Employing Fuzzy Concepts for Digital Improvisational Theatre

Brian Magerko, Peter Dohogne, and Chris DeLeon
Georgia Institute of Technology
{magerko, pdohogne3, cdeleon3}@gatech.edu

Abstract

This paper describes the creation of a digital improvisational theatre game, called *Party Quirks*, that allows a human user to improvise a scene with synthetic actors according to the rules of the real-world *Party Quirks* improv game. The AI actor behaviors are based on our study of communication strategies between real-life actors on stage and the fuzzy concepts that they employ to define and portray characters. This paper describes the underlying fuzzy concepts used to enable reasoning in ambiguous environments, like improv theatre. It also details the development of content for the system, which involved the creation of a system for animation authoring, design for efficient data reuse, and a work flow centered on Google Docs enabling parallel data entry and rapid iteration.

Introduction

Improvisational theatre (or "improv") has been a source of inspiration for approaches in interactive narrative since Hayes-Roth and Perlin's works in the early 1990s (Hayes-Roth and Van Gent 1996; Perlin and Goldberg 1996). More recent approaches have similarly used aspects of improvisational acting, such as the concept of *status* (i.e. how powerful or meek a character is on stage) (Harger 2008) or object creation (i.e. introducing objects in the world that have not been explicitly stated as not existing there) (Swartjes, Kruizinga, and Theune 2008), to drive interactive narrative works. However, these approaches have been limited in scope because we lack a formal understanding of the processes involved in improvisational acting. In other words, it is difficult to build computational improv actors because we do not understand the phenomenon of improv acting well enough.

The work presented in this paper introduces the knowledge and processes related to character creation that we observed in our study of the cognitive processes involved in human improvisation (Magerko et al. 2009; Fuller and Magerko 2011). Our study of human improvisers has involved actors engaging in different improv games at our request, both in lab settings and in professional theatre performances. After the engaging in these games, we interviewed actors using retrospective protocol, group, and, in the case of a professional performance, semi-structured interview techniques. A grounded theory (i.e. data-driven instead of hypothesis-driven) approach to data analysis has uncovered several main facets of improvisation that are relevant to building computational agents. The first is the concept of how improvisers build shared mental models (i.e. how actors "get on the same page" during a scene), which is also called building *cognitive consensus*. We have explored this phenomenon formally in other work and have found that it is a ubiquitous aspect of improvisation. Actors are perpetually trying to reach a shared understanding on stage between each other and between themselves and the audience. The second is the construction of narrative on stage. Many improv games have a strong storytelling component, which results in the improvisers collaborating in real-time on stage to develop characters, a relationship between them, dramatic conflicts, etc. as part of the improvised performance in front of an audience. A third major aspect of our findings is the *referents* used by improvisers (i.e. the constraints for a scene and tacit knowledge about improvisation that they employ), which has so far been beyond the scope of our work in terms of rigorous analysis or computational modeling.

Both the general improv process of constructing shared mental models and the more specific process of improvising narratives (which often involves creating shared mental models) require handling ambiguity on stage. If an actor A comes on stage, peering at something, and saying "Hrrrmmm, very interesting, very interesting…" another actor B may come on stage and interpret the utterance as A indicating that they are a clinician, or perhaps a scientist, or even a detective. Improvisation is a continuous practice of instantiating and interpreting symbols without clear mappings for those symbols. As improvisers work on building a shared mental model, they make progress on agreeing on said mappings (e.g. who is playing what character, where they are, what they are doing together, what the main conflict in the story is, etc.). This ability to reason about ambiguous symbols in a collaboratively constructed story environment is quite unlike the traditional methods used for story representation

in the computational world of interactive narrative (e.g. planning operators, beats, story graphs, etc.).

This paper presents the current formalism that the *Digital Improv Project* uses to handle ambiguity and construct shared mental models. This formalism is based in Lakoff's concepts of categories and human cognition (Lakoff 1989) and the computational approaches employed in fuzzy logic (Bellman and Zadeh 1970; Alexander 2002). This paper presents this formalism within the context of a working digital improv installation called *Party Quirks*.

A typical game of *Party Quirks* involves four players: one who plays the role of party host, and three others who play as party guests. When the game begins, the host briefly leaves the room, at which point each party guest is assigned a "quirk" – some special trait for each guest that is public knowledge to the guest actors and the audience but not the party host. The host player then returns and, within the context of hosting a party, aims to figure out what quirk each guest is portraying through their interactions. A guest typically leaves the scene when the host has successfully guessed their quirk. The game ends when there are no guests remaining or when too much time has passed. In the case of time running out, the presenter of the improv show may prompt the party host to guess the quirks of any remaining guests.

The digital version of this improv game consists of software agents acting independently to emulate the communication processes and reasoning about ambiguity that live actors demonstrated during performances in our empirical studies (see Magerko et al. 2009 for an overview of the empirical methods used). A human controls the host in the virtual scene by using a menu-based system on an iPad, which allows the user to stand and physically take part in the virtual performance. The iPad's touchscreen also enables buttons to be dynamically labeled, reducing the complexity of the interface by providing only the interactions needed by the user at a given time. This paper explains the underlying reasoning done by the AI actors, provides an example of a system run, and reviews evaluations of the work.

Modeling Character Prototypes

The authoring for a digital version of *Party Quirks* involves the creation of *character prototypes* (e.g. *Cowboy*) as possible quirks to portray. A *prototype* refers to an idealized, socially recognizable construct that maps to a certain kind of character (Lakoff 1989). The criteria for selecting prototypes for inclusion were a) general recognizability (e.g. *Cowboy*, *Witch*, and *Mob Boss*), b) distinctiveness (i.e. *Town Drunk* and *Mad Scientist* have relatively little in common), and c) potential for ambiguous overlap of attributes (e.g. *Pirate*, *Knight*, and *Ninja* all use swords).

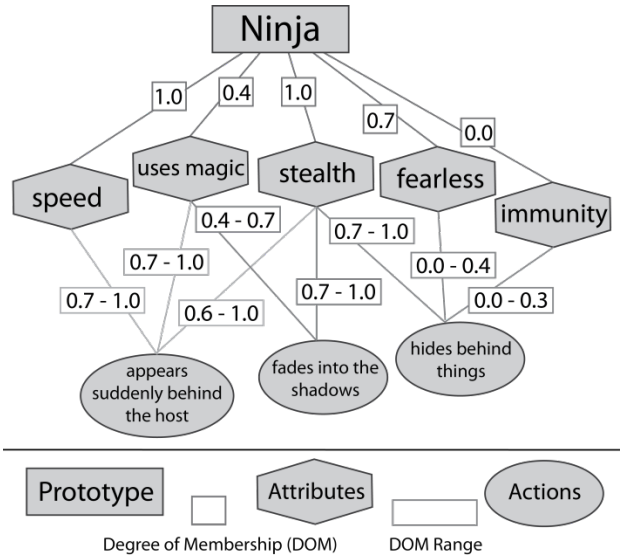

Figure 1. Degrees of membership between prototypes, attributes, and actions. *Ninja* has very strong (1.0) membership in *speed* and medium (0.4) membership in *uses_magic*. A *Ninja* can execute the *fades into the shadows* action because it is available to prototypes that have a *uses_magic* value between 0.4 and 0.7.

Each prototype is defined as a collection of properties with varying degrees of membership (DOM) in sets that represent character attributes (see the top half of Figure 1). For example, *Ninja* has a low value for *clumsiness* but a high value for *sword_use*, whereas *Town Drunk* has opposite values. This approach is similar to how we have seen portrayals of prototypes in our human data and matches well to contemporary thoughts on how humans categorize fuzzy concepts (Lakoff 1989; Rosch and Lloyd 1978). This layer of abstraction is important because expressive actions often imply DOM for multiple attributes (e.g. whether a sword is used in a clumsy or adept manner communicates multiple association values.

Attributes are adjectives that define a prototype. *Actions* are subsequently the physical acts that are used to communicate attributes and are associated with at least one <attribute, DOM range> pair (see the bottom half of Figure 1). For example, <*uses_magic*, 0.7-1.0> implies a high association with magic usage, which is connected to the action *appearSuddenly*. This same action also has a high association with the attribute *moves_stealthily*, and represents the range <*moves_stealthily,* 0.7-1.0>. Any character with *uses_magic* or *moves_stealthily* between these values can therefore execute the *appearSuddenly* action on stage. This provides both a modular, data-driven approach to authoring character behaviors and a shared resource for all characters to draw from. Actions are authored with four components:

1. 1. The name of the action
2. The set of attributes and attribute ranges that this action is appropriate for (e.g. *swashbuckling*

action would be associated with <*uses_sword*, 0.7-1.0>)
3. Environmental requirements and consequences (discussed below)
4. Dialogue associated with performing this action
5. The visual animation for acting in the virtual environment

Prototype / attribute information is encoded in a Google Docs spreadsheet, which is read into the system at run-time. This allows us to easily change DOM values, add actions, reassign animations, etc., without touching the *Party Quirks* codebase.

We use a series of Boolean variables to describe the state of the environment in a scene. However, unlike normal boolean variables, their values are not always known. Party Quirks, like much of improve, operates under an *open world assumption* similar to the approach used in (Swartjes 2010), which states that before a variable is observed its value is unknown. In this context, the open world assumption means, if a particular aspect of the environment has not been explicitly stated to be true or false, an agent can observe it to be whichever value it prefers as required. Actions can thus have requirements and consequences for the environment. For example, the action *feedAnimal* has the prerequisite *AnimalExists*, meaning it must either already have been explicitly stated that animals are in the scene or nobody can have mentioned animals at all, allowing the agent to introduce an animal into the scene without disrupting any previous statements. After the agent has executed the action, the *AnimalExists* variable changes to true, alerting the other agents to the existence of an animal (as implemented, these variables are currently shared between agents though future plans intend for them to be part of each agent's individual knowledge base).

Character portrayal is currently solely based on the relationship between prototypes, attributes, and actions that portray attributes. This does not currently address issues in goal-based behavior, emotive performance, narrative reasoning, etc., mainly because our work in improvisational characters is iterative. We explicitly build small agents as a means of providing a computational lens for better understanding our data. Building these agents provides for a cyclic process of data collection / analysis, building agents, and a repeat of this process with a better understanding of what to look for in our data or what kind of data we need more of, which in turn enables us to build more complex agents, etc. (Magerko, Fiesler, and Baumer 2010). We chose *Party Quirks* as our initial domain because it lacks complicating factors (such as narrative development) that make building a working agent too monolithic and difficult without first understanding the myriad working parts that go into creating improvising agents.

Ambiguity on Stage

Calculating Ambiguity

The primary benefit of using fuzzy membership of sets is that it captures the ambiguity inherent in improvisational theatre. For instance, if an actor comes onto stage and makes takes a long drink out of a make believe bottle, that actor may be thinking about portraying a "town drunk," but that action is open to interpretation to other actors involved in the scene. They could reasonable interpret him to be a pirate, rock star, or perhaps a politician. In other words, knowledge presented on stage can have varying degrees of concreteness to them and that can be used by improvisers to misinterpret an actor's intentions, take a scene in a surprising direction, or even intentionally push an actor in a different direction than they had intended.

The ambiguity of a particular value for an attribute represents how easy it is to determine the agent's prototype from that <attribute, value> pair. As such, the ambiguity of a pair is dependent on how far the value is from the average value for that attribute (how easy it is to tell it is unusual) and how many other prototypes have a similar value (how many possible prototypes could it be confused with). After calculation, it is normalized according to the other values for the prototype to allow for easy comparison. We define the ambiguity of a data point n for the set N of all values of a given attribute as:

$$A_n = (1 - |\text{avg}(N) - n|) / (|\text{all values}| - |\text{values within 1 } \sigma|)$$

Normalization step:

$$NA_n = (A_n - \min(A)) / (\max(A) - \min(A))$$

In other words, the ambiguity of an attribute value n is (1-absolute value of the average of all values for the attribute – n) / (the set of all values for the given attribute – the set of values for the attribute within one standard deviation, i.e. the number of values outside one standard deviation of n). This value is then normalized according to the values for the prototype's other attributes (the smallest value according to the above equation becomes 0, the largest becomes 1, and the values in between are assigned according to their distance between the minimum and maximum).

An action's ambiguity is based on the number of attributes it is associated with and the number of prototypes with appropriate values. For example, *appearSuddenly* has two associated attributes (*uses_magic* and *moves_stealthily*) and six prototypes with appropriate values for at least one of those attributes. The formula we use for calculating action ambiguity is simply (number of associated attributes) * (number of relevant prototypes).

Decision Making Based on Ambiguity Values

Actors in *Party Quirks* can reason about this ambiguity to give the host a natural path of discovery for the scene. Our agents reason about this by comparing authored data to determine the relative ambiguity of associations with each prototype. For example, most characters have a low degree of membership for *bites_people,* so this information is very ambiguous and does little to clarify an actor's quirk. Conversely, because only a few prototypes (*Kindergartner* and *Caveman*) have a high DOM for *bites_people*, that high DOM information has low ambiguity and can steer the host to fewer potential prototype matches. Therefore, an agent portraying the *Caveman* prototype may avoid executing an action associated with the *bites_people* attribute early on since it represents a very unique <attribute, value> pair. Portraying it too early would make it easy for the host to guess *Caveman*, causing that actor to exit early in the scene.

The calculated ambiguity values also provide the means to determine how much the host's interactions indicate their convergence with the "reality" of the scene. In other words, the actions that a human host executes indicate how close they are to guessing a guest's quirk (i.e. building a shared mental model or reaching cognitive consensus (Fuller and Magerko 2011)).

Users, playing as the host, have the opportunity to interact with the agents. They can guess the agent's prototypes, ask questions or make assumptions about individual attributes, prompt the guest with information about the scene, or ask for help in various forms. Whenever the host interacts with a guest agent, the agent compares what the host did to the corresponding DOM values. For example, the prototype *Town Drunk* has a DOM of 0.5 for *facial_hair*. If the host asks if a guest with the prototype *Town Drunk* has a middle value for the attribute *facial_hair*, the agent sees the host's assumption is correct and thus notes the interaction as being convergent, indicating the host is on the right track to guessing the prototype. The convergence function also accounts for the ambiguity of the different attributes. For example, if the host asks if the *Town Drunk* has a low value for *sword_use*, while the assumption is correct, the value is common, and thus less convergent than if the question had been for a high value of *clumsiness*, which also correct but less common. This means interactions that give more information are more convergent, as they show the host is closer to the correct answer. It normalizes these factors to a value representing how well the host's action converges with the guest's actual quirk, with higher values indicating the host is on the right track and lower values indicating the host needs more help.

Convergence also plays a role in the kind of feedback the host receives. When the host interacts with an agent, an "applause" sound effect plays appropriate to the convergence value. A lot of applause indicates the host is close, whereas little to no applause lets the host know they may not be guessing correctly. The decision to incorporate this feedback element was based directly on feedback given during public demo sessions of an early *Party Quirks* prototype.

Character Portrayal Strategies

Agents can reason about the type of behaviors their character should display to represent their prototype. They have several options derived from our empirical observations of improv actors (Fuller and Magerko 2011) for how they can demonstrate their character's quirk. At the moment, these techniques can either be pre-selected by whoever is running the system or randomly selected at runtime. We have not been able to reliably discern the heuristic information that improvisers use for selecting from these strategies.

In one common technique, a guest presents ambiguous clues early on and gets more specific with time, which we call *reverse scaffolding*. The purpose of *reverse scaffolding* is to keep a scene interesting; there would be little point to the game if the host guessed the guest's quirk immediately, so actors like to start with less obvious clues to keep the host guessing (note: this process is alluded to in the earlier section). However, the game is also more satisfying when the host is successful, so actors tend to get more obvious over time, giving very specific clues near the end of the scene.

Another technique, usually chosen for more humorous purposes, is *caricature*, where a guest takes the quirk they have been assigned and exaggerates it as much as possible, creating more comedic situations. This involves selecting very low ambiguous actions at the onset. Computationally, we accomplish this technique by reducing the pool of potential attributes to the least ambiguous, then selecting only the least ambiguous action for each attribute.

Finally, guests may choose to take an alternative approach to portraying their character and *oppose* a key attribute, which computationally means the opposed attribute's DOM value is inverted (the new value is equal to 1.0 minus the old value), the guest chooses the most unambiguous actions when portraying the attribute, and the attribute is chosen for presentation (i.e. a display to the host) more often to emphasize it. One example of this technique is a character with the quirk *Pirate* who chooses to portray it by behaving according to the prototype except with a low value of *sword_use* instead of the normally characteristic high value.

Once an agent has decided how to portray its quirk (either by random selection or having it pre-determined), it will join the scene and make offers of information to encourage the host to guess the prototype. It does this by

choosing and executing actions, with or without prompting from the host. The decisions an agent makes are dependent on whether or not the host has directly interacted with it, as described below.

If a host has not spoken directly to the agent, the agent may choose to continue performing its idle behavior (basically just walking around) or to execute an action anyway. Presenting an action in this case represents a natural behavior for the character prototype, which is something a character would do without provocation. The algorithm for such a situation is as follows:

1. The agent considers its prototype and the DOM values it has for each attribute compared to the DOM values other prototypes have for those attributes. It calculates how ambiguous its values are for each attribute, then disregards the most ambiguous attributes (the ones which, essentially, have nothing to do with defining the prototype).
2. Once it has decided which attributes are important, the agent looks at the <attribute, DOM range> pairs defining each action in order to create a pool of possible actions – that is, every action it can do where the value for one of its important attributes falls into the range specified by that action (see the Modeling Character Prototypes section above for a definition of *actions*). If the agent is portraying a caricature of its prototype, it selects only the least ambiguous actions for each relevant attribute.
3. If the agent is not reverse scaffolding, it chooses an action probabilistically according to how recently each has been executed (the more time has passed since an action has been presented, the more likely it is to be chosen again). If the agent is reverse scaffolding, however, it bases its action selection on the ambiguities of the actions. First, it orders the actions in order of decreasing ambiguity. The agent calculates how much time it has spent on stage compared to the amount of time left in the scene, then applies that proportion to the number of actions in the list (so, if three minutes have passed in a five minute scene and there are ten actions in the list, it would pick the sixth position). It probabilistically selects an action in the pool, with the probability of each action being chosen based on a Gaussian distribution centered on the position designated by the proportion.

Agents also account for the current state of the environment. Any action which has a requirement that is currently opposed (e.g. if an action requires food in the scene, but *FoodExists* has been observed as False) is not allowed for consideration. Also, the probabilities of actions that have environmental requirements are adjusted according to how recently the relevant environmental variable has been observed; the more recent the observation, the more likely the action is to be selected. This simulates a recency effect in which people are more likely to remember and/or react to more recent pieces of information than older ones.

If the host has indeed prompted the guest with an interaction of some sort, the agent's response depends on the nature of the host's communication. The host can ask the guest for their value for a particular attribute, which is called a *Targeted Offer* based on our research on shared mental models (Fuller and Magerko 2011), such as asking "What do you think about using swords?" In this case, the agent responds with an action associated with their value for the requested attribute. If the host attempts to verify their concept of a guest's value for a particular attribute, called a *Verification*, the agent responds with a statement confirming or denying the assertion and a presentation to demonstrate its actual value. For example, if the guest agent has the prototype *Terminator* and the host makes the statement "I think you like to eat a lot," the guest might respond with "No, I do not require food" and a refusal animation. The host may try to confirm the most recent presentation the guest made as being associated with a particular attribute (called a *Confirmation*), in which case, again, the agent will answer with a confirmation or denial and a presentation (i.e. a display to the host) for the relevant attribute. In some cases, the host may get frustrated and ask for better clues, called a *Clarification Request*, and in response the guest will shrink the possible action pool to be less ambiguous and make a new presentation. The host can alter the environment if they wish by making an *Environmental Offer* about one of the various environment variables, in which case the guest agent will respond with an action relating to the new variable value if they can and a normal action if they cannot. Finally, when the host thinks they know the guest's quirk, they can make a guess. If the guess is correct, the agent acknowledges the host's success and exits the scene. If, however, the guess was incorrect, the agent considers its prototype in relation to the guessed one. It selects an attribute that is both significantly different between the two prototypes and significantly un-ambiguous for the correct prototype and then presents an action associated with the chosen attribute (e.g. The guest says, "No, I'm not a ninja," then knocks something over to demonstrate clumsiness). This indicates that the guess was incorrect and gives justification for the difference.

All of these responses to the host also consider the ambiguity of the possible options in relation to the ambiguity of what has already happened in the scene. The agent attempts to choose an action that is less ambiguous if at all possible in order to push the host towards the correct conclusion. If the previous actions are less ambiguous than any of the options for response (such as if the agent is using *caricature*, where all presentations are as

unambiguous as possible), the agent picks the least ambiguous.

Multiple-Agent Portrayals

In our studies of improv actors, actors occasionally helped emphasize another guest's qualities in order to help the host. These interactions name the helped guest directly and target an attribute very specific to that guest's prototype, so as to give as much aid as possible. We call these specific joint behaviours *guest-to-guest interactions* since they involve interactions between a pair of guests to portray information about a single guest. Guest-to-guest interactions in our *Party Quirks* system are authored as joint dialogue and animation instances that can be triggered when the host is struggling. Two general situations tend to trigger real-life guest-to-guest interactions, which we have modelled. The first occurs when one guest has not interacted with the host for an extended period of time, which often happens if the host focuses on one actor instead of interacting with everyone on stage. The second situation happens when the game is nearly over, the host is focusing on one actor, and the host-guest interactions are significantly divergent from that actor's quirk.

Example

Below is a simplified scenario involving a single guest (multiple guests make the example too long for the constraints of publishing this article) as a demonstration of how users interact with the agents:

The stage on screen shows no guests present until the user presses "Be the Host" on the iPad to begin. The guest agent then appears on the projected virtual stage. The guest has been randomly assigned a prototype, in this case *Pirate*. The agent chooses to use reverse scaffolding to portray its character.

The user is presented with buttons on the iPad to ask the virtual guest about attributes (e.g. *attractiveness, fearlessness*, etc.). The user decides to defer, waiting until the virtual guest volunteers information before asking the guest something.

The virtual guest pretends to be playing cards. Text appears under the stage, saying, "I've got a full house. Read'em and weep." This action provides information about multiple attributes, including playfulness and willingness to gamble.

The user, to confirm whether this previous action was about gambling (implying a prototype such as *Mob Boss*) rather than playfulness (e.g. *Kindergartener*) chooses to ask about the attribute *gambles*. On the iPad, the user selects to ask a question about the guest's gambling activities, and asks whether the guest gambles a lot.

The guest gestures aggressively, with text underneath, "Yes. I won that money fair and square. Well, sort of..." This communication includes two components: confirmation that the host was correct, and a new action conveying related information. A small amount of applause plays, indicating 'Gambles a lot' is converging on the guest's prototype but that there are also attributes more relevant to this guest that the host could talk about.

The user chooses *Guess Identity* on the iPad to see which prototypes might gamble a lot. These options are displayed:

Wizard	*Knight*	*Ninja*
Pirate	*Terminator*	*Sumo Wrestler*
Superhero	*Mob Boss*	*Town Drunk*
Witch	*Princess*	*Normal Person*
Caveman	*Kindergartner*	*Grandmother*
Cowboy	*Mad Scientist*	*Alien Invader*

Suspecting that *Town Drunk, Pirate, Mob Boss*, and *Cowboy* would be most likely to gamble, the user thinks of an attribute to narrow down that list. Under *Possessions* in the iPad menu, *Sword* is an option, which the user recognizes as unique to *Pirate*. The user follows the menus to inquire whether the virtual guest often uses a sword.

In response, the guest moves both hands to an imagined hilt. Text below clarifies, "Yes. Make one wrong move and I'll finish you off." The host hears a burst of applause because this interaction was highly significant to the guest's prototype and has a high convergence value.

The user returns to *Guess Identity*, and guesses *Pirate*.

Because the guess is correct, the virtual guest leaves stage. The text, "That is correct!" is displayed on-screen, and applause plays to confirm the host's guess.

Evaluation

The evaluation of the *Party Quirks* prototype has been a difficult process, as is inherent for interactive narrative systems. An evaluation of an interactive narrative system, when it occurs, typically takes the form of lesion experiments (e.g. "our system does A, so we provided the system with and without A to study participants and measured the qualitative and / or qualitative differences") or the subjective measure of public acceptance (e.g. "we have over 5,000 downloads of our system). The most common approach to evaluation is unfortunately to avoid the issue altogether, due to the complex issues in even creating a working, full-fledged system.

Evaluation of the *Party Quirks* prototype was done in two stages. The first stage of evaluation focused on usability and interface testing. There were multiple issues at play when building this system, such as getting novices accustomed to playing the improv game, using an iPad to help the novices interact in a more naturalistic setting (e.g.

standing and facing the virtual actors on a projection), having an appropriate representation of shared mental model moves for the user to select from, and having users actually attend to what was occurring on the projected virtual stage as opposed to continuously staring at the iPad.

The second stage of evaluation was the formal submission of the system to the Chicago Improv Festival (CIF), a 13-year-old festival that hosts professional improv troupes from all over the world. Following the same guidelines as any other improv troupe, we submitted a video of lab members using the *Party Quirks* application. *Party Quirks* was formally accepted based on a review by the experts judging the submissions and was presented as a CIF performance at the ComedySportz Theatre as a three-day installation.

The evaluation of submitting to an improv festival has both its merits and drawbacks. The positive side of this evaluation was that it was put through the same rigors as a human improv troupe for acceptance into a major theatre festival. This has a high benchmark for quality, compared to the number of downloads for a system, for instance, which says nothing about how much people enjoyed it and what their backgrounds for judging works is. On the other hand, the result of this kind of evaluation is highly uninformative. It produces a Boolean result of "Accepted" or "Not accepted," without any further detail as to why the result occurred. At this stage of the work, which is still in a preliminary stage to buttress future work, we are satisfied with the outcome as we prepare for larger scale systems.

The usability issues that we tested are of particular importance as interaction with virtual characters, especially without an avatar, is particularly challenging. We are dedicated to avoiding a "point and click" approach because we are invested in creating virtual theatre experiences that mimic real-world improvisation with human actors. However, as we build more complex agents, the medium that the performances take place in will be heavily dictated by the kinds of interactions that medium affords.

Discussion

Our implementation of the improv game *Party Quirks* has led to the creation of a new kind of digital game: a mixed-initiative theatre game where AI and humans can actively participate in an improv game together. While we are encouraged by the initial work done in *Party Quirks* in representing character prototypes, the process of building shared mental models, and an initial communication framework for interacting with improvisational characters, this initial system is not without its drawbacks. Users of the system at the Chicago Improv Festival were generally pleased and excited to use the system. However, users often got stuck just guessing repeatedly instead of making use of the other moves common in performances. This points to a major issue of presence in the system – users do not act like they are performing with the actors on a virtual stage, but like they are prodding a system to see how it responds. The virtual actors give an often-entertaining response with any guess, which provokes the user to guess again instead of selecting other moves. Future work in interface design, such as using voice commands or gesture recognition, may help actively involve the user in the performance space rather than acting outside of it and getting stuck in the most convenient menu option.

One common misconception of the *Party Quirks* system is that it is an isomorph to AI approaches to *20 Questions*, the game where one player answers "yes" or "no" questions from one or more people who are trying to guess the object they have in mind. AI approaches to this problem generally involve optimal decisions about information gain. Our approach to cognitive convergence in *Party Quirks*, and in digital improvisation in general, a) is not concerned with optimality, to the extent that agents will portray actions that "push" the human host in a direction without jumping straight to a solution, b) employs the *reverse scaffolding* strategy of being vague early on in a scene and more heavy-handed as time goes by (in general), and c) our scenario involves the human doing the guessing as opposed to the AI, though one could easily conceive switching roles.

We have considered building AI that plays as the host, letting a human user or users play as guests. While that may make the *Party Quirks* experience a more complete one, the system's purpose is still to serve as a prototype for iterative research on building improvisational agents. This system was built to explore the process of shared mental models and character portrayal, which we have done. Future efforts will be focused on applying what we have learned to the construction of more complex agents.

The agents themselves are fairly generalizable in terms of the number of attributes that can be used to describe characters and the different mappings from attribute value ranges to actions that can occur. This does represent an authoring bottleneck, but one that can potentially be resolved with the use of crowdsourcing techniques such as Amazon's *Mechanical Turk*. We are currently running a crowdsourcing data collection to help populate our description of the story elements related to a Western-themed story world called *TinyWest*.

The main limitation with the definition of character portrayals as they currently exist is that they are not mutable. The only characters the agent can be portray in a scene are the ones for which a prototype has already been fully authored. Prototypes cannot be altered, augmented, or combined. For instance, prototypes cannot be blended together to create new prototypes (e.g. a mosquito that acts like a drunk when it drinks blood) nor can they be created with some antithetical property (e.g. a plumber who is afraid of water). Our initial work focused on alternate portrayal techniques like *negation*, where a prototype with an extreme attribute value has that value inverted, which

relies on authoring an "afraid of water" attributes to create "a plumber who is afraid of water." However, even if that approach is satisfactory, it does not answer how to create amalgams of different prototypes, like *mosquito* and *drunk*. This points to the need for future work to focus on how agents can employ the process of *conceptual blending* (Fauconnier and Turner 2003).

Another major limitation of these improv agents is that they have no concept of narrative. They are incapable of constructing a story or having dialogue acts that logically progress over time. One possible approach to this would be to create joint plans that could be selected and performed based on user actions, as seen in *Façade* (Mateas and Stern 2002), but that would hinder our goal of equal co-creation between humans and AI agents (i.e. the computer would have privileged pre-authored story knowledge rather than both the computer and human starting off as equals in the story creation process). The narrative limitations of the *Party Quirks* agents have fueled our current research agenda of exploring conceptual models of equal mixed-initiative collaborative story construction (i.e. AI and humans are both on the virtual stage and equally share responsibility in constructing the story). Continuing research explores how agents can set up the initial elements of a scene (e.g. where the scene takes place, who the characters are, what joint activity they are doing together, etc.) and how agents can find the "tilt" for the scene (i.e. the main dramatic focus of the scene). This work directly builds on what we have learned from building the *Party Quirks* installation in terms of how to interact with agents, the fuzzy knowledge formalism for representing ambiguity in the world, and the construction of support tools for creating animated improvisational agents. The future of this work will be a synthesis of these lessons learned from *Party Quirks*, resulting in a troupe of synthetic improvisers than can jointly construct narratives on stage with or without a human equal acting with them.

The fuzzy approach to agents for Game AI in general has yielded promising avenues for reuse. While we have currently applied it to the prototype definition of characters, we have preliminarily found that it fits well for prototypes of the other elements of a scene, such as character relationships, joint activities, and the associations between scene elements, such as the association between motions on stage and the semantic actions those motions are associated with. We have also begun to explore its use in non-improv related settings that deal with semantic ambiguity, such as the surrealistic guessing game *DixIt*, which involves providing clues to other game players that are ambiguous but not *too* ambiguous so that some people (i.e. at least one player) understand the clue but not everyone. Our model of fuzzy semantic descriptions and communication based on ambiguity levels seems perfect for this kind of environment and potentially other game situations that involve communicating fuzzy concepts.

Acknowledgements

This work was funded by the NSF Grants IIS 0757567, 1036457, and 1129840.

References

Alexander, Thor. 2002. An Optimized Fuzzy Logic Architecture for Decision-Making. In *AI Game Programming Wisdom*, ed. Steve Rabin, 367-374. 1st ed. Charles River Media.

Bellman, R. E., and L. A. Zadeh. 1970. "Decision-Making in a Fuzzy Environment." *Management Science* 17 (4) (December): B141-B164.

Fauconnier, Gilles, and Mark Turner. 2003. *The Way We Think: Conceptual Blending and the Mind's Hidden Complexities*. Basic Books.

Fuller, D., and B. Magerko. 2011. Shared Mental Models in Improviational Theatre. In Atlanta, GA.

Harger, Brenda. 2008. Project Improv. *Project Improv*. http://www.etc.cmu.edu/projects/improv/, accessed 8/1/2011.

Hayes-Roth, B., and R. Van Gent. 1996. Story-Making with Improvisational Puppets and Actors. In *Technical Report KSL-96-09*. Palo Alto, CA: Stanford University.

Lakoff. 1989. Cognitive models and prototype theory. In *Concepts and conceptual development*, ed. Ulric Neisser, 63-100. CUP Archive.

Magerko, B., C. Fiesler, and A. Baumer. 2010. Fuzzy Micro-Agents for Interactive Narrative. In *Proceedings of the Sixth Annual AI and Interactive Digital Entertainment Conference*. Palo Alto, CA: AAAI Press.

Magerko, B., W. Manzoul, M. Riedl, A. Baumer, D. Fuller, K. Luther, and C. Pearce. 2009. An Empirical Study of Cognition and Theatrical Improvisation. In *Proceeding of the Seventh ACM Conference on Creativity and Cognition*, 117–126.

Mateas, M., and A. Stern. 2002. "A Behavior Language for Story-Based Believable Agents." *IEEE Intelligent Systems* 17 (4): 39-47.

Perlin, Ken, and Athomas Goldberg. 1996. Improv: A System for Scripting Interactive Actors in Virtual Worlds. In *SIGGRAPH \uc0\u8217{}96*. New Orleans, LA.

Rosch, E., and B. Lloyd. 1978. *Principles of Categorization, Cognition and Categorization*. Erlbaum, Hillsdale NJ.

Swartjes, I. 2010. Whose Story is it Anyway? How Improv Informs Agency and Authorship of Emergent Narrative. Enschede, The Netherlands: University of Twente.