn from past results, we might attempt something like Genetics-Based Machine Learning (Harrison 2007). And so forth.

The root of the Component Reasoner contains a single *option*. An option is a structure which contains one or more *actions*, all of which will be executed if that option is selected (the root option is always selected). As in a BT, these actions may be *concrete* or they can contain another reasoner. If they contain a reasoner, it can use whatever approach makes the most sense for that particular decision to pick from among its own options. Control works its way from the root option down through its actions to the reasoners they contain, into the options those reasoners select, and so on until we reach the concrete actions in the leaves.

It's worth emphasizing that this structure supports parallel execution. Because an option can contain multiple actions, including multiple subreasoners, it can do more than one thing – and even make more than one set of decisions – at once. This is a capability which is missing from all too many AI techniques.

All of the configuration data which drives a character's performance is contained in XML. We support limited inheritance, allowing us to specify default XML for a particular option, and then reference it (and overload specific values, if needed) elsewhere. This data-driven approach allows for rapid configuration without fear of introducing code bugs or long recompiles.

While the Component Reasoner can support a wide variety of reasoners, in practice we found that Granny needed only two types: a simple sequence selector, and the Weight-Based Reasoner.

The Weight-Based Reasoner

In keeping with our desire to create a nondeterministic AI, we want our choices to be random but reasonable. We also want to be reactive and believable, responding to the current situation in a human-like way. Utility-based AI works well for this. The Weight-Based Reasoner is a modular, utility-based reasoner, and is our primary mechanism for achieving the above. We discuss the utility-based and modular aspects below.

Dual Utility AI

Utility-based approaches are widely used in games, particularly games which have more complex decision making to do, and which require autonomous, reactive AI, such as strategy games (e.g. Civilization, Empire Earth, Dawn of War), sandbox games (e.g. The Sims or Zoo Tycoon), and those FPSs with more believable AI.

There are two common approaches to utility-based AI. One is to score every option, perhaps including a random component in that score, and then take the option with the highest score. The second is to assign a weight to every option, and then use weight-based random selection to pick an option. With the latter approach, the probability for selecting each option is determined by dividing the weight for that option by the total weight of all options. In other words, if you had two options, one with a weight of 2 and one with a

weight of 1, you would select the former 2/3 of the time and the latter the remaining 1/3 of the time. Each of these approaches has strengths and weaknesses, but past experience has led us to believe that they are synergistic when used together.

With this in mind, the Weight-Based Reasoner has two utility values: *force* and *weight*. Both are calculated dynamically at run time based on the current situation, which is what allows the AI to be reactive to the ongoing situation. Conceptually, force is used to divide options into categories, where a category with higher force is always preferred over one with lower force. Weight, on the other hand, indicates how "good" an option is – that is, how appropriate it is to the current situation and, when it is appropriate, how likely it is to succeed – relative to other options within the same category. Thus an option that is completely irrelevant will be given a weight of 0 and will not be selected. One which is appropriate but unlikely to work, or which is only marginally appropriate, is given a low weight. One that is both very appropriate and very likely to succeed is given a high weight.

When selecting an option we first eliminate any completely inappropriate options (i.e. those with a weight of 0). Next, we find the highest force category, and eliminate all options that don't belong to it. Third, we narrow down our field to only the most reasonable options by finding the weight of the best option, and eliminating all options whose weight is significantly below it. The exact meaning of "significantly below" is data-driven, and is defined as a percentage of the best weight, typically between 5% and 25%. At this point we should have only reasonable options, but some may still be better than others, so we use weight-based random to select from among them.

Again, those steps are as follows:

1) Eliminate all options with a *weight* of 0.
2) Determine the highest *force* of any remaining option, and eliminate all options with lower force.
3) Eliminate all options whose *weight* is less than some percentage of the best remaining weight.
4) Use weight-based random selection to choose from among the options that remain.

As an example of how this works in practice, consider the options related to reacting to a grenade. These options would have a high force, because reacting to a grenade is more important than most other things (such as reloading, firing at the enemy, or buying a sandwich). Their weight reflects appropriateness and chance of success. Thus if there is no grenade to react to, the weights would be zero and the options would be eliminated (despite the high force). If there is a grenade, the weights of these options would depend on their suitability given the details of the situation. Throwing the grenade back or kicking it away is a bad choice if the grenade lands out of reach, for example. Likewise, diving on top of the grenade doesn't make any sense if there is nobody nearby to protect, or if everybody else has cover. Diving behind cover is typically a good option, but only if there is cover nearby.

During execution the AI for a particular character might end up with diving behind cover, diving onto the grenade, and throwing the grenade back as valid options. These are all reasonable options, but they are not equally likely to be selected. Diving behind cover might have the highest weight, for example, throwing the grenade back might have an intermediate weight, and the weight for diving on the grenade might be quite low. The weight based random selection will ensure that the option with the highest weight is selected most frequently, but that the others all have a chance of being selected as well.

Modular Decision Making

The modular approach to decision making used in the Weight Based Reasoner was first discussed in Game Programming Gems 8 (Dill 2010). The key idea is that the logic for a decision can be broken in to one or more discrete *considerations*, where a consideration is a piece of code which looks at one aspect of the situation in isolation and then returns its evaluation in such a way that it can be combined with that of other considerations to drive the overall decision.

While there are a great many decisions that need to be made by the AI, there are actually relatively few types of considerations, and those considerations are reused frequently from decision to decision. Thus, considerations represent reusable patterns which can be implemented once and then plugged together to create decision-making logic.

Our implementation uses considerations which consist of three parts: force, base weight, and multiplier. The overall force of an option is the maximum of that of all its considerations, while the option's weight is calculated by first adding the base weights together and then multiplying by the multipliers. Thus, each consideration can do any of the following:

- Assign the option to a higher force category by returning the appropriate force value
- Eliminate the option by setting the multiplier to 0
- Add a positive or negative modifier to the base weight
- Scale the weight with a multiplier that is not 1

There are two major advantages to this approach. First, it greatly reduces code duplication and standardizes the implementation of each instance of a consideration. This reduces the likelihood of bugs and simplifies the task of fixing them (because each bug only has to be fixed in one place). It also reduces the size of the code base, making it easier to remember how things work. Second, creating decision logic by plugging considerations together is much faster than implementing the same logic in C++ from scratch. This makes sense, because we are dealing in larger chunks of code, not writing things a line at a time. Thus, we are able to create and to modify the Angry Grandmother's behavior extremely quickly. This allowed the AI team to remain agile, responding to dramatic changes in design right up to the last minute, although the need to create animation and audio assets was a constraint.

Granny's AI uses six standard types of considerations:

- The **Tuning Consideration** applies default values which are set in XML and never change. If a Tuning Consideration is not specified in XML then a default one is created with a base weight of 1, a multiplier of 1, and a force of 0.
- The **Timing Consideration** evaluates how long the option has been executing (if it is currently executing), or how long it has been since it last executed (if it is not executing). This can be used to encourage the AI to continue executing an option that it just picked (so that we don't immediately allow random selection to pick something different), or to prevent us from reselecting an option for a specified "cooldown" period once it is completed (so we don't repeat the same action twice in close succession).
- The **Hostile Fire Consideration** returns specific values when shots have been fired nearby.
- The **Was Shot Consideration** returns specific values when the character is hit by gunfire.
- The **External Variable Consideration** tracks the values of variables which can be set remotely. These are used to allow external systems to provide input to the AI.
- The **Utterance Finished Consideration** keeps track of the current line of dialog being spoken, and selects its values accordingly. In practice, it is used to encourage Granny to continue executing the option which triggered a line of speech until that line of speech is finished (so that she doesn't interrupt herself or suffer repeated failures in attempting to start a new utterance before the last one completes).

As a simple example of how considerations are combined to generate utility values, consider the option which causes Granny to die when shot. This option has a Was Shot consideration that sets the force to 1,000,000 if Granny has been shot, and sets the multiplier to 0 otherwise. It uses the default Tuning consideration. Thus, if she has been shot then this option will have a force of 1,000,000 (from the Was Shot consideration) and a weight of 1 (from the tuning consideration). Since 1,000,000 is the highest force of any option, this will result in this option always being executed, no matter what else she might be doing at the time (which is appropriate – death doesn't wait for you to finish your sentence). On the other hand, if Granny has not been shot then this option will have a force of 0 (from the Tuning consideration) and a weight of 0 (due to the multiplier on the Was Shot consideration). Thus in this case the option will be eliminated during the first step of evaluation.

UTTERANCE SELECTION

In this section we will discuss the logic used by Granny to decide what to say. Figure 2 shows the corresponding Component Reasoner layout. The root is shown to the left, with the children of each option connected by a solid arrow. A dotted arrow indicates an option whose children are not shown. An option with no arrow has no children. Thus, for example, the Rant option has three children: Hot, Warm, and Cold. Cold has 26 children in turn, each of which represents a distinct line of dialog. Examples of cold lines include "You are frightening my Grandson, you should go!", "This is not helping, you think we trust you?", and "Go home American!" (all spoken in Pashto).

Granny's AI has only 9 top-level options. We divide these options into three categories: *behavioral states*, *situational responses*, and *single lines of dialog*. The key conceptual difference is that behavioral states define Granny's overall performance, and she stays in them over the long term (often many minutes). In contrast, situational responses and single lines of dialog

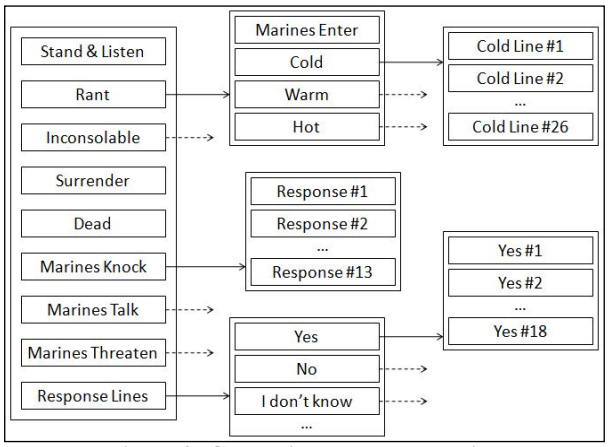

Figure 2: Granny's utterance selection.

take control only briefly, typically for a single line of dialog. When they finish, control returns to the current behavioral state. All of this is accomplished through appropriate use of the considerations.

Behavioral States

The Angry Grandmother has five behavioral states:

- **Stand and Listen**: This is the default state. Granny simply stands quietly in place.
- **Rant**: Granny berates the Marines in an attempt to get them to leave. This state is used while the Marines search the room. It has four substates:
  - **Marines Enter**: Played the first time Marines come into the room. Granny screams and launches into her tirade.
  - **Cold**: Marines are not close to finding anything, and Granny is mostly angry.
  - **Warm**: Marines are getting closer, and Granny is getting nervous.
  - **Hot**: Marines are about to find something. Granny is very anxious.
- **Inconsolable**: Granny cries inconsolably, beseeching Allah to help her poor grandson.
- **Surrender**: Granny places her hands on her head and kneels on the floor.
- **Dead**: If shot, Granny will die.

In most cases, transition to a particular state results either from an operator hitting a button in the Instructor/Operator Station (IOS), or from a Marine moving into a particular location.

Regardless of the trigger, the result is that a remote system (the IOS or the tracking system) will set the "GrannyMood" variable to a specific value, indicating the need to change to a particular state (0 for Stand & Listen, 1 for Rant, 2 for Inconsolable, etc.). This is detected using an External Variable consideration

When the action begins execution, it sets GrannyMood to an alternate value (100 for Stand & Listen, 101 for Rant, 102 for Inconsolable, etc.). This second set of values is necessary to keep track of the currently executing state so that we can return to it after a situational response or single line of dialog is selected.

With that in mind, the full set of considerations driving the selection of most of these options is as follows:

- A Tuning consideration, which sets a force of -1, base weight of 1, and multiplier of 1. We ensure that there is always an option with a force of 0 available (discussed below), so if this force isn't overridden by one of the other considerations then the associated option will not be selected.
- An External Variable consideration, which watches for the GrannyMood variable to indicate that a new request has occurred, and sets the force to 10 when it does.
- A second External Variable consideration, which checks the GrannyMood variable to see if this is the currently executing state, and sets the force to 3 when it does.

Thus, the currently executing state will have a force of 3. When it is time to change that state, the GrannyMood variable will be set accordingly and the corresponding option will be assigned a force of 10. Otherwise, any unselected state will have a force of -1.

There are two exceptions to the above pattern. First, the Stand & Listen state has a default force of 0, rather than -1. Thus if there is no other option with a positive force, this one will always be selected. This only happens when the simulation first starts up (after that, there is always a currently executing state). Second, the Dead state is selected as described in the previous section.

Situational Responses

We identified three common situations which merit a response from Granny:

- **Marines Knock**: This response is played when the Marines knock on Granny's door, or if they stand outside for an extended period of time.
- **Marines Talk**: If the Marines try to talk to Granny while she is ranting, these lines can be used to "shout them down."
- **Marines Threaten**: This response is used if the Marines threaten Granny, point weapons at her, or fire their weapons nearby.

Situational responses are generally selected by the operator, although Marines Knock can be triggered from the tracking data, and Marines Threaten is automatically triggered if shots are fired nearby. Thus, like the behavioral states, these actions are typically triggered when an external variable, in this case "MarineAction" is set to a corresponding value. Note that we do not use the GrannyMood variable, because we want to retain knowledge of the currently selected state.

The considerations are as follows:

- A Tuning consideration, which sets a force of -1, the same as we have for the behavioral states.
- An External Variable consideration, which sets the force to 10 when the action is activated.

- An Utterance Finished consideration which sets the force to 7 until the line of dialog is complete.

In addition, the Marines Threaten response has a Was Shot consideration which can set force to 10.

Thus, these actions will not be selected until the External Variable (or Was Shot) consideration sets their force to 10. Their default force of -1 will always lose to the force of 0 assigned to the Stand and Listen state. Once started, their force will be 7 until the line of dialog is complete, at which point it goes back to -1 and the current behavioral state (which still has a force of 3) will resume.

Single Lines of Dialog

In addition to the above, we have roughly 40 different lines of dialog which are intended to be used toward the end of the scenario, when the Marines are questioning Granny. These include everything from simple "yes," "no," and "I don't know" answers to more detailed responses to likely questions (e.g. "That's just medicine for my headaches!"). These lines can only be triggered by an operator, and the considerations for them are identical to those described for situational responses.

Selecting a Sound File

So far we have discussed only the decision making at the top level of the reasoner – that is, the selection between the 9 options shown all the way to the left in Figure 2. A few of these (Stand & Listen, Surrender, and Dead) are concrete actions which take direct control of the performance. The remainder contain subreasoners. In most cases, the subreasoners pick the specific line to play from within a larger pool.

The simplest examples of this are the single lines of dialog. It may be that Granny will use the same line many times in close succession, particularly for generic lines like "yes" or "I don't know." When this occurs, we don't want the user to hear the exact same sound file, with the exact same intonation, as this obvious repetition may break the suspension of disbelief. Instead, we recorded many versions of each line. We use a *sequence selector* (instead of a weight-based reasoner) to select the actual line to use. This selector returns each of the lines in a pre-specified order. In the unlikely event that all of the lines have been used, the sequence starts over with the first line. This maximizes the time before a given sound file is reused.

The selection of dialog for the situational responses can be a bit more complex. One advantage of the weight-based reasoner is that it allows us to define complex sequences with relative ease. Marines Knock is one example of this. In this case, we have seven utterances that we want to play in a specific order, which become increasingly irate. In addition, we have six more utterances that can be randomly selected if the Marines continue to knock. The expectation is that the Marines won't knock more than seven times, but we need to be prepared for this eventuality.

Remember that due to the hierarchical nature of the Component Reasoner, these options are only evaluated against one another, not against any of the higher level options. This allows us to isolate the decision to play a response from the selection of the actual response to play, which makes configuration much easier. With that in mind, the initial seven responses have the following three considerations:

- A Tuning consideration sets their force to 7, 6, 5, 4, 3, 2, or 1, defining the order in which they will be played.
- An Utterance Finished consideration sets their force to 9 while they are playing, preventing the selection of a new line before the old one ends.
- A Timing consideration applies a multiplier of 0 after the first time the option finishes, with a cooldown of 31,536,000 seconds (one year). This ensures that each line will only play once.

The six follow-on responses are similar, except that they all have a force of 0 and their cooldown is only 20 seconds. Thus they will be chosen at random, but the same line won't be used twice in close succession.

Other reasoners, such as that for the Inconsolable state and the remaining situational responses, have similar configurations, although the details vary. For the Rant state, Marines Enter is always selected first, and then the remaining options are selected as described below.

Tracking the Marines

The FITE effort includes two systems which track the position of all participants, including the Marines. The first is a 6 Degree of Freedom (DOF) system that uses four helmet-mounted cameras to track the position and orientation of the user's head with extreme accuracy. The second is an RFID system which is much less intrusive on the user, but also much less accurate.

We can use this information by setting up "trigger zones" which set predefined values on external variables when one or more Marines are inside of them. Figure 3 shows Granny's room, including the approximate position of her image on the wall and the four caches of contraband, as well as the trigger zones. Those zones are as follows:

- **The Alleyway Zone** covers the gated entryway just outside of Granny's room. When triggered, Granny will periodically play a Marines Knock response with a 20-45 second cooldown.
- **The Entrance Zones** trigger when Marines first enter the room and select the Rant state.
- **The Cold Zone** covers the entire room. When Marines are in this zone (but not in the Warm or Hot Zones) then Granny will play her Cold performance, as described above.
- **The Warm Zone** is an area that is closer to Granny and the major caches. When Marines are in this zone (but not in the Hot zone), Granny will play her Warm performance.
- **The Hot Zones** are placed either right on top of a cache, or right in front of Granny. When Marines are in one of more of these zones, Granny will play her Hot performance.

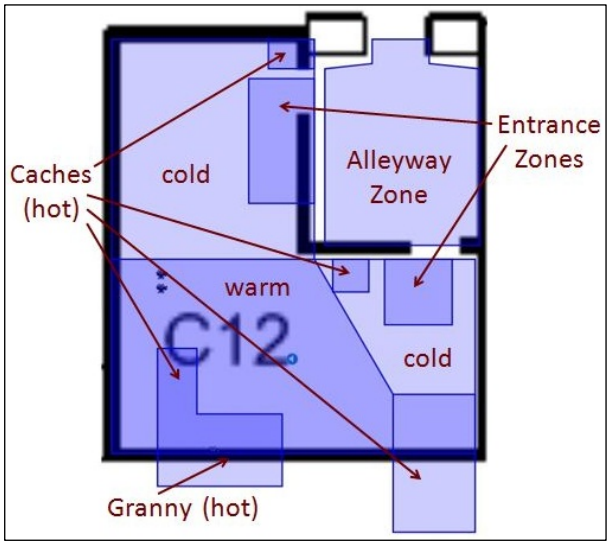

Figure 3: A map of Granny's room, with trigger zones.

The logic driving the state selection triggered by these zones uses the same sorts of consideration configurations as we described above. Validity is checked using External Variable considerations, prioritizing between simultaneously valid options uses Tuning considerations, and we ensure that an option that has just been selected continues for a reasonable amount of time and prevent options from being reselected too quickly using Timing considerations.

Unfortunately, while the tracking systems were tremendous successes in other portions of the effort, this approach was not as successful we might have hoped. There were two major challenges. The first problem was that the more accurate 6 DOF system was only worn by a few participants – typically the squad and team leaders. The RFID system was not only less accurate, but also extremely noisy. As a result, we got nearly constant false positives on the triggers – to the point where Granny would often begin her performance when the Marines were simply walking down the street outside her house, and once they entered she would immediately go into her "Hot" performance regardless of whether a Marine was standing in that area of the room. We experimented with a variety of techniques for addressing the noise, including shrinking the trigger zones, providing gates that have to be triggered first (the Entrance Zones in Figure 3), and smoothing the input, but we only managed to reduce the frequency of false positives – and at the same time, we introduced so much latency that Granny's reactions were markedly late, significantly decreasing believability.

The second problem is that when you have four or five Marines packed into one small room they tend to stand everywhere. Thus the fact that somebody is standing right next to a cache of contraband doesn't necessarily mean that they are close to finding it – it might just be the easiest place to get out of everybody else's way. The 6 DOF system provides detailed information on the location and orientation of the head, which might have been sufficient to overcome this problem if all Marines had been wearing it, but that was not the case. A more robust alternative is discussed in Future Work, below.

Tracking Alternatives

Given the challenges associated with the use of tracking data, we implemented a version of Granny which relies exclusively on operator control for most decisions. This puts a heavy load on the operator to closely track the situation and respond in a timely fashion, but does appear to result in good training.

The one decision that is still not operator controlled is the transition between the Hot, Warm, and Cold states within the Rant performance. We felt that the operator had enough to do, so instead we just allow Granny to choose between these randomly. This is done using a default Tuning consideration, as well as a Timing consideration which applies a higher force for 20-30 seconds after she enters a new state.

ANIMATION SELECTION

Just as we worked hard to avoid repetitive dialog selection, if we want Granny to be believable then we need to avoid repetitive animation selection as well. Animations are expensive to produce, however – much more expensive than lines of dialog – and thus our pool of resources is limited. Fortunately, animation blending technology allows us to select multiple different animations and combine them, creating a performance

that is unique even though the individual animations being played have all been used before.

With this in mind, we created separate base pose, gesture, pacing, lipsynch, blinking, and emotion animations, each of which can be selected independently of the others. Thus if we reuse a gesture animation, for example, it's likely that the pacing animation or the base pose will be different, changing the overall motion of the character. We also use short animations where possible, and use random but reasonable selection to choose among them. Thus when you see a particular gesture –a hand wave, for example, or reaching for the sky while beseeching Allah – the gestures immediately before and after it are likely to be different, again making the overall motion feel different.

Of course, we do have certain lines of dialog that are only used once, and which mark big moments in Granny's performance. The "Marines Enter" performance played when Granny first enters the Rant state is one example, and there's a similar performances the first time Marines Threaten, Marines Talk, or Inconsolable are selected. Since these lines are only used once, it's fine to have a very distinctive full-body animation associated with them – and doing so not only enhances believability by further breaking up any repetition, but also draws the user's attention to these important moments by having her motions become even larger and more dramatic.

Unfortunately, there is not sufficient space to describe the details of animation selection in this paper. In brief, we created several decoupled animation managers, such as the gesture manager and the pacing manager. Each of these used a weight-based reasoner to handle random but reasonable selection of the animations under its control.

RESULTS

We ran a number of squads through the IIT using the FITE technology over the course of this effort, and an independent evaluation was conducted to determine the efficacy of the various technologies.

The evaluation cites the Angry Grandmother as the only system which all participants agreed provided good animation realism. Other systems rated included the animatronics, VISTA screen, and CHAOS room. In addition, they quote one squad leader as saying "the angry [grandmother] acted exactly like women I experienced in country." (Gerrond & Reist 2011)

Another success was the ease of modification. The modular approach taken, as well as the data-driven design of the IOS, allowed us to quickly change the AI and associated controls as the design of the scenario changed. For example, late in the project we were asked to produce an alternate scenario in which Granny had a gun. Building the AI and operator controls took less than two days.

One significant challenge is the expense of creating the audio and animation assets. Although we were able to reconfigure the AI and scenario to support a gun very quickly, for example creating the associated animations took quite a bit longer. This pattern was consistent throughout Granny's development – AI configuration was always much faster than asset creation. In the long run, the obvious solution is to build a library of assets that can be reused across projects. Some libraries exist, but they often don't include the assets that we need, are not of the quality that we need, or put other constraints on the project (such as requiring the use of VBS2).

As discussed above, the tracking data was not as useful as we had hoped. Although this limited the amount of autonomy which we were able to provide within the scope of the JCTD, we have ongoing work which may address this issue.

FUTURE WORK

While we are pleased with our results, there is always more to do. The following are a few areas which we hope to investigate in order to further improve our virtual human technology.

We have been experimenting with the Microsoft Kinect, and early results are promising. We believe that this could ultimately provide more accurate tracking of characters than the RFID system, but without any Marine-carried hardware (such as the helmet-mounted cameras required for the 6 DOF tracking system). Furthermore, we could apply computer vision technology (Yang et. al. 2008) to track head orientation, weapon orientation, the exact moment when a door is opened, and perhaps even changes to the geometry of the room which indicate the progress of the Marines toward finding the contraband caches. This additional data could allow us to automate the majority of Granny's performance, with the exception of the single lines of dialog played during interrogation at the end of the scenario.

Furthermore, we could use voice recognition software to listen for specific phrases from the translator, and use those phrases to drive some of the specific lines of dialog, allowing us to automate the interrogation as well. Any unrecognized questions could be answered with "I don't know!" or a similar line. Parsa, another mixed reality character created by Alelo for FITE

JCTD, showed good results for recognizing a limited set of specific phrases (Saunders 2011).

In both cases, we would still want to keep a human operator in the mix, and allow them to override the AI when appropriate, but this would allow us to automate responses when an operator isn't available, and to make Granny's reactions to big events more immediate.

CONCLUSION

We have presented a game AI-based approach to control of a mixed reality character. It uses a random but reasonable approach to decision making which endeavors to eliminate or disguise reuse of assets, resulting in a non-repetitive, believable character. It mixes autonomous and operator control cleanly, allowing the character to function independently while preserving the ability of the operator to take control if they wish. Where appropriate the AI is utility-based, which allows us to create AI which is highly reactive, evaluating the situation and responding appropriately. Finally, it is built using assets gathered from Afghan natives, resulting in a culturally authentic performance.

ACKNOWLEDGEMENTS

The author would like to thank TEC, ONR, JFCOM, and JIEDDO for their support of this work. In addition, we would like to thank all the participants of the FITE JCTD, and especially the staff of Alelo and Sarnoff who directly contributed to Granny, for their tremendous efforts in making this demonstration a resounding success. Finally, we would like to thank the US Marine Corps and the staff of the IIT for their patience and support in allowing us the use of their facility while still executing a heavy training schedule.

REFERENCES

Dawe, M., Champandard, A., Mark, D., Rich, C., & Rabin, S. (2010). Deciding on an AI Architecture: Which Tool for the Job. *Game Developer's Conference*.

Dill, K. (2008). Embracing Declarative AI with a Goal Based Approach. *AI Game Programming Wisdom 4*. Boston, Massachusetts: Cengage Learning.

Dill, K. (2010). A Pattern-Based Approach to Modular AI for Games. *Game Programming Gems 8*. Boston, Massachusetts: Cengage Learning.

Dill, K., & Dreger, O. (2011). Building an Angry Grandmother. *Spring Simulation Interoperability Workshop*.

Dill, K. & Mark, D. (2010). Improving AI Decision Modeling through Utility Theory. *Game Developer's Conference*.

Dill, K., Rabin, S., & Schwab, B. (2010). AI Architecture Mashups: Insights into Intertwined Architectures. *Game Developer's Conference*.

Gorniak, P. & Davis, I (2007). SquadSmart: Hierarchical Planning and Coordinated Plan Execution for Squads of Characters. *Proceedings, The Third Artificial Intelligence and Interactive Digital Entertainment Conference.*

Harrison, G. (2007). Genetically Programmed Learning Classifier System Description and Results. *Genetic and Evolutionary Computation Conference Proceedings*

Isla, D. (2005). Handling Complexity in the *Halo 2* AI. *Game Developer's Conference* and retrieved April 25, 2011, from http://www.gamasutra.com/gdc2005/features/20050311/isla_01.shtml.

Gerrond J. & Reist J. (2011). Future Immersive Training Environment Joint Capabilities Demonstration Operational Demonstration 2 Independent Assessment Report. *Prepared for the Joint Technology Assessment Activity on behalf of the United States Joint Forces Command*

Mark, D. (2009). *Behavioral Mathematics for Game AI*, Boston, Massachusetts: Cengage Learning.

Muller, P. (2010). The Future Immersive Training Environment (FITE) JCTD: Improving Readiness Through Innovation. *Intraservice/Industry Training, Simulation & Education Conference*.

Orkin, J. (2004). Applying Goal Oriented Action Planning to Games. *AI Game Programming Wisdom 2*. Boston, Massachusetts: Cengage Learning.

Saunders, K. et. al. (2011). Cultural Training in a Mixed Reality Environment. *HSCB Focus*.

Yang, P., Liu, Q., Cui, X., & Metaxas, D.N. (2008). Facial expression recognition using encoded dynamic features. *IEEE Conference on Computer Vision and Pattern Recognition*.