Behavior Authoring

Pritish Sahu,* Roland Gorzkowski,† Ciera Jones,‡ Harrison Truong,§ Kun Wang,¶

Rohan Vernekar,‖ Firas Sattar,** Eric Song,†† Leonard Wohl,‡‡ Tae-Min Kim

Department of Computer Science,

Rutgers University

RUTGERS

Abstract

In this report, we describe various methodologies currently used in behavior authoring. We discuss in detail the difference between their approaches, benefits, and the applications in which they are widely used. We compare and analyze these methods on the basis of performance, complexity, and ease of specification. Finally we propose a sought-after "holy grail" for the field of behavior authoring.

CR Categories: I.3.3 [Computer Graphics]: STAR Topic—CRcat index I.3.7 [Computer Graphics]: STAR Topic—CRcat index;

Keywords: Behavior Authoring, Finite State Machine, Behavior Trees, Behavior Planning, GOAP, Hierarchical Task Network

1 Introduction

The motivational force behind research in behavior authoring is the desire to create a believable world that humans can see and find inspiration in. The key to realizing this is by creating and modeling behaviors after real life actions in a simulated environment. Several real world applications are urban planning, disaster prevention, and security analysis. The final goal of Behavior Authoring is to create meaningful behaviors that others can believe. Before we can reach that goal, we must consider all that goes into making something realistic. When a spectator looks at a behavior, they must believe that it is real and only then will they be able to get inspiration from it. In the world of artificial intelligence and behavior authoring, the elusive "Holy Grail" is to convince the viewer that they are a part of and can interact with the world that is created. When we are able to successfully do this, the viewer can become truly immersed in the world. The "Holy Grail" seems like a daunting task when we account for the more complex interactions between virtual agents and the environment inherent in the behavior of large crowds - for example, slowdown in exit from bottleneck effects when many individuals are rushing out of a narrow area. Managing the many variables and interactions when studying human and crowd behavior can help solve various problems like natural disasters and determining the safest exits of buildings, and can also prove to be quite a challenge.

Interaction is key in the world of behavior authoring, and the implementation used to achieve this is an important factor. For example simple, linear or exponentially branching stories can be created with behavior trees, but a stronger approach might be to allow the viewer to interact with the world through the use of parameterized behavior trees[Shoulson et al. a]. This level of abstraction allows multiple different endings to a story and gives a unique approach to storytelling. Complex interactions like these where the player can meaningfully impact the story are an effective way to make a video game awe-inspiring. When we create a game that has many optional endings or multiple influence-able paths, each player has a unique experience. This is because they are in full control of finding their own story using their player-world interactions as a torch to lead the way. Ideally, each player will be able to journey down their own path and have their own experience. As long as we keep that goal in mind, we can create unique and interesting behaviors that are able to inspire others to do the same. Developing the field of behavior authoring is an ongoing process and as long as research efforts and academia continue to work towards it, we will certainly get closer and closer to that "Holy Grail". We all want to be the leaders of our own lives, and that is what we can do in video games. That is why video games are art: they allow us to choose our own destiny and that is the ultimate goal. The ultimate goal is to cre-

*e-mail:ps851@cs.rutgers.edu
†e-mail:rwg51@scarletmail.rutgers.edu
‡e-mail:cej55@scarletmail.rutgers.edu
§e-mail:ht169@scarletmail.rutgers.edu
¶e-mail:kw423@cs.rugers.edu
‖e-mail:rav84@scarletmail.rutgers.edu
**e-mail:fs263@scarletmail.rutgers.edu
††e-mail:eric.song@rutgers.edu
‡‡e-mail:llw57@scarletmail.rutgers.edu

ate behaviors that are so believable that the spectator can't tell the difference between what is authored and what is reality.

This field has a vast amount potential applications in the gaming, cinema, and academic industries which require studying human and social behaviors for tasks like planning, architecture, and urban design. The popular Unity game engine has already implemented finite state machines and behavior trees for users as tools for implementing agent behaviors. Games like Grand Theft Auto, Halo 2, and Spore have already applied behavior trees and have achieved huge success, while movies like Lord of the Rings have scaled simple agents to allow large armies to act on simple tasks. In academia, researchers have proposed various softwares to build roles, for instance BrainFrame [Daniel and Houlette], "Photoshop of AI" [Hecker] by Chris Hecker , and "Behavior Shop" [Heckel F. W. P.] for non-programmers by Heckel et al where the end user doesn't have to write any code. Researchers are also trying to put all of these together in a tool which includes authoring complicated behaviors where developers can create application in less time. For non-playing characters, Manish Mehta et al. [Mehta Manish] proposed a prototype for authoring behaviors for novice users. Research will always remain slightly ahead of industry practice, because the industry only cares for the techniques which have been tried and tested when developing a product. These techniques that are researched can be evaluated both qualitatively and quantitatively, depending on the goal of the research. When a certain functionality or behavior is highly desired, the research is evaluated qualitatively due to the subjective nature of determining how accurate and realistic a model behaves. When researchers are looking to refine or increase efficiency, the research may be evaluated quantitatively. In this field, researchers are able to improve the world in many ways, such as finding the best way to handle situations such as burning buildings through crowd simulations, as well as improving the way humans interact with computers. In producing believable behaviors, developers are able to better immerse users in their products. However, problems arise such as if artificial intelligence becomes too difficult for normal players to keep up with, which would take away the player's freedom and ruin some enjoyable elements of the product. While researchers are constantly pushing towards smarter behaviors, it may not prove useful for implementation when it comes to developing interesting narratives in games, the field which most benefited from behavior authoring research.

This leads us to the next question: "How can behavior authoring change the world?" There are three main approaches to Behavior Authoring: Agent Centric, Event Centric, and Story Centric. The pros and cons of each approach must be taken into consideration when designing behaviors. The Agent Centric Authoring approach is useful when the designer is looking to create a world where the importance is placed on the individual. Each agent is designed carefully and meticulously to act and react in a believable way. The problem to this approach is that it is very cost heavy because so much care is put into each agent to react as believably as possible. Agents are very complex in this approach and it is difficult to control them effectively and efficiently. That is what leads to the next approach, Event Centric Authoring. This approach focuses on the Event rather than the Agent. With this approach, the designer is looking to create meaningful interactions between Agents. The key to Event Centric Authoring is to create behaviors where Agents come together to accomplish similar goals. This results in behaviors that can only be completed by multiple Agents, such as dancing or talking. The problem with this approach is that the more complex the environment is, the harder it is to create Events. As the complexity of the world increases, so does the number of events that need to be present to make that world believable. That is what leads to the last approach, Story Centric Authoring. This approach is where user interaction comes into play, because once the Agents and Events are in play, the designer is able to author a story that is interesting to the user. If the story is interesting, the user or the player will be more motivated to complete it. If it is boring and predictable, then that motivation is lost. The key to a good story is that anything is possible. If the designer can get the user to believe the unexpected, than they will have the richest experience.

In this report we compare various approaches that have been studied and present how far we have come in achieving the goal. People have tried to construct graph-like data structures using motion capture data [Arikan and Forsyth 2002][L. et al. 2002][Lee and Shin 1999], although these algorithms use large amounts of motion capture data which increases complexity and computation time, and requires a large database. More common implementations in the gaming industry involve move trees [M. et al. 2001]. Various planning algorithm hierarchical task networks [EROL et al. 1994a], cognitive orientated planning[EROL et al. 1994b] are used to search over the behavior states most recent one is A* search.[Lau and Kuffner 2005] to increase efficiency various versions of A* were used, such as truncated A* and inflated A*, as these variations lower the search time. However, quality also suffers - this is the trade off between A* and other versions of A*. Approaches on Planning mechanisms have also been implemented to handle multi-agent interaction such as Scripted Approaches[Loyall 1997a] [Mateas 2002a], usage of smart events[STOCKER et al. 2010], parametrized behavior trees[Shoulson et al. a] Various crowd simulation techniques have also been proposed such as social force models [Helbing and Molnar 1995], reactive behaviors[REYNOLDS 1999], synthetic vision[ONDR EJ et al. 2010] and predictive models[VAN DEN BERG et al. 2008]. Several mechanisms to simulate the cognitive process such as neural networks[Old Tricks and Creatures 1997], partially observable markov decision problems[SI et al. 2005]. The "holy grail" of this field is to author behavior with complex interactions between multiple actors in which the virtual agents can react to a dynamic environment, and learn based on the interactions. Furthermore, the end users should be able to develop applications with having to code complex situations, and domain experts are able to create such actors with ease. The end result is that this should not be too complex to handle in Finite State Machines if there are lots of transitions it becomes complex to handle all the transition phases, or like in Behavior Trees if the tree depth is too large it takes time to check for conditions(internal nodes) and take actions(leaf nodes) based on the conditions.

Our Holy Grail is to provide the user with a fulfilling experience that is as close to reality as possible. Behavior Authoring is our way to provide the user with the experience they are looking for. When we create tools that are easy to use and understand while minimizing cost, it makes the life of the designer easier. Once all these tools are in place, we can provide developers and creative individuals with everything they need to make the Holy Grail possible. As long as we keep reaching, we will get closer and closer to that goal. In the future, we will be able to author behaviors that are so believable that a user or spectator won't be able to tell the difference between reality and simulation. That is the ultimate goal. That is what the Holy Grail represents. If enough research and planning is done, that goal can be accomplished one day, but every goal comes with its set of challenges. Pros and cons must be taken into consideration when designing the perfect behavior. The designer must find the perfect balance in order to make the most believable experience. If the game is too hard, it will stop being fun. If it is too easy or predictable, then it is not believable. If it is too real, then the user becomes uncomfortable because of the uncanny valley. The perfect balance between all things must be found in order to create the perfect behavior.

2 Related Work and Background

This section describes various approaches for Behavior Authoring. They are Behavior Trees, Behavior Planning Approach, Hierarchical Task Networks and GOAP.

2.1 Behavior Planning Approach

Several research efforts have been made to automate real-world motion in the synthetic world. For instance, using static clips like key framing and motion capture[Lee and Shin 1999] [Gleicher] . Other concepts involve writing scripts for animations[Perlin and Goldberg]. Automated behavior control is one of the main challenges which people are still researching. Several methods have been proposed such as storing motion graphs to make the behavior look real[Lee et al.]. In order to make the virtual agents behave more like humans, Huang and Mubbasir [Huang et al. 2013]introduce a sound based propagation and perception system. Virtual narratives can discover and locate unknown backgrounds by identification of virtual voice added to the system. There have also been efforts to author behavior for multimedia by the Human-Computer Interaction Institute at Carnegie Mellon University [Myers] As lack of creation of mature virtual characters, Alexander Shoulson et al. [Shoulson 2013]propose a real-time planning framework to guide interactive characters from virtual humans. Instead of narrative capabilities, they plan with events and then allow high fidelity in synchronizing cooperation of interactive narratives. Besides, Shouldson and Norman[Shoulson and Badler. 2011] also develop an event-centric framework, but they put distribution features into crowed system as random variables to generate different behaviors.

But there's also a cost involved with possibly huge space requirements to store motion data, and failure to adapt to dynamic situations. The daunting task with researchers is how to synthesize real life behavior with minimal memory.

New methods have been proposed using AI where a graph like data structure is used to reuse the motion graphs by dividing them into abstract high level behaviors. Similarly, authoring autonomous characters to populate a simulated environment is a daunting task such as peoples in stadium, life threatening situation in a crowded place. Generally these situation are scripted manually or motion clips are used to define the synthetic characters behavior but there's very few flexibility and complexity. And actors have a knowledge of environment and their actions are limited, complex behaviors depending on dynamic situations are not possible and theirs lot of efforts developers put into scripting every details for the actors. Various planning approaches are used to solve the problem described above using AI, most common among them are A*[Orkin][Lim et al. 2009] and FSM[Lau and Kuffner 2005][Fu and Houlette]. Motion graph data structure[Arikan and Forsyth 2002] generate real life feelings but these are stored in motion clips and constrained with the environment where it can be used and it also takes huge memory to store motion clips. FSM is a faster approach which cuts down the motion clips into high level behaviors. Real time dynamic planning can also be applied on other virtual environments such as computer games and real world simulations. Marcelo and Mubbasir [Kallmann and Kapadia. 2014]provide a navigation structure to find path on virtual world and review previous works. Fu, Houlette, Jensen, and Bascara from Stottler Henke Associates, Inc also provide an alternative visual, object-oriented approach to simulation behavior authoring [Fu et al.].

2.2 Behavior Trees

A Behavior Tree is a plan organized in a tree data structure. These plans are dynamic, since each branch of the tree is reached depending on the state of its parent. The parent nodes in the tree represent goals and the left child checks a condition. On meeting the condition, the right child is followed which represents another goal action. Each node attempts to execute an action and reports success or failure to its parent node. They are easily scalable, and because each node represents an inividual goal-oriented action, they are also highly modular. Sub-trees function independently as smaller goals and can be used by other similar tasks.[Lim et al. 2009] As a framework, behavior trees are commonly implemented for use in agent controllers in interactive game environments, and interactions between groups of actors are often defined using parametrized behavior trees. Research is done on where to use the Behavior Tree, (AI behaviors for example), and how to make it more efficient. Many times behavior trees are used in programming AI for video games, seen in Delmer's implementation to control an AI player for a Real Time Strategy game [Delmer 2012], as well as Lim, Baumgarten, and Colton's implementation for the game DEFCON[Lim et al. 2009].

2.3 Goal Oriented Action Planning

Goal Oriented Action Planning (GOAP) is an AI system used to let agents intelligently determine a series of actions to complete a goal. Actions in GOAP must include cost, precondition and consequence. An action can only be executed if its preconditions are met and once it is executed, its consequences are applied to the system. GOAP uses this to construct a valid, low cost series of actions that accomplishes the given goal. GOAP improves on Finite State Machines by making it much easier to add, remove, and maintain actions. The modularity of actions in GOAP allows for more dynamic and complex agents when compared to FSM. One application is the research done at MIT by Bruce Blumberg, where he tried to create a virtual brain that rivals the complexity of a dog [Burke et al. 2001]. GOAP is also used in many games [Orkin 2003] [Bjarnolf 2008] and autonomous robots [Kolbe 2013]

2.4 Hierarchical Task Networks

Hierarchical Task Networks (HTN) are another type of plan, and each step in the plan may have another series of steps. This kind of network works well when tasks easily are organized in a hierarchy. This kind of network can speed up planning, since search is guided with human knowledge. This is especially important for behavior authoring since it facilitates authoring done in real time at a fast rate. The research into this related to crowd structure, and how hierarchy relates to the crowd, and the complex structures that make the crowd dynamic[Thalmann and Raupp Musse 2001].

3 Behavior Planning Approach

3.1 Finite State Machines

3.1.1 Introduction

Finite state machines are a wonderful option when tasked with creating some form of AI in a game or simulation. The way a finite state machine works is by making each individual action into a state and then connecting these states with transitions that show the flow of the character's thought process. The states are similar to nodes in a graph and the transitions between them can be considered as the edges. Finite state machines' main use comes in the form of turning this organization and representation of flow into a form of AI for other characters within the game or simulation which the player does not have to make the decisions.

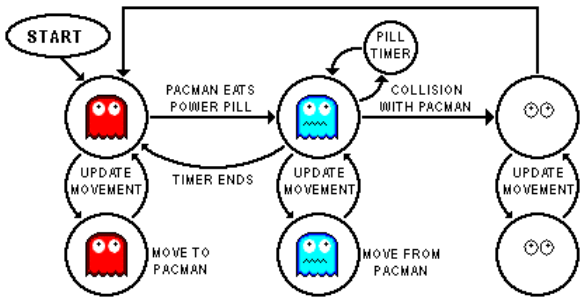

Figure 1: *Finite State Machine*

3.1.2 Applications

Finite state machines are not merely limited to being used as a decision maker for an opponent of the player. It can be used in any situation that can be represented by a limited number of conditions or otherwise known as states and can transition between them based upon a fixed set of rules. For example, if someone wanted to implement a choose your own adventure style game then using a finite state machine would be perfect as it could keep track of the player's choices and transition to the next appropriate point of the story.

Industrial/Commercial Applications: Finite state machines are not limited to being used within video games. There are a lot of situations where finite state machines are implemented one such example could be when a person goes shopping online. They are first in the state of browsing, then they move on to putting things in their shopping cart, then move on to putting in their payment information and paying. All of these can be also moved between very easily due to the simple nature of finite state machines.

3.1.3 Features

Finite State Machines are states representing high level behavior. Only a single state can be active at the same time. It is basically represented as a graph and each transition from state to state has a label like "IsNear". Every state has a particular motion clip embedded in it which starts to play when the current state is active.

3.1.4 Challenges

FSM's are easy to understand and easy to implement in a small scale project, but their complexity and branching factors increase exponentially in a large scale project. Various challenges like running multiple states, time taken for transition from several sequential nodes, increasing complexity are the main ones which needs optimal solutions.

3.1.5 Benefits and Trade-offs

Though finite state machines are simple to implement when dealing with a small number of states this gets exponentially more complex as more states are added. The problem is that whenever work is being done to modify or add to a finite state machine it needs to be explicitly connected to all of it's necessary transitions at a low level and due to the high number of states people can get lost in the complexity. Another problem is the fact that finite state machines lack standardization. Implementations are done for very specific situations and there doesn't seem to be an overarching implementation that can be easily modified to fit most needs. The main huge drawback for finite state machines is that they lack concurrency. Trying to run multiple in parallel could easily lead to deadlocks and other problems. For example if the finite state machine was implemented using switch statements then state would be affected by all of the concurrently running finite state machines. The positives of finite state machines though is that they can easily be visualized with their implementation and work perfectly fine on small scale jobs that do not involve concurrency.

3.1.6 Conclusion

Finite state machines have usually been the go-to for implementations of AI and in games, but it can be seen that they have started to show their age as the scale of games have grown larger and the interactions between characters is not simply limited to small deterministic cases. Finite state machines do not work well anymore as means of representing AI due to the fact that thoughts work more along the lines of many choices being presented to the person and each one having a certain weight to them, Any of them can be chosen and the situation advances based upon that choice. Rather than just waiting for an event to occur so that the transition to the next state can occur.

3.2 Goal Oriented Action Planning

3.2.1 Introduction

GOAP, Goal-Oriented Action Planning, is a behavior authoring method in which actions are planned at run time rather than predefined through a method such as Finite State Machines. Actions within GOAP are defined with preconditions, effects, and costs. Preconditions are conditions that must be fulfilled in order for an action to be executed. Effects define how the environment is changed after the action executes. Costs define how expensive it is to execute an action. This metric is necessary for the GOAP planner to find an optimal series of actions. When the GOAP planner is given a goal, it analyzes the current environment to figure out which actions are available. It then uses a path finding algorithm such as A* to figure out an optimal series of actions (path) in order to fulfill its goal. GOAP was developed by Jeff Orkin during the production of F.E.A.R. in 2005. The system was largely inspired by STRIPS, Stanford Research Institute Problem Solver, an automated planner developed by Richard Fikes and Nils Nilsson in 1971.

3.2.2 Features

GOAP offers a modular approach to behavior authoring. It can offer similar capabilities of a complex Finite State Machine with much more maintainability and scaling. It's ease of development lends itself to faster, bug free code and enables developers to add more complex behavior.

3.2.3 Benefits and Trade-offs

The advantages and disadvantages will be discussed in comparison to Finite State Machines as they are still the prevalent method for behavior authoring. The key advantage that GOAP brings is ease of development. GOAP is inherently modular since it is defined primarily through its actions. Adding new behavior to GOAP is simply defining and adding new actions. New actions do not have to be aware of existing actions so the complexity of adding new behavior stays the same, even as the system grows. This is very different from FSM, where adding new behavior becomes an increasingly complex task as the system grows. The ease of adding new behavior to GOAP naturally leads to other advantages such as more varied and complex behavior.

Another key difference is in how the agent executes at runtime. FSM must write out how an agent will behave in every situation whereas GOAP will let the agent decide its own actions at runtime.

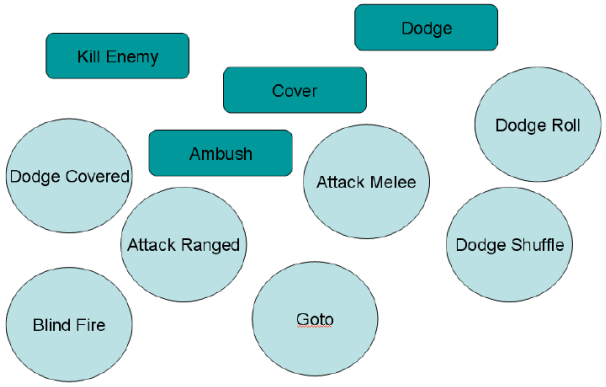

Figure 2: *Goal Oriented Action Planning*

Jeff Orkin makes the analogy of comparing them to pre-rendered and real-time rendered graphics, "If we render in real-time, we can simulate the effects of lighting and perspective, and bring the gaming experience that much closer to reality. The same can be said of planning. By planning in real-time, we can simulate the effects of various factors on reasoning, and adapt behavior to correspond." Some debated weaknesses of Goal Programming as an approach are that its solutions are not Pareto efficient and assigning appropriate costs can be quite difficult. However, there are several proposed solutions for both weaknesses. Moreover, for our purposes of creating realistic agents, the most efficient series of actions is not strictly necessary as real life agents do not perform perfectly. Case Study: The following sections discuss the effectiveness of GOAP in F.E.A.R, one of the first games utilizing goal programming. The information is from the paper, "Three States and a Plan: The A.I. of F.E.A.R". The developers of F.E.A.R. concluded that there were three benefits from using GOAP: Decoupling Goals and Actions, Layering Behaviors, and Dynamic Problem Solving. Decoupling of Goals and Actions: One of the biggest benefits they experienced from using GOAP was the ease of adding a new character even at a very late stage in development. The modularity of actions made it so that they could create new characters by pulling actions from existing agents. In one scenario, the developers had to create a flying drone character and they were able to do so by combining the aerial movement actions from the existing ghost character and the tactical fighting actions from the existing soldier character. Another benefit to decoupling goals and actions was that it forced the goals to share an external working space. Previously, goals were self contained black boxes that had limited communication between each other. Layering Behaviors: The developers of F.E.A.R. added complex behaviors to their agents by taking a layered approach. They first specified the most action behavior their agent should exhibit such as "KillEnemy", and then added more layers such as "Dodge" if they were under pressure or "AttackMelee" if they were in suitable range of their target. For each new layer and behavior that they added, all they had to do was specify a few actions. New layers didn't require testing its interaction with any existing behaviors.

Dynamic Problem Solving: GOAP also let their A.I. react to far more different scenarios. When a human player is added, the number of scenarios an agent might have to react to increases greatly. Since GOAP computes its actions at runtime, the number of possible issues that it might have to handle is a non issue.

3.2.4 Evaluation

Creating a realistic dialogue experience proved to be a difficult task, even when creating dialogue as GOAP actions. There's additional complexity to be considered when an agent can fulfill a goal by either executing the necessary actions itself or telling another agent to fulfill the goal. Depending on the environment, it may be difficult to figure out which agent can fulfill the goal at the smallest cost and for the agents to communicate these factors effectively. This is an example of one of GOAP's weaknesses, the difficulty in assigning appropriate costs for complex systems. GOAP was also not capable of creating complex A.I. that could cooperate with human players. A.I. were able to cooperate with each other since their set of goals and actions were known and predictable. When an unpredictable element such as a human player is added, GOAP agents can only react to the player's actions. In order to cooperate with the player at a deeper level, the A.I. must be able to analyze the player's actions, predict what the player is trying to accomplish, and execute actions that cooperate with the predicted actions. This is a layer of complexity that GOAP was not able to solve for F.E.A.R.

3.2.5 Conclusion

For F.E.A.R, GOAP's primary contribution was in easing the development process for creating complex A.I. Critics positively claimed that F.E.A.R. had a great A.I which reminded them of agents from Half Life 1, a game shipped in 1998. On the surface, it wasn't clear if F.E.A.R. was able to implement A.I. that were more advanced than existing systems. However, the modular approach that GOAP provides gives developers hope that the shortcomings listed above are problems that can soon be solved.

3.3 Behavior Trees

3.3.1 Introduction

Behavior trees are directed acyclic graph structures which provide a manner for designers to plan, organize, and define artificially intelligent control of agents and scenarios. Each behavior tree itself is simple, scalable, and modular, and is oriented toward a specific goal. By combining behavior trees, one can link together the individual goals to create a complex scenario. Behavior trees have several applications, both in research and industry when artificial intelligence is involved. In the field of research, researchers are searching for both new ways to utilize behavior trees, and how to make them more efficient. Successful utilization of behavior trees include authoring multi-actor behaviors in crowds with diverse personalities[Kapadia et al. 2013a], creating artificial intelligence players for games such as DEFCON[Lim et al. 2009],and creating artificial intelligence players for real time strategy games[Delmer 2012]. Researchers have also done work on improving behavior trees, developing techniques such as parameterizing[Shoulson et al. a], and evolving[Lim et al. 2009]. In the industry, behavior trees are very commonly utilized for artificial intelligence such as in the games Halo 2 and Spore. Along with this, as uses in the field of research would suggest, behavior trees can be utilized to make simulations of scenarios which can help to plan the infrastructure of buildings or possible scenarios to prepare for at an event. Behavior trees are not yet found in many different places, however, research is still being done to see where they can be used effectively.

3.3.2 Features

Behavior trees are constructed from primitive constructs and composite constructs – primitive constructs can be actions or conditions (leaf nodes), while composite constructs are used to group the primitive constructs together using sequences, selectors, and decorators. Behavior trees are dynamic by nature, allowing for easy additions and deletions of both simple actions and complex scenarios to the overall story.

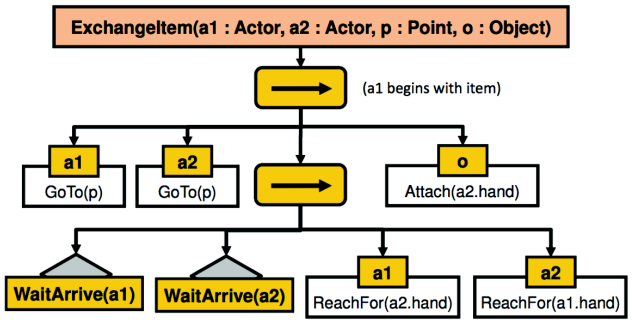

Figure 3: *Behavior Tree*

3.3.3 Benefits and Trade-offs

There are many benefits of using behavior trees over other methods of behavior authoring. Behavior trees are organized in a manner where states and decision logic are separate – each state can be run separately, allowing for easy testing and implementation. Along with this, it has a weak dependency which allows for easy additions and deletions of behaviors without requiring modification of another one. With behavior trees' flexibility, they can be very simple and customizable for both the original designer and any other developers who pick up the work. However, behavior trees have a high performance cost and are difficult to optimize so they are not always the best option depending on the use case. And while behavior trees may be easy to understand and use, the logic can seem not intuitive to somebody who is trying to learn the system, especially with the use of parallel nodes.

3.3.4 Evaluation

Behavior trees can be evaluated both qualitatively and quantitatively. Aspects of behavior trees such as smoothness and believably can be evaluated qualitatively, and efficiency can be tested quantitatively through comparison of run times and costs. In finding new possible implementations of behavior trees, one can test the trees' effectiveness in comparison to other trees (regular behavior trees vs. Parameterized behavior trees), as well as other methods such as GOAP and HTN, and test the data both qualitatively (seeing the results of simulations) and quantitatively (analyzing data such as cost and efficiency). But when researching the use of behavior trees, there are two main challenges which must be confronted – where behavior trees can be used effectively, and how behavior trees can be made more effective. While there have been some answers to these questions, they are still open problems which may continue to be researched, as there can be multiple answers for each one, and no way to tell whether behavior trees can be improved beyond their current state or not.

3.3.5 Conclusion

Behavior Tree are simple and customizable. They have more speed over FSM and other authoring models, as it moves in branch of the tree so doesn't do a complete search.

3.4 Hierarchical Task Networks(HTN)

3.4.1 Introduction

Hierarchical Task Networks are an automated planning approach in which dependencies among actions are represented as networks. HTN imposes hierarchical, sequential, and conditional structure on behaviors by breaking complex tasks down into sub-tasks with potential constraints. Constraints on tasks are implemented as prerequisite conditions which must be met before an action is performed.

3.4.2 Applications

HTN is an automated planning algorithm, so it is applicable to almost any situation that desires automated dynamic planning. This includes everything from video game character AI, to emergency response plans, to drone control. The industrial applications of HTN are numerous. HTN could be implemented in order to create dynamic plans for construction projects, where large tasks (like "build a house") could be broken down into simpler tasks (like "set foundation"), with prerequisites (like "have permits approved"). On the opposite end of the spectrum, HTN could be used to implement AI for video game characters or drones, which could make dynamic plans based on their current goal. A simple goal could be "Get to the other side of the river", which might break down into sub tasks like "Build a boat" with prerequisites like "Collect wood". HTN has been implemented in emergency response systems(like SIADEX) to dynamically create evacuation and control systems. Using knowledge of the environment coupled with the state of the emergency at hand, HTN can produce a specific evacuation plan for an emergency. If, for example, the front door of a burning building is blocked by flaming debris, HTN could create a plan which routes the evacuation through other escapes, rather than putting evacuees in harm's way.

3.4.3 Features

HTN requires a large library of complex actions, which break down into simpler primitive tasks.For large scale projects compiling this library is a difficult task. The library must be constructed by experts with the relevant knowledge, and is highly error prone due to this human implementation. The holy grail of Hierarchical Task Networks would be an immense library of complex and primitive actions that would apply to many different use-cases. Such a library would be nearly impossible to compile due to its unrealistic size and complexity

3.4.4 Benefits and Trade-offs

Because each action can be broken down into a series of other actions, HTN are incredibly flexible. Given a proper library of domain knowledge, HTN can create dynamic tasks for many situations, making the creation of the actual plan very simple. This comes at the cost of overhead. The domain knowledge for the planner must be detailed and expansive enough to give it sufficient actions with which to plan any number of tasks; this is a daunting task that requires time and expertise.

3.4.5 Evaluation

HTN are primarily evaluated quantitatively based on their effectiveness at producing a variety of plans with different goals and conditions.Incomplete knowledge bases would not allow HTN to effectively produce plans for certain cases.

3.4.6 Future Work

The main problem with HTN to be addressed by the community in the future is the difficulty in constructing/maintaining the knowledge base required for making plans. Possible solutions have been suggested to solve the problems concerning the development and

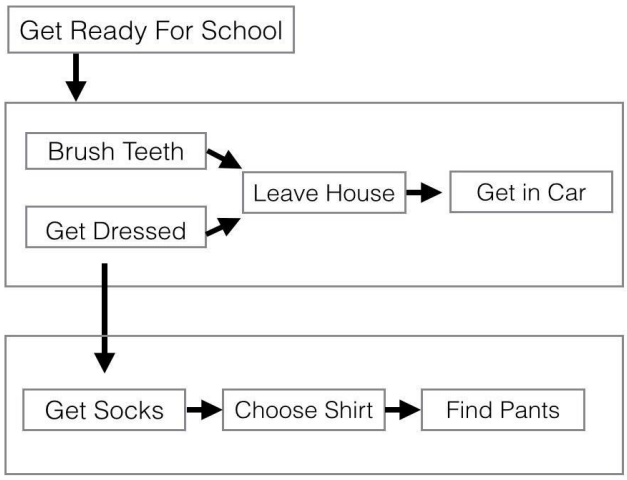

Figure 4: *Hierarchical Task Networks*

maintenance of the knowledge base from which plans are generated. One idea is crowd sourcing the library; or creating a method with which users or other interested individuals could add to the knowledge base themselves. Much like the recent success of the "Kickstarter" website, this would spread the burden of implementation to thousands of people rather than a small development team. Given enough incentive, this would allow a much larger group to work towards a single, expansive library, making the "Holy Grail" a much more achievable goal. Alternatively, some research has been put towards automating the creation of the knowledge base through machine learning by providing a machine learning algorithm access to expert behavior for a set of sample situations, the algorithm could generalize the actions performed into solutions for other tasks, and add new knowledge to the library without the need for human implementation. This approach, however, comes with its own difficulties; learned behaviors can be limited or poorly learned, and without enough resources to learn from it can be impossible to complete many complex tasks.

3.5 Conclusion

The Holy Grail was discussed extensively in our introduction, because that is what we are trying to reach, but before we can get there we must find balance in all things, and that is an extremely difficult task. Each approach has its advantages and disadvantages and everything must be considered in order to find the best approach. Nothing should be ignored when it comes to finding a Holy Grail, because as long as the designer considers the advantages from everything they have learned, they will be able to achieve the Holy Grail. Costs and disadvantages must also be taken into consideration, because no advantage comes without a cost. Every method has its own purpose and even though it might approach perfection when it comes to achieving that specific purpose, there is always a larger purpose. If this is taken into consideration, then the Holy Grail is impossible, but that is why we consider it Holy. As long as research and academia keep their eye on what is important to them today, individuals in the future will be able to take from them and expand upon it in order to build bigger and better behaviors. Behavior Authoring is a way for humans to find inspiration from a simulation. We inspire to create realistic and believable models of humanity, because we are all trying to find ourselves. The subject of replicating humanity using computers is one of great interest as well as controversy. It is the idea of computer one day being able to perfectly replicate humanity that scares many people away from that idea, because they are afraid of the unknown, but that is merely a false construction of the human mind. It is the doubt in our own abilities that restricts us from achieving that goal. If man believed that creating a perfect simulation of the human mind within an Artificial Intelligence was possible and necessary, we would have already done it. The human race is the strongest when we are struggling and that is why must continue to struggle to achieve the Holy Grail no matter how far away it is.

Our solution to finding the Holy Grail is to take the most out of today in order to make a better tomorrow. GOAP, Behavior Trees, and HTN are the three main methods discussed in our paper, but it is not hard to find a fourth or a fifth. As long as people are willing to always go one step beyond, and create better and more powerful solutions to the problem, the Holy Grail can be achieved. Our Grail is constantly changing depending on what matters most to us. For GOAP, the Grail was to design an approach that was simple to change and allow for unique behaviors through defining goals. This leads to Behavior Trees, where the simplicity of modulation was kept and the ability to allow for better user interaction was added with the use of Parameterized Behavior Trees. However, once Behavior Trees become too complex, they can be difficult to understand and use. The ease of use is extremely important when it comes to finding the perfect solution. If Behavior Authoring was capable of being understood by anybody, then more people would find it interesting and meaningful. More people will be able to apply their talents to a common goal. Give a man a fish and it will feed him for the day. Teach him how to fish and it will feed him for a lifetime. That is the secret to achieving the Holy Grail.

4 Tables

A comparison of various works in the field of Behavior Authoring

References

ARIKAN, O., AND FORSYTH, D. A., 2002. Interactive motion generation from examples.

BECROFT DAVID, E. A., 2011. Aipaint: A sketch-based behavior tree authoring tool.

BJARNOLF, P. 2008. *Threat Analysis Using Goal-Oriented Action Planning*. Master's thesis, University of Skövde.

BRAUN, A., MUSSE, S. R., DE OLIVEIRA, L. P. L., AND BODMANN, B. E. J. Modeling individual behaviors in crowd simulation. http://gamma.cs.unc.edu/LARGE/papers/individual_behaviors.pdf.

BURKE, R., ISLA, D., DOWNIE, M., IVANOV, Y., AND BLUMBERG, B., 2001. Creature smarts: The art and architecture of a virtual brain. http://characters.media.mit.edu/Papers/gdc01.pdf.

DANIEL, F., AND HOULETTE, R. Aipaint: A sketch-based behavior tree authoring tool.

DELMER, S., 2012. Behavior trees for hierarchical rts ai. http://www.smu.edu/~/media/Site/guildhall/Documents/Theses/Delmer_Stephan_Thesis_Final.ashx?la=en.

DILLER, D. E., FERGUSON, W., LEUNG, A. M., BENYO, B., AND FOLEY, D. Behavior modeling in commercial games. http://seriousgames.bbn.com/behaviorauthoring/BRIMS_Behavior_Authoring_in_Games_2004.pdf.

Approaches	Ease of Specification	Performance	Flexibility
Behavior Tree	✓	X	✓
FSM	X	X	X
GOAP	X	X	✓
HTN	✓	X	✓
Behavior Tree + FSM	✓	X	✓
Holy Grail	✓	✓	✓

EROL, K., HENDLER, J., AND NAU, D. S., 1994. Htn planning: complexity and expressivity.

EROL, K., HENDLER, J., AND NAU, D. S., 1994. Htn planning: complexity and expressivity.

ET AL, D. D. E. Behavior modeling in commercial games.

FIKES, R. E., AND NILSSON, N. J. Strips: A new approach to the application of theorem proving to problem solving. http://ai.stanford.edu/~nilsson/OnlinePubs-Nils/PublishedPapers/strips.pdf.

FU, D., AND HOULETTE, R. The ultimate guide to fsms in games.

FU, D., HOULETTE, R., JENSEN, R., AND BASCARA, O. A visual, object-oriented approach to simulation behavior authoring. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.128.1633&rep=rep1&type=pdf.

FUNGE, J., TU, X., AND TERZOPOULOS, D., 1999. Cognitive modeling: knowledge, reasoning and planning for intelligent characters.

GLEICHER, M. Retargetting motion to new character. http://ai.stanford.edu/~latombe/cs99k/2000/gleicher.pdf.

HECKEL F. W. P., YOUNGBLOOD G. M., H. D. H. Behaviorshop: An intuitive interface for interactive character design.

HECKER, C. Structure vs. style.

HELBING, D., AND MOLNAR, P., 1995. Social force model for pedestrian dynamics.

HOANG, H., LEE-URBAN, S., AND MUNOZ-AVILA, H., 2005. Hierarchical plan representations for encoding strategic game ai.

HUANG, P., MUBBASIR, K., AND BADLER., N. I., 2013. Spread: Sound propagation and perception for autonomous agents in dynamic environments.

IGARASHI, T. Digital-content authoring. *IEEE Computer Graphics and Applications*.

KALLMANN, M., AND KAPADIA., M., 2014. Navigation meshes and real-time dynamic planning for virtual worlds.

KAPADIA, M., SINGH, S., AND OTHERS. A behavior authoring framework for multi-actor simulations. http://www.cs.rutgers.edu/~mk1353/pdfs/2011-cga-behavior.pdf.

KAPADIA, M., SHOULSON, A., DURUPINAR, F., AND BADLER, N., 2013. Authoring multi-actor behaviors in crowds with diverse personalities.

KAPADIA, M., SHOULSON, A., DURUPINAR, F., AND BADLER, N. I., 2013. Authoring multi-actor behaviors in crowds with diverse personalities.

KELLY, J.-P., BOTEA, A., AND KOENIG, S. Planning with hierarchical task networks in video games. http://icaps07-satellite.icaps-conference.org/workshop8/Planning%20with%20Hierarchical%20Task%20Networks%20in%20Video%20Games.pdf.

KOLBE, F. 2013. *Goal Oriented Task Planning for Autonomous Service Robots*. Master's thesis, Hamburg University of Applied Sciences.

L., K., M., G., AND F., P., 2002. Motion graphs.

LAU, M., AND KUFFNER, J. J., 2005. Behavior planning for character animation. http://graphics.cs.cmu.edu/projects/behavior_planning/behavior_planning_sca05.pdf.

LEE, J., AND SHIN, S. Y., 1999. A hierarchical approach to interactive motion editing for human-like figures.

LEE, J., CHAI, J., REITSMA, P. S. A., HODGINS, J. K., AND POLLARD, N. S. Interactive control of avatars animated with human motion data. http://graphics.cs.cmu.edu/projects/Avatar/avatar.pdf.

LEMERCIER, S., JELIC, A., KULPA, R., HUA, J., FEHRENBACH, J., DEGOND, P., APPERT-ROLLAND, C., DONIKIAN, S., AND PETTRÉ, J. 2012. Realistic following behaviors for crowd simulation. *Computer Graphics Forum*.

LIM, C.-U., BAUMGARTEN, R., AND COLTON, S. Evolving behaviour trees for the commercial game defcon. http://ccg.doc.gold.ac.uk/papers/lim_evogames10.pdf.

LIM, C.-U., BAUMGARTEN, R., AND COLTON, S., 2009. An a.i. player for defcon: An evolutionary approach using behavior trees. http://www.doc.ic.ac.uk/teaching/distinguished-projects/2009/c.lim.pdf.

LOYALL, A. B., 1997. Believable agents: building interactive personalities.

LOYALL, A. B., 1997. Believable agents: Building interactive personalities. https://www.cs.cmu.edu/Groups/oz/papers/CMU-CS-97-123.pdf.

M., M., J., B., AND T., C., 2001. Data driven motion transitions for interactive games.

MATEAS, M., 2002. Interactive drama, art and artificial intelligence.

MATEAS, M., 2002. Interactive drama, art and artificial intelligence. http://www.cs.cmu.edu/~dod/papers/CMU-CS-02-206.pdf.

MEHTA, M., AND CORRADINI, A., 2014. An approach to behavior authoring for non-playing characters in digital games, jun.

MEHTA, M., ONTANON, S., AMUNDSEN, T., AND RAM, A. Authoring behaviors for games using learning from demonstration. http://gaia.fdi.ucm.es/sites/cbrcg09/iccbr09_submission_89.pdf.

MEHTA MANISH, A. C. An approach to behavior authoring for non-playing characters in digital games.

MYERS, B. A. Authoring interactive behaviors for multimedia. http://www.cs.cmu.edu/~bam/papers/necjapandoc.pdf.

NAU, D. S. Current trends in automated planning. http://www.aaai.org/ojs/index.php/aimagazine/article/view/2067/2054.

OLD TRICKS, N. D. E., AND CREATURES, I. 1997. *Bruce Mitchell Blumberg*. PhD thesis, MASSACHUSETTS INSTITUTE OF TECHNOLOGY.

ONDR EJ, J., PETTRE, J., OLIVIER, H., A., AND DONIKIAN, S., 2010. A synthetic-vision based steering approach for crowd simulation.

ORKIN, J. Three states and a plan: The a.i. of f.e.a.r. http://alumni.media.mit.edu/~jorkin/gdc2006_orkin_jeff_fear.pdf.

ORKIN, J., AND ROY, D. Automatic learning and generation of social behavior from collective human gameplay. http://www.media.mit.edu/cogmac/publications/orkin_aamas2009.pdf.

ORKIN, J., SMITH, T., RECKMAN, H., AND ROY, D. Semi-automatic task recognition for interactive narratives with eat and run. http://web.media.mit.edu/~dkroy/papers/pdf/fdg_2010_camera_ready.pdf.

ORKIN, J., 2003. Applying goal-oriented action planning to games. http://alumni.media.mit.edu/~jorkin/GOAP_draft_AIWisdom2_2003.pdf.

PELECHANO, N., ALLBECK, J., AND BADLER, N., 2008. Virtual crowds: Methods, simulation, and control.

PELECHANO, N., ALLBECK, J., AND BADLER, N. 2008. Virtual crowds: Methods, simulation, and control. *Synthesis Lectures on Computer Graphics and Animation*.

PERLIN, K., AND GOLDBERG, A. Improv: A system for scripting interactive actors in virtual worlds. http://mrl.nyu.edu/publications/sig96-improv/sig96-improv.pdf.

REYNOLDS, C., 1999. Steering behaviors for autonomous characters.

RILLING, S., AND GRAESER, S., 2011. Authoring dynamic object behavior within virtual environments, jul.

SHOULSON, A., AND BADLER., N. I., 2011. Event-centric control for background agents.

SHOULSON, A., GARCIA, F. M., AND OTHERS. Parameterizing behavior trees. http://www.seas.upenn.edu/~shoulson/papers/2011-mig-pbts.pdf.

SHOULSON, A., GILBERT, M. L., KAPADIA, M., AND BADLER, N. I. An event-centric planning approach for dynamic real-time narrative. http://www.seas.upenn.edu/~shoulson/papers/2013-mig-event-planning.pdf.

SHOULSON, A., MARSHAK, N., KAPADIA, M., AND BADLER, N. I. 2014. Adapt: The agent development and prototyping testbed. *IEEE Transactions on Visualization and Computer Graphics 20*, 7 (July/Apr.), 1035–1047.

SHOULSON, ALEXANDER, E. A., 2013. An event-centric planning approach for dynamic real-time narrative.

SI, M., MARSELLA, S. C., AND PYNADATH, D. V., 2005. Thespian: An architecture for interactive pedagogical drama.

STOCKER, C., SUN, L., HUANG, P., QIN, W., ALLBECK, J. M., AND BADLER, N. I., 2010. Smart events and primed agents.

THALMANN, D., AND RAUPP MUSSE, S., 2001. Hierarchical model for real time simulation of virtual human crowds.

VAN DEN BERG, J., LIN, M. C., AND MANOCHA, D., 2008. Reciprocal velocity obstacles for real-time multi-agent navigation.

VILHJALMSSON, H., CANTELMO, N., CASSELL, J., CHAFAI, N. E., KIPP, M., KOPP, S., MANCINI, M., MARSELLA, S., MARSHALL, A. N., PELACHAUD, C., RUTTKAY, Z., THÓRISSON, K. R., VAN WELBERGEN, H., AND VAN DER WERF, R. J., 2005. Procedural authorship: A case-study of the interactive drama facade. http://www.interactivestory.net/papers/MateasSternDAC05.pdf.

VIRMANI, S., KANETKAR, Y., MEHTA, M., ONTANON, S., AND RAM, A. An intelligent ide for behavior authoring in real time strategy games. http://www.cc.gatech.edu/faculty/ashwin/papers/er-08-08.pdf.

WEBER, B. G., MATEAS, M., AND JHALA, A., 2010. Applying goal-driven autonomy to starcraft.

YOUNG, J., IGARASHI, T., SHARLIN, E., SAKAMOTO, E., AND ALLEN, J. Design and evaluation techniques for authoring interactive and stylistic behaviors.